AIX Version 6.1

Commands Reference, Volume 6, v - z

SC23-5248-02

AIX Version 6.1

Commands Reference, Volume 6, v - z

SC23-5248-02

Note
Before using this information and the product it supports, read the information in Appendix C, “Notices,” on page 315.

Third Edition (October 2009)

This edition applies to AIX 5L Version 5.3 and to all subsequent releases of this product until otherwise indicated in new editions.

A reader’s comment form is provided at the back of this publication. If the form has been removed, address comments to Information Development, Department 04XA-905-6B013, 11501 Burnet Road, Austin, Texas 78758-3400. To send comments electronically, use this commercial Internet address: pserinfo@us.ibm.com. Any information that you supply may be used without incurring any obligation to you.

Contents

About This Book vii
How to Use This Book vii
ISO 9000 ix
32-Bit and 64-Bit Support for the Single UNIX Specification x
Related Information x

Alphabetical Listing of Commands 1
vacation Command 1
val Command (SCCS) 3
varyoffvg Command 4
varyonvg Command 5
vc Command 9
vgrind Command 11
vi or vedit Command 13
view Command 31
vmh Command 31
vmo Command 33
vmstat Command 38
vpdadd Command 51
vpddel Command 52
vsdatalst Command 53
vsdchgserver Command 55
vsdelnode Command 56
vsdelvg Command 58
vsdnode Command 59
vsdsklst Command 61
vsdvg Command 63
vsdvgts Command 65
w Command 66
wait Command 67
wall Command 68
wallevent Command 69
watch Command 70
wc Command 72
what Command 74
whatis Command 75
whatnow Command 76
whereis Command 80
which Command 81
which_fileset Command 81
who Command 83
whoami Command 86
whodo Command 86
whois Command 88
whom Command 89
wlmassign command 91
wlmcheck command 93
wlmcntrl Command 95
wlmmon and wlmperf Commands 98
wlmstat Command 102
wol command 107
wparexec Command 108
write Command 115

writesrv Daemon 119
wsm Command 119
wsmaccess Command 121
wsmserver Command 122
wtmpfix Command 123
wump Command 124
X Command 125
x_add_fs_fpe Command 137
x_add_nfs_fpe Command 138
x_rm_fpe Command 139
xargs Command 140
xauth Command 143
xclock Command 146
xcmsdb Command 148
xdm Command 149
xfindproxy Command. 163
xfs Command 164
xget Command 166
xhost Command 168
xinit Command 169
xkbcomp Command 171
xkbevd Daemon 172
xkbprint Command 174
xlock Command 175
xlsfonts Command 177
xmbind Command. 178
xmkmf Command 179
xmwlm Command 180
xmodem Command 181
xmodmap Command. 183
xmpeek Command 185
xmscheck Command. 187
xmtopas Command 188
xntpd Daemon 189
xntpdc Command 192
xpr Command 200
xpreview Command 202
xprofiler Command 204
xrdb Command 206
xsend Command 209
xset Command 210
xsetroot Command 214
xss Command 215
xstr Command 216
xterm Command 217
xwd Command 241
xwud Command 242
yacc Command. 244
yes Command 246
ypbind Daemon. 247
ypcat Command 248
ypinit Command 249
ypmatch Command 251
yppasswd Command. 252
yppasswdd Daemon 253
yppoll Command 255

yppush Command 256
ypserv Daemon. 257
ypset Command 258
ypupdated Daemon 259
ypwhich Command 260
ypxfr Command 262
zcat Command 264
zdump Command 265
zic Command 266

Appendix A. Command Support for Files Larger than 2 Gigabytes 271
Commands That Do Not Support Files Larger Than 2 Gigabytes 271

Appendix B. Functional List of Commands 273
Communications 274
Commands List: Message Handler. 277
Files and Directories 286
General Operations 294
Commands List: Numerical Data 302
Commands List: Performance Tuning. 302
Programming Tools 311

Appendix C. Notices 315
Trademarks 316

Index 319

About This Book

This book provides end users with complete detailed information about commands for the AIX operating system. The commands are listed alphabetically and by category, and complete descriptions are given for commands and their available flags. If applicable, each command listing contains examples. This volume contains AIX commands that begin with the letters v through z. This publication is also available on the documentation CD that is shipped with the operating system.

How to Use This Book

A command is a request to perform an operation or run a program. You use commands to tell the operating system what task you want it to perform. When commands are entered, they are deciphered by a command interpreter (also known as a shell) and that task is processed.

Some commands can be entered simply by typing one word. It is also possible to combine commands so that the output from one command becomes the input for another command. This is known as pipelining.

Flags further define the actions of commands. A flag is a modifier used with the command name on the command line, usually preceded by a dash.

Commands can also be grouped together and stored in a file. These are known as shell procedures or shell scripts. Instead of executing the commands individually, you execute the file that contains the commands.

Some commands can be constructed using Web-based System Manager applications or the System Management Interface Tool (SMIT).

Highlighting

The following highlighting conventions are used in this book:

Bold	Identifies commands, subroutines, keywords, files, structures, directories, and other items whose names are predefined by the system. Also identifies graphical objects such as buttons, labels, and icons that the user selects.
Italics	Identifies parameters whose actual names or values are to be supplied by the user.
`Monospace`	Identifies examples of specific data values, examples of text similar to what you might see displayed, examples of portions of program code similar to what you might write as a programmer, messages from the system, or information you should actually type.

Format

Each command may include any of the following sections:

Purpose	A description of the major function of each command.
Syntax	A syntax statement showing command line options.
Description	A discussion of the command describing in detail its function and use.
Flags	A list of command line flags and associated variables with an explanation of how the flags modify the action of the command.
Parameters	A list of command line parameters and their descriptions.
Subcommands	A list of subcommands (for interactive commands) that explains their use.
Exit Status	A description of the exit values the command returns.
Security	Specifies any permissions needed to run the command.
Examples	Specific examples of how you can use the command.
Files	A list of files used by the command.
Related Information	A list of related commands in this book and related discussions in other books.

Reading Syntax Statements

Syntax statements are a way to represent command syntax and consist of symbols such as brackets ([ ]), braces ({ }), and vertical bars (|). The following is a sample of a syntax statement for the **unget** command:

unget [-**r***SID*] [-**s**] [-**n**] *File* ...

The following conventions are used in the command syntax statements:

- Items that must be entered literally on the command line are in **bold**. These items include the command name, flags, and literal charactors.
- Items representing variables that must be replaced by a name are in *italics*. These items include parameters that follow flags and parameters that the command reads, such as *Files* and *Directories*.
- Parameters enclosed in brackets are optional.
- Parameters enclosed in braces are required.
- Parameters not enclosed in either brackets or braces are required.
- A vertical bar signifies that you choose only one parameter. For example, [a | b] indicates that you *can* choose a, b, or nothing. Similarly, { a | b } indicates that you *must* choose either a or b.
- Ellipses (...) signify the parameter can be repeated on the command line.
- The dash (-) represents standard input.

Listing of Installable Software Packages

To list the installable software package (fileset) of an individual command use the **lslpp** command with the **-w** flag. For example, to list the fileset that owns the **installp** command, enter:

```
lslpp -w /usr/sbin/installp
```

Output similar to the following displays:

```
File                            Fileset                 Type
-----------------------------------------------------------------------
/usr/sbin/installp                bos.rte.install           File
```

To list the fileset that owns all file names that contain `installp`, enter:

```
lslpp -w "*installp*"
```

Output similar to the following displays:

```
File                            Fileset                 Type
-----------------------------------------------------------------------
/usr/sbin/installp                bos.rte.install           File
/usr/clvm/sbin/linstallpv         prpq.clvm                 File
/usr/lpp/bos.sysmgt/nim/methods/c_installp
                                  bos.sysmgt.nim.client     File
```

Running Commands in the Background

If you are going to run a command that takes a long time to process, you can specify that the command run in the background. Background processing is a useful way to run programs that process slowly. To run a command in the background, you use the & operator at the end of the command:

```
Command&
```

Once the process is running in the background, you can continue to work and enter other commands on your system.

At times, you might want to run a command at a specified time or on a specific date. Using the **cron** daemon, you can schedule commands to run automatically. Or, using the **at** and **batch** commands, you can run commands at a later time or when the system load level permits.

Entering Commands

You typically enter commands following the shell prompt on the command line. The shell prompt can vary. In the following examples, $ is the prompt.

To display a list of the contents of your current directory, you would type `ls` and press the Enter key:

```
$ ls
```

When you enter a command and it is running, the operating system does not display the shell prompt. When the command completes its action, the system displays the prompt again. This indicates that you can enter another command.

The general format for entering commands is:

Command Flag(s) Parameter

The flag alters the way a command works. Many commands have several flags. For example, if you type the **-l** (long) flag following the **ls** command, the system provides additional information about the contents of the current directory. The following example shows how to use the **-l** flag with the **ls** command:

```
$ ls -l
```

A parameter consists of a string of characters that follows a command or a flag. It specifies data, such as the name of a file or directory, or values. In the following example, the directory named **/usr/bin** is a parameter:

```
$ ls -l /usr/bin
```

When entering commands, it is important to remember the following:

- Commands are usually entered in lowercase.
- Flags are usually prefixed with a - (minus sign).
- More than one command can be typed on the command line if the commands are separated by a **;** (semicolon).
- Long sequences of commands can be continued on the next line by using the \ (backslash). The backslash is placed at the end of the first line. The following example shows the placement of the backslash:

  ```
  $ cat /usr/ust/mydir/mydata > \
  /usr/usts/yourdir/yourdata
  ```

When certain commands are entered, the shell prompt changes. Because some commands are actually programs (such as the **telnet** command), the prompt changes when you are operating within the command. Any command that you issue within a program is known as a subcommand. When you exit the program, the prompt returns to your shell prompt.

The operating system can operate with different shells (for example, Bourne, C, or Korn) and the commands that you enter are interpreted by the shell. Therefore, you must know what shell you are using so that you can enter the commands in the correct format.

Stopping Commands

If you enter a command and then decide to stop that command from running, you can halt the command from processing any further. To stop a command from processing, press the Interrupt key sequence (usually Ctrl-C or Alt-Pause). When the process is stopped, your shell prompt returns and you can then enter another command.

ISO 9000

ISO 9000 registered quality systems were used in the development and manufacturing of this product.

32-Bit and 64-Bit Support for the Single UNIX Specification

Beginning with Version 5.2, the operating system is designed to support The Open Group's Single UNIX Specification Version 3 (UNIX 03) for portability of UNIX-based operating systems. Many new interfaces, and some current ones, have been added or enhanced to meet this specification, making Version 5.2 even more open and portable for applications, while remaining compatible with previous releases of AIX.
To determine the proper way to develop a UNIX 03-portable application, you may need to refer to The Open Group's UNIX 03 specification, which can be accessed online or downloaded from http://www.unix.org/ .

Related Information

The following books contain information about or related to commands:

- *AIX 5L Version 5.3 Commands Reference, Volume 1*
- *AIX 5L Version 5.3 Commands Reference, Volume 2*
- *AIX 5L Version 5.3 Commands Reference, Volume 3*
- *AIX 5L Version 5.3 Commands Reference, Volume 4*
- *AIX 5L Version 5.3 Commands Reference, Volume 5*
- *AIX 5L Version 5.3 Commands Reference, Volume 6*
- *AIX 5L Version 5.3 Files Reference*
- *Printers and printing*
- *Installation and migration*
- *AIX 5L Version 5.3 AIX Installation in a Partitioned Environment*
- *AIX 5L Version 5.3 Network Information Services (NIS and NIS+) Guide*
- *Performance management*
- *AIX 5L Version 5.3 Performance Tools Guide and Reference*
- *Security*
- *Operating system and device management*
- *Networks and communication management*
- *AIX 5L Version 5.3 Technical Reference: Base Operating System and Extensions Volume 1*
- *AIX 5L Version 5.3 Technical Reference: Base Operating System and Extensions Volume 2*
- *AIX 5L Version 5.3 Technical Reference: Communications Volume 1*
- *AIX 5L Version 5.3 Technical Reference: Communications Volume 2*
- *AIX 5L Version 5.3 Technical Reference: Kernel and Subsystems Volume 1*
- *AIX 5L Version 5.3 Technical Reference: Kernel and Subsystems Volume 2*
- *AIX 5L Version 5.3 Web-based System Manager Administration Guide*
- *Performance Toolbox Version 2 and 3 for AIX: Guide and Reference*

Alphabetical Listing of Commands

vacation Command

Purpose

Returns a message to the sender that the mail recipient is on vacation.

Syntax

vacation [{ **-I** | *User* }] | [{ **-f** *Number* [*Unit*] | *User* }]

Description

The **vacation** command returns a message to the sender of a mail message to notify the sender that the recipient is on vacation. The intended use is in a **$HOME/.forward** file that allows messages to come to you while also sending a message back to the sender.

Note: Sendmail version 8.9.3 and subsequent releases have a security enhancement that will ignore the **.forward** file if *either* of the following conditions exist:

- The **.forward** file has group or world writeable permissions
- Any of **.forward** file's parent directories have group or world writable permissions

If you think that the vacation program is not working because the **.forward** file is being ignored, check the permissions. If you must have group or world writeable permissions on any of the parent directories of the **.forward** file, then set the DontBlameSendmail option in the sendmail configuration file with the appropriate values.

The **vacation** command expects a **$HOME/.vacation.msg** file containing a message to be sent back to each sender. If this file does not exist, the **vacation** command looks for **/usr/share/lib/vacation.def**, a systemwide default vacation message file. It should be an entire message, including any desired headers, such as `From` or `Subject`. By default, this message is sent only once a week to each person who sends mail to you. Use the **-f** flag to change the frequency intervals at which the message is sent. The names of the people who send messages are kept in the files **$HOME/.vacation.pag** and **$HOME/.vacation.dir**. These files are created when the **vacation** command is initialized for your user ID using the **-I** (uppercase i) flag.

If the **-I** flag is not specified, the **vacation** command reads the first line from the standard input for a `From` line to determine the sender. If no text is available from standard input, the command returns an error message. All properly formatted incoming mail should have a `From` line. No message is sent if the `From` header line indicates that the message is from `Postmaster`, MAILER-DAEMON, or if the initial `From` line includes the string-`REQUEST@` or if a `Precedence: bulk` or `Precedence: junk` line is included in the header.

Flags

-I Initializes the **$HOME/.vacation.pag** and **$HOME/.vacation.dir** files. Execute the **vacation** command using this flag before you modify your **$HOME/.forward** file.

-f*Number* **[***Unit* **]** Specifies the frequency interval at which the vacation message is sent. The *Number* parameter is an integer value and the *Unit* parameter specifies a time unit. The *Unit* parameter can be one of the following:

s	Seconds
m	Minutes
h	Hours
d	Days
w	Weeks

Note: The **-f** flag cannot be used with the **-I** flag.

Examples

1. Before you use the **vacation** command to return a message to the sender saying that you are on vacation, you must initialize the **$HOME/.vacation.pag** and **$HOME/.vacation.dir** files. To initialize these files, type:

   ```
   vacation -I
   ```

2. Modify the **.forward** file. For example, Mark types the following statement in the **.forward** file:

   ```
   mark,|"/usr/bin/vacation mark"
   ```

   The sender receives the message that is in the **$HOME/.vacation.msg** file, or if the file does not exist, the default message found in the **/usr/share/lib/vacation.def** file. If neither of these files exist, no automatic replies are sent to the sender of the mail message and no error message is generated. If either of these files exist, the sender receives one vacation message from `mark` per week, regardless of how many messages are sent to `mark` from the sender.

3. If the following entry is contained in your **.forward** file,

   ```
   mark, |"/usr/bin/vacation -f10d mark"
   ```

   The sender receives one vacation message from `mark` every ten days, regardless of how many messages are sent to `mark` from the sender.

4. To create a vacation message that is different from the default vacation message, create the file **$HOME/.vacation.msg** and add your message to this file. The following is an example of a vacation message:

   ```
   From: mark@odin.valhalla (Mark Smith)
   Subject: I am on vacation.
   Delivered-By-The-Graces-Of: the Vacation program
   I am on vacation until October 1. If you have something urgent,
   please contact Jim Terry <terry@zeus.valhalla>.
      --mark
   ```

5. To cancel the vacation message, remove the **.forward** file, **.vacation.dir** file, **.vacation.pag** file, and **.vacation.msg** file from your **$HOME** (login) directory:

   ```
   rm .forward .vacation.dir .vacation.pag .vacation.msg
   ```

Files

$HOME/.forward	Contains the names of people who you want your mail to be forwarded to.
/usr/share/lib/vacation.def	Contains the systemwide default **vacation** message.
$HOME/.vacation.dir	Contains the names of people who have sent mail to you while the **vacation** command was being used.
$HOME/.vacation.msg	Contains your personalized **vacation** message.

$HOME/.vacation.pag	Contains the names of people who have sent mail to you while the **vacation** command was being used.
/usr/bin/vacation	Contains the **vacation** command.

Related Information

The **mail** command, **sendmail** command.

The **.forward** file.

Mail applications and Forwarding mail, Sending a vacation message notice in *Networks and communication management*.

Directories in *Operating system and device management*.

---

val Command (SCCS)

Purpose

Validates SCCS files.

Syntax

val [**-s**] [**-r***SID*] [**-m***Name*] [**-y***Type*] *File* ...

Description

The **val** command reads the specified file to determine if it is a Source Code Control System (SCCS) file meeting the characteristics specified by the accompanying flags. If you specify a **-** (minus) for the *File* value, the **val** program reads standard input and interprets each line of standard input as **val** flags and the name of an SCCS file. An end-of-file character terminates input.

The **val** command displays messages to standard output for each file processed.

Flags

Each flag or group of flags applies independently to each named file. The flags can appear in any order.

-m*Name*	Compares the *Name* value with the SCCS **31** identification keyword in the specified file. For identification keyword information, see the **get** command.
-r *SID*	Specifies the SID of the file to be validated. The SID must be valid and unambiguous.
-s	Suppresses the error message normally written to standard output.
-y*Type*	Specifies a type to compare with the SCCS identification keyword in the specified file.

Exit Status

The **val** command returns 0 if successful for all files; otherwise, it returns an 8-bit code that is a disjunction of the possible errors. It is interpreted as a bit string in which set bits (from left to right) are interpreted as follows:

0x80	Missing file argument.
0x40	Unknown or duplicate option.
0x20	Corrupted SCCS file.
0x10	Cannot open file or file not SCCS.
0x08	SID is invalid or ambiguous.
0x04	SID does not exist.
0x02	, y mismatch.

0x01 31, m mismatch.

Note: The **val** command can process two or more files on a given command line and can process multiple command lines (when reading standard input). In these cases, an aggregate code is returned; a logical OR of the codes generated for each command line and file processes.

Example

To determine if file `s.test.c` is an SCCS text file, enter:

```
val -ytext s.test.c
```

Related Information

List of SCCS Commands in *AIX 5L Version 5.3 General Programming Concepts: Writing and Debugging Programs*.

The **admin** command, **delta** command, **get** command, **prs** command.

The **sccsfile** file format.

Source Code Control System (SCCS) Overview in *AIX 5L Version 5.3 General Programming Concepts: Writing and Debugging Programs*.

---

varyoffvg Command

Purpose

Deactivates a volume group.

Syntax

varyoffvg [**-s**] *VolumeGroup*

Description

The **varyoffvg** command deactivates the volume group specified by the *VolumeGroup* parameter along with its associated logical volumes. The logical volumes first must be closed. For example, if the logical volume contains a file system, it must be unmounted.

To activate the volume group, use the **varyonvg** command.

Note: To use this command, you must either have root user authority or be a member of the **system** group.

You can use the System Management Interface Tool (SMIT) to run this command. To use SMIT, enter:

```
smit varyoffvg
```

Note: A volume group that has a paging space volume on it cannot be varied off while the paging space is active. Before deactivating a volume group with an active paging space volume, ensure that the paging space is not activated automatically at system initialization, and then reboot the system.

Flag

-s Puts the volume group into System Management mode, so that only logical volume commands can be used on the volume group. In this mode, no logical volume can be opened or accessed by users.

Exit Status

This command returns the following exit values:

0	Successful completion.
>0	An error occurred.

Security

RBAC Disclaimer

Examples

1. To deactivate volume group `vg03`, enter:

   ```
   varyoffvg vg03
   ```

2. To deactivate volume group `vg02`, but allow logical volume commands to continue to take effect, enter:

   ```
   varyoffvg  -s vg02
   ```

   Logical volumes within the volume group cannot be opened, but logical volume commands continue to take effect.

File

/usr/sbin/varyoffvg	Contains the **varyoffvg** command.

Related Information

The **exportvg** command, **mount** command, **umount** command, **varyonvg** command.

The **System Management Interface Tool (SMIT)** in in *AIX 5L Version 5.3 General Programming Concepts: Writing and Debugging Programs* explains the structure, main menus, and tasks that are done with SMIT.

The **Logical volumes** in *Operating system and device management* explains the Logical Volume Manager, physical volumes, logical volumes, volume groups, organization, ensuring data integrity, and allocation characteristics.

RBAC in *Security*.

Trusted AIX in *Security*.

---

varyonvg Command

Purpose

Activates a volume group.

Syntax

varyonvg [**-b**] [**-c**] [**-f**] [**-M** *ltgsize*] [**-n**] [**-p**] [**-r**] [**-s**] [**-t**] [**-u**] [**-k loc**|**rem**] [**-d**] [**-o**]*volumegroup*

Description

The **varyonvg** command activates the volume group specified by the *volumegroup* parameter and all associated logical volumes. A volume group that is activated is available for use. When a volume group is

activated, physical partitions are synchronized if they are not current. Physical volumes that are in the PVMISSING state and that have been replaced will be returned to the PVACTIVE state by the **varyonvg** command.

Note: If a physical volume is part of a dump device, the **varyonvg** command cannot return it to the PVACTIVE state. To run the command effectively, temporarily change the dump device.

A list of all physical volumes with their status is displayed to standard output whenever there is some discrepancy between the Device Configuration Database and the information stored in the Logical Volume Manager (LVM). The volume group may or may not be varied on. You must carefully examine the list and take proper action depending on each reported status to preserve your system integrity.

While varying on in concurrent mode, if the varyon process detects that there are logical volumes which are not previously known to the system, their definitions are imported. The permissions and ownership of the new device special files are duplicated to those of the volume group special file. If you have changed the permissions and/or ownership of the device special files of the logical volume on the node it was created, you will need to perform the same changes on this node.

Restriction: Classic Concurrent mode is not supported in AIX 5.3.

If the *volume group* cannot be varied on due to a loss of the majority of physical volumes, a list of all physical volumes with their status is displayed. To varyon the *volume group* in this situation, you will need to use the force option.

The **varyonvg** will fail to varyon the volume group if a majority of the physical volumes are not accessible (no Quorum). This condition is true even if the quorum checking is disabled. Disabling the quorum checking will only ensure that the volume group stays varied on even in the case of loss of quorum.

The *volume group* will not varyon if there are any physical volumes in `PV_MISSING` state and the quorum checking is disabled. This condition is true even if there are a quorum of disks available. To varyon on in this situation either use the force option or set an environment variable `MISSINGPV_VARYON` to TRUE (set this value in **/etc/environment** if the volume group needs to be varied with missing disks at the boot time).

In the above cases (using force varyon option and using `MISSINGPV_VARYON` variable), you take full responsibility for the *volume group* integrity.

Requirement: To use this command, you must either have root user authority or be a member of the **system** group.

You can use the System Management Interface Tool (SMIT) to run this command. To use SMIT, enter:

```
smit varyonvg
```

Flags

-b Breaks disk reservations on disks locked as a result of a normal **varyonvg** command. Use this flag on a volume group that is already varied on.
Note:

- This flag unlocks all disks in a given volume group.
- The **-b** flag opens the disks in the volume group using **SC_FORCED_OPEN** flag. For SCSI and FC disks this forces open all luns on the target address that this disk resides on. Volume Groups should therefore not share target addresses when using this **varyon** option.
- The **-b** flag can cause a system hang if used on a volume group that contains an active paging space.

-c Varies the volume group on Enhanced Concurrent mode. This is only possible if the volume group is Concurrent Capable or Enhanced Concurrent Capable and the system has the HACMP™ product loaded and available. If neither is true, the volume group fails the varyon. **Requirement:** Enhanced Concurrent volume groups use Group Services. Group Services must be configured prior to activating a volume group in this mode.

-d Allows data divergence. The **-d** flag only takes effect when you try to bring the volume group online while the cache at the opposite site might contain nonmirrored data updates and that cache is not accessible. If the **varyonvg** command detects that you might use back-level data and you do not specify the **-d** flag, the command fails with a meaningful error message.

For more information about asynchronous mirroring of geographic LVM, see *HACMP/XD for Geographic LVM: Planning and Administration Guide*.

-f Allows a volume group to be made active that does not currently have a quorum of available disks. All disk that cannot be brought to an active state will be put in a removed state. At least one disk must be available for use in the volume group. The **-f** flag (used to override quorum loss) is ignored if the volume group has not lost quorum. If a disk is put into removed state, use the **chpv -v a** *PVname* command to bring the disk back to active state.

-k loc | **rem** Keeps data from the local mirror copy or remote mirror copy. You can specify the following attributes with the **-k** flag:

loc Retains the local mirror copy data. Local means local physical volumes and not primary site

rem Retains the remote mirror copy data. Remote means remote physical volumes and not remote site.

For more information about asynchronous mirroring of geographic LVM, see *HACMP/XD for Geographic LVM: Planning and Administration Guide*.

-M *ltgsize* Statically sets the *ltgsize* of the volume group. Valid values for *ltgsize* include 4K, 8K, 16K, 32K, 64K, 128K, 1M, 2M, 4M, 8M, 16M, 32M, and 128M. If any disk in the volume group is not configured with a max transfer of *ltgsize* or greater, the varyonvg will fail.

-n Disables the synchronization of the stale physical partitions within the *volumegroup.*

-o Allows using data from partitions that are stale in the copy you selected but fresh in the other copy. The **varyonvg** command fails if you specify the **-k** flag to preserve either local copy or remote copy in the data divergence case and the **varyonvg** command cannot preserve the complete copy because some partitions are not fresh in the local or remote copy that you selected. You can override the failure by specifying the **-o** flag to use data from partitions that are stale in the copy that you selected but fresh in the other copy. The **-o** flag is only valid with the **-k** flag.

For more information about asynchronous mirroring of geographic LVM, see *HACMP/XD for Geographic LVM: Planning and Administration Guide*.

-p All physical volumes must be available to use the **varyonvg** command.

-r Varies on the volume group in read-only mode. This mode prevents:

- Write operations to logical volumes
- LVM meta-data updates
- Stale partitions synchronization

Restriction: Mounting a JFS filesystem on a read-only logical volume is not supported.
Restriction: All LVM high-level commands that require the LVM meta-data update will fail the request in this mode.

-s Makes the volume group available in System Management mode only. Logical volume commands can operate on the volume group, but no logical volumes can be opened for input or output.
Restriction: Logical volume commands also cannot read or write to or from logical volumes in a volume group varied on with the **-s** flag. Logical volumes that attempt to write to a logical volume in a volume group varied on with the **-s** flag (such as **chvg** or **mklvcopy**) may display error messages indicating that they were unable to write to and/or read from the logical volume.

-t Checks the timestamps in the Device Configuration Database and the Logical Volume Manager. If there is a discrepancy in the timestamps, the **synclvodm** command is issued to synchronize the database.
Tip: This check is always done if the Volume Group is varied on in concurrent mode.

-u Varies on a volume group, but leaves the disks that make up the volume group in an unlocked state. Use this flag as part of the initial varyon of a dormant volume group. This flag only applies to AIX 4.2 or later.

Attention: The base design of LVM assumes that only one initiator can access a volume group. The HACMP product does work with LVM in order to synchronize multi-node accesses of a shared volume group. However, multi-initiator nodes can easily access a volume group with the **-b** and **-u** flags without the use of *HACMP*. Your must be aware that volume group status information may be compromised or inexplicably altered as a result of disk protect (locking) being bypassed with these two flags. If you use the **-b** and **-u** flags, data and status output cannot be guaranteed to be consistent.

Exit Status

This command returns the following exit values:

0 Successful completion.
>0 An error occurred.

Security

RBAC Disclaimer

Examples

1. To activate volume group `vg03`, enter:
   ```
   varyonvg vg03
   ```
2. To activate volume group `vg03` without synchronizing partitions that are not current, enter:
   ```
   varyonvg -n vg03
   ```

Files

/usr/sbin Contains the **varyonvg** command directory.
/tmp Stores the temporary files while the command is running.

Related Information

The **chvg** command, **lspv** command, **lslv** command, **lsvg** command, **varyoffvg** command.

The System Management Interface Tool (SMIT) in *AIX 5L Version 5.3 General Programming Concepts: Writing and Debugging Programs* explains the structure, main menus, and tasks that are done with SMIT.

The Logical volumes in *Operating system and device management* explains the Logical Volume Manager, physical volumes, logical volumes, volume groups, organization, ensuring data integrity, and allocation characteristics.

RBAC and Trusted AIX in *Security*.

HACMP/XD for Geographic LVM: Planning and Administration Guide provides information about asynchronous mirroring of geographic LVM.

---

vc Command

Purpose

Substitutes assigned values for identification keywords.

Syntax

vc [**-a**] [**-t**] [**-s**] [**-c***Character*] [*Keyword***=***Value*]...

Description

The **vc** command copies lines from standard input to standard output. The flags and keywords on the command line and control statements in the input modify the resulting output. The **vc** command replaces user-declared keywords with the value assigned on the command line. Keywords can be replaced both in text and in control statements.

Control Statements

A control statement is a single line beginning with a control character (the default control character is a **:** (colon)). Control statements provide conditional processing of the input. The allowable types of control statements are:

:if *Condition*

Text

:end	Writes all the lines between the **:if** statement and the matching **:end** to standard output only if the condition is true. You can nest **:if** and **:end** statements. However, once a condition is false, all remaining nested **:if** and **:end** statements are ignored. See the ″Condition Syntax″ section for the syntax of conditions and allowable operators.
:dcl *Keyword***,** [*Keyword* . . .]	Declares specified keywords. All keywords must be declared.
:asg *Keyword***=***Value*	Assigns the specified value to the specified keyword. An **:asg** statement takes precedence over keyword assignment on the **vc** command line. A later **:asg** statement overrides all earlier assignments of the associated keyword. The keywords that are declared but not assigned *Values,* have null values.
:: *Text*	Removes the two leading control characters, replaces keyword*s* with their respective values, and then copies the line to standard output.
:on or **:off**	Turns on or off keyword replacement on all lines.
:ctl *Character*	Changes the control character to the *Character* value.
:msg *Message*	Writes a message to standard error output in the form: Message(n): message `where n is number of the input line on which the message appeared.`

:err *Message*	Writes an error message to standard error. The **vc** command stops processing and returns an exit value of 1. The error message is in the form:
	`ERROR: message`
	`ERROR: err statement on line n (vc15)`

Condition Syntax

The items and statements allowed are:

```
condition           ::=OR statement
                    ::=NOR statement
OR statement        ::=AND statement
                    ::=AND statement | OR statement
AND statement       ::=expression
                    ::=expression & AND statement
expression          ::=( OR statement )
                    ::=value operator value
operator value      ::= = or != or < or >
                    ::= ASCII string
                    ::= numeric string
```

The available condition operators and their meanings are:

=	Equal
!=	Not equal
&	AND
&|	OR
>	Greater than
<	Less than
( )	Used for logical groupings
NOT	May only occur immediately after the *if*, and when present, inverts the value of the entire condition.

The > and < (greater-than and less-than) operate only on unsigned integer values; for example, 012 > 12 is false. All other operators take strings as modifiers; for example, 012 ! = 12 is true. The precedence of the operators, from highest to lowest precedence, is as follows:

- = ! = > < (all of equal precedence)
- &
- &|

Parentheses can be used to alter the order of precedence.

Values must be separated from operators or parentheses by at least one blank or tab.

Keyword Replacement

A keyword must begin and end with the same control character used in control statements. A keyword may be up to nine alphanumeric characters, where the first character must be alphabetic. Keyword values can be any ASCII string. A numeric keyword *Value* is an unsigned string of digits. Values cannot contain tabs or spaces.

Flags

-a	Replaces keyword*s* surrounded by control characters with their assigned value in all text lines (not just those beginning with two control characters).
-c*Character*	Uses the *Character* value as the control character. The *Character* parameter must specify an ASCII character.
-s	Does not display the warning messages normally displayed to standard error.
-t	Ignores all characters from the beginning of a line up to and including the first tab character for detecting a control statement. If the **vc** command finds a control character, it ignores all characters up to and including the tab.

Exit Status

This command returns the following exit values:

0 Successful completion.
>0 An error occurred.

Examples

1. Examples of *Keyword*=*Value* assignments are:

   ```
   numlines=4
   prog=acctg
   pass4=yes
   ```

   The **vc** command removes all control characters and keywords from input text lines marked with two control characters as it writes the text to standard output.
2. To prevent a control character from being interpreted, precede it with a backslash, as in the following example:

   ```
   ::the :prog: program includes several of the following\:
   ```

   The **:prog:** keyword is replaced by its value, but the **\:** is passed to standard output as **:** (colon).

   Input lines beginning with a \ (backslash) followed by a control character are not control lines, and are copied to standard output without the backslash. However, the **vc** command writes lines beginning with a backslash and no following control character without any changes (including the initial backslash).

File

/usr/bin/vc Contains the **vc** command.

Related Information

The **admin** command, **delta** command, **get** command.

List of SCCS Commands in *AIX 5L Version 5.3 General Programming Concepts: Writing and Debugging Programs*.

Source Code Control System (SCCS) Overview in *AIX 5L Version 5.3 General Programming Concepts: Writing and Debugging Programs*.

---

vgrind Command

Purpose

Formats listings of programs that are easy to read.

Syntax

vgrind [**-f**] [**-n**] [**-t**] [**-x**] [**-P***Printdev*] [**-T***Name*] [**-**] [**-d***File*] [**-h** *Header*] [**-l***Language*] [**-s***Size*] [*File* ...]

Description

The **vgrind** command formats (grinds) the program sources specified by the *File* parameters in an easily readable style using the **troff** command. Comments are placed in italics, keywords in boldface, and the name of the current function is listed down the margin of each page as it is encountered.

The **vgrind** command runs in either filter mode or regular mode.

In filter mode, the **vgrind** command acts as a filter in a manner similar to the **tbl** command. Standard input is passed directly to standard output except for lines bracketed by the following **troff**-like macros:

.vS Starts processing.
.vE Ends processing.

The preceding lines are formatted according to the **vgrind** command conventions. The output from this filter can be passed to the **troff** command for output. There is no particular ordering with the **eqn** or **tbl** command.

In regular mode, the **vgrind** command accepts input files, processes them, and passes them in order to the **troff** command, the appropriate postprocessor, and then the printer.

In both modes, the **vgrind** command passes without converting lines, beginning with a decimal point.

The **vgrind** command supports only ASCII keywords defined in either the standard **/usr/share/lib/vgrindefs** language definitions file or any alternately specified file by the **-d** flag.

Flags

-f	Forces filter mode.
-n	Forces no keyword bolding.
-t	Causes formatted text to go to standard output.
-x	Outputs the index file in an easily readable format. The index file itself is produced whenever the **vgrind** command is run with the **index** file in the current directory. The index of function definitions can then be run off by running the **vgrind** command with the **-x** flag and the *File* parameter.
-P*PrintDev*	Sends the output to *Printdev* Printer using the **qprt** command. If this flag is not specified, the **PRINTER** environment variable is used. If the **PRINTER** environment variable is not set, the system default is used.
-T*Name*	Creates output for a **troff** device as specified by the *Name* parameter. The output is sent through the appropriate postprocessor. The default is the **ibm3816** postprocessor.
-	Forces input to be taken from standard input (default if the **-f** flag is specified).
-d*File*	Specifies an alternate language definitions file (default is the **/usr/share/lib/vgrindefs** file).
-h *Header*	Specifies a particular header to put on every output page (default is the file name).

Note: A blank space is required after the **-h** flag before the *Header* variable.

-l*Language* Specifies the language to use. Currently known languages are:

c	C (the default). Function names can be preceded on a line only by spaces, tabs, or an asterisk. The parenthetical options must also be on the same line.
csh	CSH.
p	PASCAL. Function names must be displayed on the same line as the **function** or **procedure** keywords.
m	MODEL. Function names must be displayed on the same line as the **isbeginproc** keyword phrase.
sh	SHELL.
r	RATFOR.
mod2	MODULA2.
yacc	YACC.
isp	ISP.
l	ICON.
-s *Size*	Specifies a point size to use on output (exactly the same as a **.ps** request).

Files

index	Contains the file the where source for the index is created.
/usr/bin/vgrind	Contains the **vgrind** command.
/usr/share/lib/tmac/tmac.vgrind	Contains the macro package.
/usr/share/lib/vfontedpr	Contains the preprocessor.
/usr/share/lib/vgrindefs	Contains the language descriptions.

Related Information

The **qprt** command, **tbl** command, **troff** command.

The **vgrindefs** File Format.

---

vi or vedit Command

Purpose

Edits files with a full-screen display.

Syntax

{ **vi** | **vedit** } [**-l**] [**-R**] [**-t***Tag*] [**-v**] [**-w***Number*] [**-y***Number*] [**-r** [*File*]] [{ **+** | **-c** } { *Subcommand* }] [*File* ...]

Description

The **vi** command starts a full-screen editor based on the underlying ex editor. Therefore, ex subcommands can be used within the vi editor. The **vedit** command starts a version of the vi editor intended for beginners. In the vedit editor, the **report** option is set to 1, the **showmode** option is set, and the **novice** option is set, making it a line editor.

You start the vi editor by specifying the name of the file or files to be edited. If you supply more than one *File* parameter on the command line, the vi editor edits each file in the specified order. The vi editor on an

existing file displays the name of the file, the number of lines, and the number of characters at the bottom of the screen. In case of multibyte locales the number of characters need to be interpreted as the number of bytes.

Since the vi editor is a full-screen editor, you can edit text on a screen-by-screen basis. The vi editor makes a copy of the file you are editing in an edit buffer, and the contents of the file are not changed until you save the changes. The position of the cursor on the display screen indicates its position within the file, and the subcommands affect the file at the cursor position.

vi Editor Limitations

The following list provides the maximum limits of the vi editor. These counts assume single-byte characters.

- 256 characters per global command list
- 2048 characters in a shell escape command
- 128 characters in a string-valued option
- 30 characters in a tag name
- 128 map macros with 2048 characters total
- 1,048,560 lines silently enforced
- The macro name and the macro text are limited to 100 characters.

Note: The vi editor supports a maximum of 2 GB edit buffer.

vi Editing Modes

The vi editor operates in the following modes:

command mode
When you start the vi editor, it is in command mode. You can enter any subcommand except those designated for use only in the text input mode. The vi editor returns to command mode when subcommands and other modes end. Press the Esc key to cancel a subcommand.

text-input mode
You use the vi editor in this mode to add text. Enter text input mode with any of the following subcommands: the **a** subcommand, **A** subcommand, **i** subcommand, **I** subcommand, **o** subcommand, **O** subcommand, **c***x* subcommands (where the *x* represents the scope of the subcommand), **C** subcommand, **s** subcommand, **S** subcommand, and **R** subcommand. After entering one of these subcommands, you can enter text into the editing buffer. To return to command mode, press the Esc key for normal exit or press Interrupt (the Ctrl-C key sequence) to end abnormally.

last-line mode
Subcommands with the prefix **:** (colon), **/** (slash), **?** (question mark), **!** (exclamation point), or **!!** (two exclamation points) read input on a line displayed at the bottom of the screen. When you enter the initial character, the vi editor places the cursor at the bottom of the screen, where you enter the remaining characters of the command. Press the Enter key to run the subcommand, or press Interrupt (the Ctrl-C key sequence) to cancel it. When the **!!** prefix is used, the cursor moves only after both exclamation points are entered. When you use the **:** prefix to enter the last-line mode, the vi editor gives special meaning to the following characters when they are used before commands that specify counts:

% All lines regardless of cursor position

$ Last line

. Current line

Note: The history of last line mode subcommands can be navigated using the Up and Down Arrow keys.

Customizing the vi Editor

You can customize the vi editor by:

- Setting vi editor options
- Defining macros
- Mapping keys
- Setting abbreviations

Setting vi Editor Options: The following list describes the vi editor options you can change with the **set** command. The default setting for these options is **off**. If you turn on one of these toggle options, you can turn it off again by entering the word **no** before the option. If you want to discontinue the **autowrite** vi option, enter **noaw**, where **no** turns off the option and **aw** specifies the **autowrite** option.

Note: Do not include parentheses when entering vi options.

vi Option (Abbreviation)	Description
autoindent (ai)	Indents automatically in text input mode to the indentation of the previous line by using the spacing between tab stops specified by the **shiftwidth** option. The default is **noai**. To back the cursor up to the previous tab stop, press the Ctrl-D key sequence. This option is not in effect for global commands.
autoprin (ap)	Prints the current line after any command that changes the editing buffer. The default is **ap**. This option applies only to the last command in a sequence of commands on a single line and is not in effect for global commands.
autowrite (aw)	Writes the editing buffer to the file automatically before the **:n** subcommand, the **:ta** subcommand, the Ctrl-A, Ctrl -], and Ctrl -T key sequences, and the **!** subcommand if the editing buffer changed since the last **write** subcommand. The default is **noaw**.
backtags (bt)	Allows the Ctrl-T subcommand to return the file editing position to the location where the previous Ctrl-] subcommand was issued. If **nobacktags** is set, then Ctrl-T is the same as Ctrl-]. The default is **backtags**.
beautifying text (bf)	Prevents the user from entering control characters in the editing buffer during text entry (except for tab, new-line, and form-feed indicators). The default is **nobf**. This option applies to command input.
closepunct (cp=)	Handles a list of closing punctuation, especially when wrapping text (**wraptype** option). Precedes multicharacter punctuation with the number of characters; for example, `cp=3.;)}`. The **vi** command does not split closing punctuation when wrapping.
directory (dir=)	Displays the directory that contains the editing buffer. The default is **dir = /var/tmp**.
edcompatible (ed)	Retains **g** (global) and **c** (confirm) subcommand suffixes during multiple substitutions and causes the **r** (read) suffix to work like the **r** subcommand. The default is **noed.**
exrc (exrc)	If not set, ignores any **.exrc** file in the current directory during initialization, unless the current directory is that named by the **HOME** environment variable. The default is **noexrc**.
hardtabs (ht=)	Tells the vi editor the distance between the hardware tab stops on your display screen. (This option must match the tab setting of the underlying terminal or terminal emulator.) The default is **ht=8**.
history (hist=)	Sets the limit on last line mode history commands. The initial value is **hist=32**. The history size is zero (**hist=0**) for the **tvi** command.
ignorecase (ic)	Ignores distinction between uppercase and lowercase while searching for regular expressions. The default is **noic**.

vi Option (Abbreviation)	Description
linelimit (ll=)	Sets the maximum number of lines, as per the **-y** command-line option. This option only is effective if used with the **.exrc** file or the **EXINIT** environment variable.
lisp (lisp)	Removes the special meaning of ( ), { }, [[, and]] and enables the = (formatted print) operator for s-expressions, so you can edit list processing (LISP) programs. The default is **nolisp.**
list (list)	Displays text with tabs (^I) and the marked end of lines ($). The default is **nolist.**
magic (magic)	Treats the . (period), [(left bracket), and * (asterisk) characters as special characters when searching for a pattern. In off mode, only the ( ) (parentheses) and $ (dollar sign) retain special meanings. However, you can evoke special meaning in other characters by preceding them with a \ (backslash). The default is **magic**.
mesg (mesg)	Turns on write permission to the terminal if set while in visual mode. This option only is effective if used with the **.exrc** file or the **EXINIT** environment variable. The default is **on**.
modeline (modeline)	Runs a vi editor command line if found in the first five or the last five lines of the file. A vi editor command line can be anywhere in a line. For the vi editor to recognize a command line, the line must contain a space or a tab followed by the `ex:` or `vi:` string. The command line is ended by a second **:** (colon). The vi editor tries to interpret any data between the first and second colon as vi editor commands. The default is **nomodeline**.
novice	Indicates whether you are in **novice** mode. You cannot change the value by using the **set** command.
number (nu)	Displays lines prefixed with their line numbers. The default is **nonu**.
optimize (opt)	Speeds the operation of terminals that lack cursor addressing. The default is **noopt**.
paragraphs (para=)	Defines vi macro names that start paragraphs. The default is **para=lPLPPPQPP\ LIpplpipnpbp**. Single-letter **nroff** macros, such as the **.P** macro, must include the space as a quoted character if respecifying a paragraph.
partialchar (pc=)	Appears in the last display column where a double-wide character would not be displayed completely. The default character is **-** (minus sign).
prompt	Prompts for a new vi editor command when in command mode by printing a **:** (colon). The default is **on**.
readonly (ro)	Sets permanent read-only mode. The default is **noreadonly**.
redraw (redraw)	Simulates a smart workstation on a dumb workstation. The default is **nore**.
remap	Allows defining macros in terms of other macros. The default is **on**.
report (re=)	Sets the number of times you can repeat a command before a message is displayed. For subcommands that produce many messages, such as global subcommands, the messages are displayed when the command sequence completes. The default is **report=5.**
scroll (scr=)	Sets the number of lines to be scrolled when the user scrolls up or down. The default is 1/2 of the window size, rounded down.
sections (sect=)	Defines vi macro names that start sections. The default is **sect=NHSHHH\ HUuhsh+c**. Single-letter **nroff** macros, such as the **.P** macro, must include the space as a quoted character if respecifying a paragraph.
shell (sh=)	Defines the shell for the **!** subcommand or the **:!** subcommand. The default is the login shell.

vi Option (Abbreviation)	Description
shiftwidth (sw=)	Sets the distance for the software tab stops used by the **autoindent** option, the shift commands **(>** and **<)**, and the text input commands (the Ctrl-D and Ctrl-T key sequences). This vi option only affects the indentation at the beginning of a line. The default is **sw=8.**
showmatch (sm)	Shows the ((matching left parenthesis) or { (left bracket) as you type the) (right parenthesis) or } (right bracket). The default is **nosm**.
showmode (smd)	Displays a message to indicate when the vi editor is in input mode. The default is **nosmd**.
slowopen (slow)	Postpones updating the display screen during inserts. The default is **noslow.**
tabstop (ts=)	Sets the distance between tab stops in a displayed file. The default is **ts=8**.
tags (tags =)	Defines the search path for the database file of function names created using the **ctags** command. The default is **tags=tags\ /usr/lib/tags**.
term (term=)	Sets the type of workstation you are using. The default is **term=$TERM**, where **$TERM** is the value of the **TERM** shell variable.
terse (terse)	Allows the vi editor to display the short form of messages. The default is **noterse**.
timeout (to)	Sets a time limit of two seconds on an entry of characters. This limit allows the characters in a macro to be entered and processed as separate characters when the **timeout** option is set. To resume use of the macro, set the **notimeout** option. The default is **to**.
ttytype	Indicates the tty type for the terminal being used. You cannot change this value from the vi editor.
warn (warn)	Displays a warning message before the **!** subcommand executes a shell command if it is the first time you issued a shell command after changes were made in the editing buffer but not written to a file. The default is **warn**.
window (wi=)	Sets the number of lines displayed in one window of text. The default depends on the baud rate at which you are operating: 600 baud or less, 8 lines; 1200 baud, 16 lines; higher speeds, full screen minus 1 line.
wrapmargin (wm=)	Sets the margin for automatic word wrapping from one line to the next. The default is **wm=0**. A value of 0 turns off word wrapping.
wrapscan (ws)	Allows string searches to wrap from the end of the editing buffer to the beginning. The default is **ws**.
wraptype (wt=)	Indicates the method used to wrap words at the end of a line. The default value is **general**. You can specify one of the following four values: **general** Allows wraps on word breaks as white space between two characters. This setting is the default. **word** Allows wraps on words. **rigid** Allows wraps on columns and before closing punctuation. **flexible** Allows wraps on columns, but one character of punctuation can extend past the margin.
writeany (wa)	Turns off the checks usually made before a **write** subcommand. The default is **nowa**.

To see a list of the vi editor settings that have changed from the default settings, enter `set` and press the spacebar. Press the Enter key to return to the command mode.

To see a complete list of the vi editor settings, enter `set all`. Press the Enter key to return to the command mode.

To turn on a vi editor option, enter `set Option`. This command automatically returns you to the command mode.

To turn on multiple vi editor options, enter `set Option Option Option`. This command turns on the three designated vi editor options and returns you to the command mode.

To turn off a vi editor option, enter `set noOption`. This command automatically returns you to the command mode.

To change the value of a vi editor option, enter `set Option=Value`. This command automatically returns you to the command mode.

You can use the **:set** subcommand of the vi editor to set options for this editing session only, or to set options for this editing session and all future editing sessions.

To set or change vi editor options *for this editing session only*, enter the **:set** subcommand from the command line.

To set vi options for *all editing sessions*, put the **:set** subcommand in the **EXINIT** environment variable in the **.profile** file (read by the shell on login) or put the **set** subcommand into a **.exrc** file. The vi editor first looks for the **EXINIT** environment variable and runs its commands. If the **EXINIT** environment variable does not exist, the vi editor then looks for the **$HOME/.exrc** file and runs its commands. Last, and regardless of any previous results, the vi editor looks for the local **.exrc** file and runs its commands.

Note: This process is true except with the **tvi** command (trusted vi). In this instance, the vi editor looks for and runs only the **/etc/.exrc** file.

For information about changing an option by setting the **EXINIT** environment variable, see the description of environment variables in the **environment** file.

The **.exrc** file can contain subcommands of the form **set** *Option*=*Value*; for example:

```
set cp=3 . . ;
```

To include a comment in the **.exrc** file, use a " (double quotation mark) as the first character in the line.

Defining Macros: If you use a subcommand or sequence of subcommands frequently, you can use the vi editor to define a macro that issues that subcommand or sequence.

To define a macro, enter the sequence of subcommands into a buffer named with a letter of the alphabet. The lowercase letters a through z overlay the contents of the buffer, and the uppercase letters A through Z append text to the previous contents of the buffer, allowing you to build a macro piece by piece.

For example, to define a buffer macro named c that searches for the word corner and makes the third line after the word corner the current line, enter the following command:

```
o /corner/+3
```

Then press the Esc key and enter the following command:

```
"c
```

where c is the name of the buffer macro.

To add text to the previous contents of the defined buffer, enter the `o viSubcommand`, press the Esc key, and enter `"CapitalLetter`, where the *CapitalLetter* variable specifies an uppercase letter A through Z. For example, to build a buffer macro named T that searches for the word corner and allows you to add more commands, enter the following command:

```
o corner
```

Then press the Esc key and enter the following command:

```
"T
```

where T is the name of the buffer macro. You can repeat this process at any time to add more vi subcommands to the same buffer.

For example, to add commands that move the cursor to the previous line and delete that line, enter the following command:

```
o -dd
```

where - (minus sign) means to move the cursor up one line, and dd means to delete the current line. Press the Esc key and enter the following command:

```
"Tdd
```

To start the macro, enter `@Letter`, where the *Letter* variable specifies the letter name of the buffer macro you want to use. To use the same macro again, enter `@@` (two at symbols). For example, enter `@T` to start the T buffer macro and run the **search**, **move cursor**, and **delete line** commands. Enter `@@T` to start the T buffer macro again.

The character set used by your system is defined by the collation table. This table affects the performance of vi macros.

Mapping Keys: You can use the **:map**, **:map!**, and **:ab** subcommands to map a keystroke to a command or a sequence of commands. The **:map** subcommand is used in the command mode. The **:map!** and **:ab** subcommands are used in the text input mode. You can map keys for this editing session and all future editing sessions or only for the current editing session from either mode.

To map keys *for all future editing sessions*, put the subcommand into a **$HOME/.exrc** file. Each time you start the vi editor, it reads this file. The mapping remains in effect for every editing session.

To map keys *for the current editing session only* from the *command mode*, start the subcommand during the vi editor session. To map keys for the current editing session only from the *text input mode*, enter the subcommand on the command line during the vi editor session. The mapping remains in effect only for the current editing session.

Attention: If you use an IBM 3161 ASCII display station, IBM 3163 ASCII display station, or IBM 3101 ASCII display station, the default key-mapping of the vi editor can cause you to lose data. To see the default mapping, issue a **:map** subcommand. Specific problems arise with the Esc-J or Shift-J key sequence. These key sequences delete all information from the current position of the cursor to the end of the file. To avoid problems, change this key sequence using a **.exrc** file.

The **:map**, **:map!**, and **:ab** subcommands are defined and used as follows:

:map Defines macros in the command mode. The **:map** subcommand allows you to run a specified command or sequence of commands by pressing a single key while in the vi editor.

To map keys in the command mode, start the vi editor with an empty editing buffer and do not name a vi file using the **vi** command or type anything into the buffer after the vi editor starts. You can use the **:map** subcommand to do the following:

- To map a character to a sequence of editing commands, enter:

  ```
  :map Letter viSubcommand
  ```

- To unmap a character previously mapped in command mode, enter:

  ```
  :unmap Letter
  ```

- To display a list of current mappings for the command mode, enter

  ```
  :map
  ```

The following keys are not used by the vi editor, but are available for use with the **:map** subcommand in the command mode:

- Letters g, K, q, V, and v
- Control key sequences Ctrl-A, Ctrl-K, Ctrl-O, Ctrl-W, and Ctrl-X
- Symbols _ (underscore), * (asterisk), \ (backslash), and = (equal sign)

Although you can map a key that is already used by the vi editor, the key's usual function is not available as long as the map is in effect. Some terminals allow you to map command sequences to function keys. If you are in LISP mode, the = (equal sign) cannot be used because it is used by the vi editor.

To map the letter v to the sequence of commands that would locate the next occurrence of the word `map` and change it to the word `MAP`, enter the following command:

```
:map v /map<Ctrl-V><Enter>cwMAP<Ctrl-V><Esc><Ctrl-V><Enter>
```

The previous example instructs the vi editor to locate the next occurrence of `map` (`/map<Ctrl-V><Enter>`), change `map` to `MAP` (`cwMAP`), end the change-word subcommand (`<Ctrl-V><Esc>`), and enter the command (`<Ctrl-V><Enter>`).

Requirement: To prevent the vi editor from interpreting the Enter key, it must be preceded by the Ctrl-V key sequence when being mapped. This condition is also true of the Esc, Backspace, and Delete keys.

To map the control characters Ctrl-A, Ctrl-K, and Ctrl-O, simultaneously press the Ctrl key and the letter. For example, to map the Ctrl-A key sequence to the sequence of commands that saves a file and edits the next one in a series, enter the following command:

```
:map <Ctrl-A> :w<Ctrl-V><Enter>:n<Ctrl-V><Enter>
```

To map the control characters Ctrl-T, Ctrl-W, and Ctrl-X, you must first escape them with the Ctrl-V key sequence.

To map the | (pipe symbol), you must first escape it with the two Ctrl-V key sequences, as illustrated by the following example that maps the character g to the sequence of commands that escapes to the shell, concatenates the file **/etc/motd**, and pipes the output to the **wc** command:

```
:map g :!cat /etc/motd <Ctrl-V><Ctrl-V>| wc<Ctrl-V><Enter>
```

If your terminal permits you to map function keys, you must reference them with the #*number* key sequence to designate the number of the function key that you want to map. In the following example, the F1 function key is mapped to the sequence of commands that deletes a word and moves the cursor three words down:

```
:map #1 dwwww
```

In order for function key mapping to work, the output of the function key for your terminal type must match the output defined in the **terminfo** file. These definitions are denoted by the kf*number* entries, where kf1 represents the F1 function key, kf2 represents the F2 function key, and so on. If the output that you get when you press the function key does not match this entry, you must use the terminal's setup mode to correct the settings to match these terminal database entries before any mapping can occur.

You can also map certain keyboard special keys, such as the Home, End, Page Up, and Page Down keys. For most terminals, these keys are already mapped in the vi editor. You can verify this mapping by using the **:map** subcommand. If these keys are not already mapped, you can use the **:map** subcommand as follows:

```
:map <Ctrl-V><End> G
:map <Ctrl-V><Home> 1G
:map <Ctrl-V><PageUp> <Ctrl-F>
:map <Ctrl-V><PageDown> <Ctrl-B>
```

To get a listing of all current maps in the command mode, enter the **:map** subcommand. The preceding examples are then displayed as follows:

```
v       v          /map<Ctrl-M>cwMAP<Ctrl-[>Ctrl-M>
<Ctrl-A>  <Ctrl-A>   :w<Ctrl-M>:n<Ctrl-M>
g         g          :!cat /etc/motd | wc <Ctrl-M>
```

Tip: The Ctrl-V and Enter key sequence is displayed as the Ctrl-M key sequence, and the Ctrl-V and Esc key sequence is displayed as the Ctrl-[key sequence.

:map! Maps character strings to single keys while in text input mode. To map keys in the text input mode, start the vi editor with an empty editing buffer and do not name a vi file using the **vi** command or type anything into the buffer after the vi editor starts. You can use the **:map!** subcommand to do the following:

- To map a letter to one or more vi strings in text input mode, enter:

  ```
  :map! Letter String
  ```

- To unmap a letter previously mapped in text input mode, enter:

  ```
  :unmap! Letter
  ```

- To display a list of existing strings that are mapped to specific keys in text input mode, enter:

  ```
  :map!
  ```

Typing the mapped key while in text input mode produces the specified string. The Ctrl-V and Esc key sequence puts you into command mode, backs up to the beginning of the current word (**bbw**), and starts the **cw** (change-word) subcommand. For example:

```
:map! % <Ctrl-V><Esc>bbwcw
```

When typing text, if you realize that you have mistyped a word, you can change it by pressing the % (percent) key and retyping the word. You are automatically returned to insert mode.

Important: Be careful when choosing keys to be used for the **:map!** subcommand. Once keys have been mapped, they can no longer be input as text without first issuing the **:unmap!** subcommand.

:ab Maps a key or sequence of keys to a string of characters for use in the text input mode. The **:ab** subcommand is useful when inputting text that possesses several repetitive phrases, names, or titles.

The following example replaces the word `city` with the phrase `Austin, Texas 78759` whenever it is typed in text input mode and followed by a white space, period, or comma:

```
:ab city Austin, Texas 78759
```

For example, if while inputting text, you type the following:

```
My current residence is city.
```

Pressing the Tab key expands the word `city` to read:

```
My current residence is Austin, Texas 78759.
```

The abbreviation is not expanded within a word. For example, if you type `My current residence iscity`, the word iscity is not expanded.

If the **:map!** subcommand is used to map abbreviations for insert mode, then all occurrences of the abbreviations are expanded regardless of where it occurs. If you used the **:map!** subcommand for the preceding example (`:map! city Austin, Texas 78759`), then whenever you type the word `city`, regardless of what precedes or follows, the word will be expanded to `Austin, Texas 78759`. Therefore, the word `iscity` becomes `isAustin, Texas 78759`.

Important: Be careful when choosing the keys that are used for the **:ab** subcommand. Once keys are defined, they can no longer be input as text without first issuing the **:unab** subcommand.

Setting Abbreviations: The **set** command has behavior similar to the **map!** command except that the **set** command substitutes the string for the abbreviation only when the abbreviation is a separate word. You can use the **set** command of the vi editor to:

- List existing abbreviations
- Remove an abbreviation
- Set (define) an abbreviation

Tip: Start the vi editor with an empty editing buffer. Do not name a vi file using the **vi** command or type anything into the buffer after the vi editor starts. Press the Esc key to be sure you are in the command mode.

To list abbreviations Enter the **:ab** command to list existing abbreviations. Press the Enter key to return to command mode.

To remove abbreviations Enter the **:anab***Abbreviation* command to remove an abbreviation, where the *Abbreviation* variable specifies the character string you do not want abbreviated any more.

To set (define) an abbreviation

Enter the **:ab** *Abbreviation String* command to set an abbreviation, where the *Abbreviation* variable specifies the character string being defined as an abbreviation and the *String* variable specifies the character string being abbreviated. The abbreviation can be substituted for the string only when the abbreviation is a separate word.

For example, if you enter the **:ab kn upper** command and then type `acknowledge` while in the text input mode, the set abbreviation string is not started because the kn string in the word acknowledge is not a separate word.

However, if you type the **:ab kn upper** command and then type `make the kn line all kncase` while in the text input mode, the result is `make the upper line all uppercase`.

Flags

-c*Subcommand*	Carries out the ex editor subcommand before viewing with **vi** begins. The cursor moves to the line affected by the last subcommand to be carried out. When a null operand is entered, as in **-c' '**, the vi editor places the cursor on the first line of the file. The **-c** flag is incompatible with the **+** flag. Do not specify both flags at the same time.
-l	Enters the vi editor in LISP mode. In this mode, the vi editor creates indents appropriate for LISP code, and the **(**, **)**, **{**, **}**, **[[**, and **]]** subcommands are modified to act appropriately for LISP.
-r[*File*]	Recovers a file after a vi editor or system malfunction. If you do not specify the *File* variable, the vi editor displays a list of all saved files.
-R	Sets the **readonly** option to protect the file against overwriting.
-t*Tag*	Edits the file containing the *Tag* variable and positions the vi editor at its definition. To use this flag, you must first create a database of function names and their locations using the **ctags** command.
-v	Enters the vi editor in the verbose mode.
-w*Number*	Sets the default window size to the value specified by the *Number* variable. This flag is useful when you use the vi editor over a low-speed line.
-y*Number*	Overrides the maximum line setting of 1,048,560 with any value greater than 1024. You should request twice the number of lines that you require because the vi editor uses the extra lines for buffer manipulation.
+[*Subcommand*]	Carries out the ex editor subcommand before editing begins. If you do not specify the *Subcommand* variable, the cursor is placed on the first line of the file. This **+** flag is incompatible with the **-c** flag. Do not specify both flags at the same time.

vi General Subcommand Syntax

Use the following general syntax to enter subcommands:

[*Named_Buffer*] [*Operator*] [*Number*] *Object*

Tip: Square brackets indicate optional items.

[*Named_Buffer*]	Specifies a temporary text storage area.
[*Operator*]	Specifies the subcommand or action; instructs the vi editor.
[*Number*]	Specifies either the extent of the action or a line address as a whole number.
Object	Specifies what to act on, such as a text object (a character, word, sentence, paragraph, section, character string) or a text position (a line, position in the current line, screen position).

Counts before Subcommands

You can put a number in front of many subcommands. The vi editor interprets this number in one of the following ways:

- Go to the line specified by the *Number* parameter:

```
5G
10Z
```

- Go to the column specified by the *Number* parameter:

```
25|
```

- Scroll the number of lines up or down specified by the *Number* parameter:

```
10Ctrl-U
10Ctrl-D
```

vi Editor Subcommands

Use the subcommands to perform these kinds of actions:

- Moving the cursor
- Editing text
- Manipulating files
- Other actions

Moving the Cursor

Use subcommands to move the cursor within a file in these ways:

- Moving within a line
- Moving within a line by character position
- Moving to words
- Moving by line position
- Moving to sentences, paragraphs, or sections
- Moving by redrawing the screen
- Paging and scrolling
- Searching for patterns
- Marking a specific location in a file and returning

Moving within a Line: Enter the following subcommands in command mode. You can cancel an incomplete command by pressing the Esc key. If you need information about the format of vi subcommands, see ″vi General Subcommand Syntax.″

Left Arrow or **h** or **Ctrl-H**	Moves the cursor one character to the left.
Down Arrow or **j** or **Ctrl-J** or **Ctrl-N**	Moves the cursor down one line (it remains in the same column).
Up Arrow or **k** or **Ctrl-P**	Moves the cursor up one line (it remains in the same column).
Right Arrow or **l**	Moves the cursor one character to the right.

Moving within a Line by Character Position: Enter the following subcommands in command mode. You can cancel an incomplete command by pressing the Esc key. If you need information about the format of vi subcommands, see ″vi General Subcommand Syntax.″

^	Moves the cursor to the first nonblank character.
0	Moves the cursor to the beginning of the line.
$	Moves the cursor to the end of the line.
f*x*	Moves the cursor to the next *x* character.
F*x*	Moves the cursor to the last *x* character.

t*x* Moves the cursor to one column before the next *x* character.
T*x* Moves the cursor to one column after the last *x* character.
; Repeats the last **f**, **F**, **t**, or **T** subcommand.
, Repeats the last **f**, **F**, **t**, or **T** subcommand in the opposite direction.
*Number***|** Moves the cursor to the specified column.

Moving to Words: Enter the following subcommands in command mode. If you need information about the format of vi subcommands, ″vi General Subcommand Syntax.″

w Moves the cursor to the next small word.
b Moves the cursor to the previous small word.
e Moves the cursor to the next end of a small word.
W Moves the cursor to the next big word.
B Moves the cursor to the previous big word.
E Moves the cursor to the next end of a big word.

Moving by Line Position: Enter the following subcommands in command mode. If you need information about the format of vi subcommands, see ″vi General Subcommand Syntax.″

H Moves the cursor to the top line on the screen.
L Moves the cursor to the last line on the screen.
M Moves the cursor to the middle line on the screen.
+ Moves the cursor to the next line at its first nonblank character.
- Moves the cursor to the previous line at its first nonblank character.
Enter Moves the cursor to the next line at its first nonblank character.

Moving to Sentences, Paragraphs, or Sections: Enter the following subcommands in command mode. You can cancel an incomplete subcommand by pressing the Esc key. If you need information about the format of vi subcommands, see ″vi General Subcommand Syntax.″

(Places the cursor at the beginning of the previous sentence, or the previous s-expression if you are in LISP mode.
) Places the cursor at the beginning of the next sentence, or the next s-expression if you are in LISP mode.
{ Places the cursor at the beginning of the previous paragraph, or at the next list if you are in LISP mode.
} Places the cursor at the beginning of the next paragraph, at the next section if you are in C mode, or at the next list if you are in LISP mode.
]] Places the cursor at the next section, or function if you are in LISP mode.
[[Places the cursor at the previous section, or function if you are in LISP mode.

Moving by Redrawing the Screen: Enter the following subcommands in command mode. You can cancel an incomplete subcommand by pressing the Esc key. If you need information about the format of vi subcommands, see ″vi General Subcommand Syntax.″

z Redraws the screen with the current line at the top of the screen.
z- Redraws the screen with the current line at the bottom of the screen.
z. Redraws the screen with the current line at the center of the screen.
/*Pattern***/z-** Redraws the screen with the line containing the character string, specified by the *Pattern* parameter, at the bottom.

Paging and Scrolling: Enter the following subcommands in command mode. You can cancel an incomplete subcommand by pressing the Esc key. If you need information about the format of vi subcommands, see ″vi General Subcommand Syntax.″

Ctrl-U Scrolls up one-half screen.
Ctrl-D Scrolls down one-half screen.
Ctrl-F Scrolls forward one screen.
Ctrl-B Scrolls backward one screen.
Ctrl-E Scrolls the window down one line.
Ctrl-Y Scrolls the window up one line.
z+ Pages up.
z^ Pages down.

Searching for Patterns: Enter the following subcommands in command mode. You can cancel an incomplete subcommand by pressing the Esc key. If you need information about the format of vi subcommands, see ″vi General Subcommand Syntax.″

[*Number***]G**	Places the cursor at the line number specified by the *Number* parameter or at the last line if the *Number* parameter is not specified.
/*Pattern*	Places the cursor at the next line containing the character string specified by the *Pattern* parameter.
?*Pattern*	Places the cursor at the next previous line containing the character string specified by the *Pattern* parameter.
n	Repeats the last search for the text specified by the *Pattern* parameter in the same direction.
N	Repeats the last search for the text specified by the *Pattern* parameter in the opposite direction.
/*Pattern***/+***Number*	Places the cursor the specified number of lines after the line matching the character string specified by the *Pattern* parameter.
?*Pattern***?-***Number*	Places the cursor the specified number of lines before the line matching the character string specified by the *Pattern* parameter.
%	Finds the parenthesis or brace that matches the one at current cursor position.

Editing Text

The subcommands for editing enable you to perform the following tasks:

- Marking a specific location in a file and returning
- Adding text to a file
- Changing text while in input mode
- Changing text from command mode
- Copying and moving text
- Restoring and repeating changes

Marking a Specific Location in a File and Returning: Enter the following subcommands in command mode. You can cancel an incomplete subcommand by pressing the Esc key. If you need information about the format of vi subcommands, see ″vi General Subcommand Syntax.″

″	Moves the cursor to the previous location of the current line.
″	Moves the cursor to the beginning of the line containing the previous location of the current line.
m*x*	Marks the current position with the letter specified by the *x* parameter.
`*x*	Moves the cursor to the mark specified by the *x* parameter.
’*x*	Moves the cursor to the beginning of the line containing the mark specified by the *x* parameter.

Adding Text to a File (Text Input Mode): Enter the following subcommands in command mode to change the vi editor into text input mode. If you need information about the format of vi subcommands, see ″vi General Subcommand Syntax.″

a*Text* Inserts text specified by the *Text* parameter after the cursor. End text input mode by pressing the Esc key.

A*Text* Adds text specified by the *Text* parameter to the end of the line. End text input mode by pressing the Esc key.

i*Text* Inserts text specified by the *Text* parameter before the cursor. End text input mode by pressing the Esc key.

I*Text* Inserts text specified by the *Text* parameter before the first nonblank character in the line. End text input mode by pressing the Esc key.

o Adds an empty line below the current line. End text input mode by pressing the Esc key.

O Adds an empty line above the current line. End text input mode by pressing the Esc key.

Changing Text While in Input Mode: Use the following subcommands only while in text input mode. These commands have different meanings in command mode. If you need information about the format of vi subcommands, see ″vi General Subcommand Syntax.″

Ctrl-D Goes back to previous autoindent stop.

^ Ctrl-D Ends autoindent for this line only.

0Ctrl-D Moves cursor back to left margin.

Esc Ends insertion and returns to command state.

Ctrl-H Erases the last character.

Ctrl-Q Enters any character if xon is disabled.

Ctrl-V Enters any character.

Ctrl-W Erases the last small word.

**** Quotes the erase and kill characters.

Ctrl-? Interrupts and ends insert or the Ctrl-D key sequence.

Changing Text from Command Mode: Use the following subcommands in command mode. An incomplete subcommand can be canceled by pressing the Esc key. If you need information about the format of vi subcommands, see ″vi General Subcommand Syntax.″

C Changes the rest of the line (same as **c$**).

cc Changes a line.

cw Changes a word.

cw*Text* Changes a word to the text specified by the *Text* parameter.

D Deletes the rest of the line (same as **d$**).

dd Deletes a line.

dw Deletes a word.

J Joins lines.

r*x* Replaces the current character with the character specified by *x*.

R*Text* Overwrites characters with the text specified by the *Text* parameter.

s Substitutes characters (same as **cl**).

S Substitutes lines (same as **cc**).

u Undoes the previous change.

x Deletes a character at the cursor.

X Deletes a character before the cursor (same as **dh**).

<< Shifts one line to the left.

<L Shifts all lines from the cursor to the end of the screen to the left.

>> Shifts one line to the right.

>L Shifts all lines from the cursor to the end of the screen to the right.

~ Changes letter at the cursor to the opposite case.

! Indents for LISP.

Copying and Moving Text: Use the following subcommands in command mode. An incomplete subcommand can be canceled by pressing the Esc key. If you need information about the format of vi subcommands, see ″vi General Subcommand Syntax.″

p Puts back text from the undo buffer after the cursor.
P Puts back text from the undo buffer before the cursor.
"*x***p** Puts back text from the *x* buffer.
"*x***d** Deletes text into the *x* buffer.
y Places the object that follows (for example, **w** for word) into the undo buffer.
"*x***y** Places the object that follows into the *x* buffer, where *x* is any letter.
Y Places the line in the undo buffer.

Restoring and Repeating Changes: Use the following subcommands in command mode. An incomplete subcommand can be canceled by pressing the Esc key. If you need information about the format of vi subcommands, see "vi General Subcommand Syntax."

u Undoes the last change.

Tip: After an undo, the cursor moves to the first non-blank character on the updated current line.

U Restores the current line if the cursor has not left the line since the last change.
. Repeats the last change or increments the "*n***p** command.

Notes:

1. This subcommand will repeat the last change, including an undo. Therefore, after an undo, repeat performs an undo rather than repeat the last change.
2. This subcommand is not meant for use with a macro. Enter **@@** (two at signs) to repeat a macro.

"*n* **p** Retrieves the *n*th last delete of a complete line or block of lines.

Manipulating Files

The subcommands for manipulating files allow you to do the tasks outlined in the following sections:

- Saving changes to a file
- Editing a second file
- Editing a list of files
- Finding file information

Saving Changes to a File: Use the following subcommands in command mode. If you need information about the format of vi subcommands, see "vi General Subcommand Syntax."

:w Writes the edit buffer contents to the original file. If you are using this subcommand within the ex editor, you do not need to type the : (colon).
:w *File* Writes the edit buffer contents to the file specified by the *File* parameter. If you are using this subcommand within the ex editor, you do not need to type the : (colon).
:w! *File* Overwrites the file specified by the *File* parameter with the edit buffer contents. If you are using this subcommand within the ex editor, you do not need to type the : (colon).

Editing a Second File: Enter the following subcommands in command mode. If you need information about the format of vi subcommands, see "vi General Subcommand Syntax."

:e *File* Edits the specified file. If you are using this subcommand from the ex editor, you do not need to type the : (colon).
:e! Re-edits the current file and discards all changes.
:e + *File* Edits the specified file starting at the end.
:e + *Number File* Edits the specified file starting at the specified line number.
:e # Edits the alternate file. The alternate file is usually the previous file name before accessing another file with a **:e** command. However, if changes are pending on the current file when a new file is called, the new file becomes the alternate file. This subcommand is the same as the **Ctrl-A** subcommand.

:r *File* Reads the file into the editing buffer by adding new lines below the current line. If you are using this subcommand from the ex editor, you do not need to type the **:** (colon).

:r !*Command* Runs the specified command and places its output into the file by adding new lines below the current cursor position.

:ta *Tag* Edits a file containing the *Tag* tag starting at the location of the tag. To use this subcommand, you must first create a database of function names and their locations using the **ctags** command. If you are using this subcommand from the ex editor, you do not need to type the **:** (colon).

Ctrl-] Edits a file containing the tag associated with the current word starting at the location of the tag. To use this subcommand, you must first create a database of function names and their locations using the **ctags** command. Ctrl-T edits a file at the editing position where the previous Ctrl-] subcommand was issued. If multiple Ctrl-] subcommands have been issued, then multiple Ctrl-T subcommands can be used to return to previous editing positions where Ctrl-] subcommands were issued.

Ctrl-A Edits the alternate file. The alternate file is usually the previous current file name. However, if changes are pending on the current file when a new file is called, the new file becomes the alternate file. This subcommand is the same as the **:e #** subcommand.

Editing a List of Files: Enter the following subcommands in command mode. If you need information about the format of vi subcommands, see ″vi General Subcommand Syntax.″

:n Edits the next file in the list entered on the command line. If you are using this subcommand from the ex editor, a : (colon) is not needed.

:n *Files* Specifies a new list of files to edit. If you are using this subcommand from the ex editor, a **:** (colon) is not needed.

Finding File Information: Enter the following subcommand in command mode. If you need information about the format of vi subcommands, see ″vi General Subcommand Syntax″.

Ctrl-G Shows the current file name, current line number, number of lines in the file, and percentage of the way through the file where the cursor is located.

Other Actions

The vi editor provides the subcommands described in the following sections:

- Adjusting the screen
- Entering shell commands
- Interrupting and ending the vi editor

Adjusting the Screen: Enter the following subcommands in command mode. An incomplete subcommand can be canceled by pressing the Esc key. If you need information about the format of vi subcommands, see ″vi General Subcommand Syntax.″

Ctrl-L Clears and redraws the screen.

Ctrl-R Redraws the screen and eliminates blank lines marked with @ (at sign).

z*Number* Makes the window the specified number of lines long.

Entering Shell Commands: The following subcommands allow you to run a command within the vi editor. Enter these subcommands in command mode. If you need information about the format of vi subcommands, see ″vi General Subcommand Syntax.″

:sh Enters the shell to allow you to run more than one command. You can return to the vi editor by pressing the Ctrl-D key sequence. If you are using this subcommand within the ex editor, a : (colon) is not needed.

:!*Command* Runs the specified command and then returns to the vi editor. If you are using this subcommand within the ex editor, a : (colon) is not needed.

Tip: The # (alternate file), **%** (current file), and **!** (previous command) special characters are expanded when following a **:!** subcommand. To prevent any of these characters from being expanded, use the **** (backslash).

:!! Repeats the last **:!***Command* subcommand**.**

*Number***!!***Command* Runs the specified command and replaces the lines specified by *Number* with the output of the command**.** If a number is not specified, the default value is 1. If the command expects standard input, the specified lines are used as input.

!*Object Command* Runs the specified command and replaces the object specified by the *Object* parameter with the output of the command**.** If the command expects standard input, the specified object is used as input.

Interrupting and Ending the vi Editor: Enter the following subcommands in command mode. If you need information about the format of vi subcommands, see ″vi General Subcommand Syntax.″

Q Enters the ex editor in command mode.

ZZ Exits the vi editor, saving changes.

:q Quits the vi editor. If you have changed the contents of the editing buffer, the vi editor displays a warning message and does not quit. If you are using this subcommand from the ex editor, a : (colon) is not needed.

:q! Quits the vi editor, discarding the editing buffer. If you are using this subcommand from the ex editor, a : (colon) is not needed.

Esc Ends text input or ends an incomplete subcommand.

Ctrl-? Interrupts a subcommand.

Exit Status

The following exit values are returned:

0 Indicates successful completion.

>0 Indicates an error occurred.

Input Files

Input files must be text files or files that are similar to text files except for an incomplete last line that contains no null characters.

The **.exrc** files must be text files consisting of **ex** commands.

The **$HOME/.vi_history** file is an auto-generated text file that records the last line mode command history.

By default, the vi editor reads lines from the files to be edited without interpreting any of those lines as any form of vi editor command.

Related Information

The **ctags** command, **ed** command, **ex** command, **sed** command, **tvi** command, **view** command.

The **.profile** file.

view Command

Purpose

Starts the vi editor in read-only mode.

Syntax

view [**-c***Subcommand*] [**-l**] [**-t** *Tag*] [**-w***Number*] [**-y**] [**-r** [*File*]] [**+** [*Subcommand*]] [*File* ...]

Description

The **view** command starts the vi full-screen editor in read-only mode. The read-only mode is only advisory to prevent accidental changes to the file. To override read-only mode, use the ! (exclamation point) when executing a command. The *File* parameter specifies the name of the file you want to browse. Use vi subcommands for moving within the file. Use the **:q** subcommand to exit the **view** command. If you modify the file you can save your modifications by pressing the Esc key and `wq!`.

Flags

-c*Subcommand*	Carries out the ex editor subcommand before viewing with vi begins. When a null operand is entered, as in `-c ''` , the editor places the cursor on the last line of the file.
-l	Enters a version of the vi editor with specialized features designed for writing programs in the LISP language. In this mode, the vi editor indents appropriately for LISP programming, and the **(**, **)**, **{**, **}**, **[[**, and **]]** subcommands are modified to act appropriately for LISP.
-r [*File*]	Recovers a file after an editor or system crash. If you do not specify a *File* parameter, the editor displays a list of all saved files.
-t*Tag*	Edits the file containing the tag specified by the *Tag* parameter and positions the editor at its definition. To use this flag, you must first create a database of function names and their locations using the **ctags** command.
-w*Number*	Sets the default window size to the value specified by the *Number* parameter. This is useful when your terminal communicates with the system running the editor over a slow communications line.
-y	Overrides the maximum line setting of 1,048,560 with any value greater than 1024.
+[*Subcommand*]	Carries out the ex editor subcommand specified by the *Subcommand* parameter before viewing with vi begins. If you do not specify a subcommand, the cursor is placed on the last line of the file.

Related Information

The **vi** command, **ctags** command.

vmh Command

Purpose

Starts a visual interface for use with MH commands.

Syntax

vmh [**-prompt** *String*] [**-vmhproc** *CommandString* | **-novmhproc**]

Description

The **vmh** command starts a visual interface for use with MH commands. The **vmh** command implements the server side of the MH window management protocol and maintains a split-screen interface to any program that implements the client side of the protocol.

The **vmh** command prompts for commands and sends them to the client side of the protocol. If the command produces a window with more than one screen of output, the **vmh** command prompts the user for a subcommand. The **vmh** subcommands enable you to display specific portions of the command output.

vmh Subcommands

Ctrl-L	Refreshes the screen.
Space	Advances to the next screen.
[*Number*] **Enter**	Advances the specified number of lines. The default is one line.
[*Number*] **d**	Advances 10 times the specified number of lines. The default for the *Number* variable is 1, for a total of 10 lines.
[*Number*] **g**	Goes to the specified line.
[*Number*] **G**	Goes to the end of the window. If the *Number* variable is specified, this command acts like the **g** flag.
[*Number*] **u**	Goes back 10 times the specified number of lines. The default for the *Number* variable is 1, for a total of 10 lines.
[*Number*] **y**	Goes back the specified number of lines. The default is one line.
h	Displays a help message.
q	Ends output.

Flags

-help	Lists the command syntax, available switches (toggles), and version information. **Note:** For MH, the name of this flag must be fully spelled out.
-novmhproc	Runs the default **vmproc** without the window management protocol.
-prompt *String*	Uses the specified string as the prompt.
-vmhproc *CommandString*	Specifies the program that implements the client side of the window management protocol. The default is the **msh** program.

Profile Entries

The following entries are entered in the *UserMhDirectory*/**.mh_profile** file:

`Path:`	Specifies the user's MH directory.
`mshproc:`	Specifies the program used for the MH shell.

Files

$HOME/.mh_profile	Contains the MH user profile.
/usr/bin/vmh	Contains the **vmh** command.

Related Information

The **msh** command.

The **mh_alias** file format, **mh_profile** file format.

Mail applications in *Networks and communication management*.

vmo Command

Purpose

Manages Virtual Memory Manager tunable parameters.

Syntax

vmo [**-p** | **-r**] [**-y**] { **-o** *Tunable* [**=** *Newvalue*] }

vmo [**-p** | **-r**] [**-y**] { **-d** *Tunable* }

vmo [**-p** | **-r**] [**-y**] **-D**

vmo [**-p** | **-r**] [**-F**] **-a**

vmo -h [*Tunable*]

vmo [**-F**] **-L** [*Tunable*]

vmo [**-F**] **-x** [*Tunable*]

Note: Multiple **-o**, **-d**, **-x** and **-L** are allowed.

Description

Note: The **vmo** command can only be executed by root.

Use the **vmo** command to configure Virtual Memory Manager tuning parameters. This command sets or displays current or next boot values for all Virtual Memory Manager tuning parameters. This command can also make permanent changes or defer changes until the next reboot. Whether the command sets or displays a parameter is determined by the accompanying flag. The **-o** flag performs both actions. It can either display the value of a parameter or set a new value for a parameter.

The Virtual Memory Manager (VMM) maintains a list of free real-memory page frames. These page frames are available to hold virtual-memory pages needed to satisfy a page fault. When the number of pages on the free list falls below that specified by the **minfree** parameter, the VMM begins to steal pages to add to the free list. The VMM continues to steal pages until the free list has at least the number of pages specified by the **maxfree** parameter.

If the number of file pages (permanent pages) in memory is less than the number specified by the **minperm%** parameter, the VMM steals frames from either computational or file pages, regardless of repage rates. If the number of file pages is greater than the number specified by the **maxperm%** parameter, the VMM steals frames only from file pages. Between the two, the VMM normally steals only file pages, but if the repage rate for file pages is higher than the repage rate for computational pages, computational pages are stolen as well.

You can also modify the thresholds that are used to decide when the system is running out of paging space. The **npswarn** parameter specifies the number of paging-space pages available at which the system begins warning processes that paging space is low. The **npskill** parameter specifies the number of paging-space pages available at which the system begins killing processes to release paging space.

Note: Options **-o**, **-d**, and **-D** which attempt to change the value of a virtual memory manager tunable parameter are not supported within a workload partition.

Understanding the Effect of Changing Tunable Parameters

Misuse of this command can cause performance degradation or operating-system failure. Before experimenting with **vmo**, you should be thoroughly familiar with both Performance overview of the Virtual Memory Manager and Enhanced JFS file system cache limit with the maxclient parameter.

Before modifying any tunable parameter, you should first carefully read about all its characteristics in the Tunable Parameters section below, and follow any Refer To pointer, in order to fully understand its purpose.

You must then make sure that the Diagnosis and Tuning sections for this parameter truly apply to your situation and that changing the value of this parameter could help improve the performance of your system.

If the Diagnosis and Tuning sections both contain only ″N/A″, you should probably never change this parameter unless specifically directed by AIX development.

Flags

-a Displays current, reboot (when used with the **-r** option), or permanent (when used with the **-p** option) values for all tunable parameters, one per line in pairs *Tunable* **= *Value***. For the permanent option, a value is only displayed for a parameter if its reboot and current values are equal. Otherwise, NONE is displayed as the value.

-d *Tunable* Resets *Tunable* to default value. If a *Tunable* needs to be changed (that is, it is currently not set to its default value) and is of type **Bosboot** or **Reboot**, or if it is of type **Incremental** and has been changed from its default value, and **-r** is not used in combination, it will not be changed but a warning will be displayed instead.

-D Resets all tunables to their default value. If tunables needing to be changed are of type **Bosboot** or **Reboot**, or are of type Incremental and have been changed from their default value, and **-r** is not used in combination, they won't be changed but a warning will be displayed instead.

-F Forces display of the restricted tunable parameters when the **-a**, **-L** or **-x** options are specified alone on the command line to list all tunables. When the **-F** flag is not specified, restricted tunables are not displayed, unless these restricted tunables are specifically named with a display option.

-h [*Tunable*] Displays help about the *Tunable* parameter if one is specified. Otherwise, displays the **vmo** command usage statement.

-L [*Tunable*] Lists the characteristics of one or all tunables, one per line, using the following format:

```
NAME                      CUR    DEF    BOOT   MIN    MAX    UNIT           TYPE
     DEPENDENCIES
------------------------------------------------------------------------------
memory_frames             128K            128K                 4KB pages         S
------------------------------------------------------------------------------
maxfree                   1088   1088    130    16     200K   4KB pages         D
     minfree
     memory_frames
------------------------------------------------------------------------------
minfree                   960    960     122    8      200K   4KB pages         D
     maxfree
     memory_frames
------------------------------------------------------------------------------
...
where:
    CUR = current value
    DEF = default value
    BOOT = reboot value
    MIN = minimal value
    MAX = maximum value
    UNIT = tunable unit of measure
    TYPE = parameter type: D (for Dynamic), S (for Static), R for Reboot),
           B (for Bosboot), M (for Mount), I (for Incremental),
           C (for Connect), and d (for Deprecated)
    DEPENDENCIES = list of dependent tunable parameters, one per line
```

-o *Tunable*[**=***Newvalue*] Displays the value or sets tunable to *Newvalue*. If a tunable needs to be changed (the specified value is different than current value), and is of type **Bosboot** or **Reboot**, or if it is of type **Incremental** and its current value is bigger than the specified value, and **-r** is not used in combination, it will not be changed but a warning will be displayed instead.

When **-r** is used in combination without a new value, the nextboot value for tunable is displayed. When **-p** is used in combination without a new value, a value is displayed only if the current and next boot values for tunable are the same. Otherwise NONE is displayed as the value.

-p When used in combination with **-o**, **-d** or **-D**, makes changes apply to both current and reboot values, that is, turns on the updating of the **/etc/tunables/nextboot** file in addition to the updating of the current value. These combinations cannot be used on **Reboot** and **Bosboot** type parameters because their current value can't be changed.

When used with **-a** or **-o** without specifying a new value, values are displayed only if the current and next boot values for a parameter are the same. Otherwise NONE is displayed as the value.

-r When used in combination with **-o**, **-d** or **-D**, makes changes apply to reboot values, for example, turns on the updating of the **/etc/tunables/nextboot** file. If any parameter of type **Bosboot** is changed, the user will be prompted to run bosboot.

When used with **-a** or **-o** without specifying a new value, next boot values for tunables are displayed instead of current values.

-x [*Tunable*] Lists characteristics of one or all tunables, one per line, using the following (spreadsheet) format:

```
tunable,current,default,reboot,min,max,unit,type,{dtunable }

where:
    current = current value
    default = default value
    reboot = reboot value
    min = minimal value
    max = maximum value
    unit = tunable unit of measure
    type = parameter type: D (for Dynamic), S (for Static), R (for Reboot),
                B (for Bosboot), M (for Mount), I (for Incremental),
                C (for Connect), and d (for Deprecated)
    dtunable = list of dependent tunable parameters
```

-y Suppresses the confirmation prompt before running the **bosboot** command.

Any change (with the **-o**, **-d**, or **-D** options) to a restricted tunable parameter results in a warning message to the user that a tunable of the restricted use type has been modified. If the **-r** or **-p** options are also specified, the user will be prompted to confirm the change. In addition, at system reboot, restricted tunables displayed in the **/etc/tunables/nextboot** file, which were modified to values that are different from their default values (using a command line specifying the **-r** or **-p** options) causes an error log entry that identifies the list of these modified tunables.

When modifying tunable, the tunable value might be specified using abbreviations such as K, M, G, T, P and E to indicate units. See the following lists for abbreviations and their correspondent values:

- $K=2^{10}$
- $M=2^{20}$
- $G=2^{30}$
- $T=2^{40}$
- $P=2^{50}$
- $E=2^{60}$

Thus, a tunable value of 1024 might be specified as 1K.

Any change (with **-o**, **-d** or **-D**) to a parameter of type Mount will result in a message being displayed to warn the user that the change is only effective for future mountings.

Any change (with **-o**, **-d** or **-D** flags) to a parameter of type Connect will result in **inetd** being restarted, and a message displaying a warning to the user that the change is only effective for future socket connections.

Any attempt to change (with **-o**, **-d** or **-D**) a parameter of type **Bosboot** or **Reboot** without **-r**, will result in an error message.

Any attempt to change (with **-o**, **-d** or **-D** but without **-r**) the current value of a parameter of type Incremental with a new value smaller than the current value, will result in an error message.

Tunable Parameters Type

All the tunable parameters manipulated by the tuning commands (**no**, **nfso**, **vmo**, **ioo**, **raso**, and **schedo**) have been classified into these categories:

Dynamic	If the parameter can be changed at any time
Static	If the parameter can never be changed
Reboot	If the parameter can only be changed during reboot

Bosboot	If the parameter can only be changed by running bosboot and rebooting the machine
Mount	If changes to the parameter are only effective for future file systems or directory mounts
Incremental	If the parameter can only be incremented, except at boot time
Connect	If changes to the parameter are only effective for future socket connections
Deprecated	If changing this parameter is no longer supported by the current release of AIX.

For parameters of type Bosboot, whenever a change is performed, the tuning commands automatically prompt the user to ask if they want to execute the **bosboot** command. For parameters of type Connect, the tuning commands automatically restart the **inetd** daemon.

Note that the current set of parameters managed by the **vmo** command only includes Static, Dynamic, and Bosboot types.

Compatibility Mode

When running in pre-AIX 5.2 compatibility mode (controlled by the **pre520tune** attribute of **sys**0), reboot values for parameters, except those of type Bosboot, are not really meaningful because in this mode they are not applied at boot time. For more information, see AIX 5.2 compatibility mode in the *Performance management*.

In pre-AIX 5.2 compatibility mode, setting reboot values to tuning parameters continues to be achieved by imbedding calls to tuning commands in scripts called during the boot sequence. Parameters of type **Reboot** can therefore be set without the **-r** flag, so that existing scripts continue to work.

This mode is automatically turned ON when a machine is migrated to AIX 5.2. For complete installations, it is turned OFF and the reboot values for parameters are set by applying the content of the **/etc/tunables/nextboot** file during the reboot sequence. Only in that mode are the **-r** and **-p** flags fully functional. For more information, see Kernel Tuning in *AIX 5L Version 5.3 Performance Tools Guide and Reference*.

Tunable Parameters

For a full list of the system tunable parameters interfaced through this command, and details of their use and characteristic values, run the **vmo** command with the appropriate list (**-a**, **-L**, **-x**) or help (**-h** <*tunable_parameter_name*>) options.

Security

RBAC Disclaimer

Examples

1. To list the current and reboot value, range, unit, type and dependencies of all tunable parameters managed by the **vmo** command, enter:

   ```
   vmo -L
   ```

2. To turn on and reserve 16MB large pages on a POWER4 system, enter:

   ```
   vmo -r -o lgpg_regions=10 -o lgpg_size=16777216
   ```

   This command will propose **bosboot** to the user, and warn that a reboot is necessary before the change will be effective.

   Note: The **-r** flag (and subsequent reboot) is not necessary for AIX 5.3 and later releases.

3. To display help on **nokilluid**, enter:

   ```
   vmo -h nokilluid
   ```

4. To turn on **v_pinshm** after the next reboot, enter:

   ```
   vmo -r -o v_pinshm=1
   ```

5. To permanently reset all **vmo** tunable parameters to default, enter:

   ```
   vmo -p -D
   ```

6. To list the reboot value for all virtual Memory Manager tuning parameters, enter:

   ```
   vmo -r -a
   ```

7. To list (spreadsheet format) the current and reboot value, range, unit, type and dependencies of all tunable parameters managed by the **vmo** command, enter:

   ```
   vmo -x
   ```

Related Information

The **ioo** command, **schedo** command, **no** command, **nfso** command, **raso** command, **tunchange** command, **tunsave** command, **tunrestore** command, **tuncheck** command, and **tundefault** command.

Performance Overview of the Virtual Memory Manager (VMM) in *Performance management*

Kernel Tuning in *AIX 5L Version 5.3 Performance Tools Guide and Reference*.

AIX 5.2 compatibility mode in the *Performance management*.

---

vmstat Command

Purpose

Reports virtual memory statistics.

Syntax

vmstat [**-f**] [**-i**] [**-s**] [**-I**] [**-t**] [**-v**] [**-h**] [**-w**] [**-l**] [**-c**][**-@** *wparname* | **ALL**] [{ **-p** | **-P** } *pagesize* | **ALL**] **ALL**] [**-S** *power*] [*physicalvolume ...*] [*interval* [*count*]]

Note: Do not use the *wparname* parameter and the **-i** flag together inside workload partitions.

Description

The **vmstat** command reports statistics about kernel threads, virtual memory, disks, hypervisor pages, traps and processor activity. Reports generated by the **vmstat** command can be used to balance system load activity. These system-wide statistics (among all processors) are calculated as averages for values expressed as percentages, and as sums otherwise. The **vmstat** command might return inconsistent statistics because the statistics are not read atomically.

If you run the **vmstat** command without flags, the report contains a summary of the virtual memory activity since system startup. If you specify the **-f** flag, the **vmstat** command reports the number of forks since system startup. The *physicalvolume* parameter specifies the name of the physical volume.

The *interval* parameter specifies the amount of time in seconds between each report. If you do not specify the *interval* parameter, the **vmstat** command generates a single report that contains statistics for the time since system startup and then exits. You can specify the *count* parameter only with the *interval* parameter. If you specify the *count* parameter, its value determines the number of reports generated and the number of seconds apart. If you specify the *interval* parameter without the *count* parameter, reports are continuously generated. Do not specify a value of zero to the *count* parameter.

AIX 4.3.3 and later contain enhancements to the method used to compute the percentage of processor time spent waiting on disk I/O (*wio* time). The method used in AIX 4.3.2 and earlier versions of the operating system can, under certain circumstances, give an inflated view of wio time on SMPs.

The method used in AIX 4.3.2 and earlier versions is as follows: At each clock interrupt on each processor (100 times a second per processor), a determination is made as to which of the four categories (usr/sys/wio/idle) to place the last 10 ms of time. If the processor was busy in usr mode at the time of the clock interrupt, then usr gets the clock tick added into its category. If the processor was busy in kernel mode at the time of the clock interrupt, then the sys category gets the tick. If the processor was not busy, a check is made to see if any I/O to disk is in progress. If any disk I/O is in progress, the wio category is incremented. If no disk I/O is in progress and the processor is not busy, the idle category gets the tick. The inflated view of wio time results from all idle processors being categorized as wio regardless of the number of threads waiting on I/O. For example, systems with just one thread doing I/O could report over 90 percent wio time regardless of the number of processors it has. The wio time is reported by the commands **sar** (*%wio*), **vmstat** (*wa*) and **iostat** (*% iowait*).

The kernel maintains statistics for kernel threads, paging, and interrupt activity, which the **vmstat** command accesses through the use of the **perfstat** kernel extension. The disk input/output statistics are maintained by device drivers. For disks, the average transfer rate is determined by using the active time and number of transfers information. The percent active time is computed from the amount of time the drive is busy during the report.

Beginning with AIX 5.3, the **vmstat** command reports the number of physical processors consumed (pc), and the percentage of entitlement consumed (ec), in Micro-Partitioning environments. These metrics will only be displayed on Micro-Partitioning environments.

The report generated by the **vmstat** command contains a system configuration row and column headings. If the **-@** flag is specified, the report will consist of system configuration and WPAR configuration. The system configuration row has the following values:

lcpu	Indicates the number of logical processors.
mem	Indicates the amount of memory.
tmem	Indicates the true memory size of the LPAR. **Note:** This is available only when the **-c** option is provided and Active Memory Expansion is enabled.
ent	Displays only when the partition is running with shared processor.
drives	Displays only when physical volume name is monitored.
WPARs	Indicates the number of active workload partitions. It is displayed only when the **-@** flag is specified.
memlim	Indicates the limit of the memory resource of the workload partition. The limit is in megabytes (MB). This information is displayed only for the WPAR that has enforced memory resource limit.
cpulim	Indicates the limit of processor resource of the workload partition in processor units. This information is displayed only for the WPAR that has enforced processor resource limit.
rset	Indicates the type of the **rset** registry that is associated with a WPAR. The type can be regular or exclusive. This information is displayed only for the WPARs that are associated with an **rset** registry.
mmode	Indicates memory mode. This metric is displayed automatically in a system with Active Memory Sharing enabled. This metric is also displayed when the**-c** option is used.
mpsz	Size of the memory pool in gigabytes. This metric is displayed only in shared-memory mode.

The followings are column headings and their descriptions:

WPAR Information about workload partitions. It displays only when the **-@** flag is specified.

WPAR Workload partition name.

Notes:

1. The *system* WPAR name indicates system-wide statistics. The *global* WPAR name indicates the statistics belong to Global only.
2. When the **vmstat** command is invoked with the **-@ ALL** option and the WPAR specific information is not available for a metric, then an en dash sign (-) is displayed instead of a value.
3. When the **vmstat** command is invoked with the **-@** *wparname* or invoked inside a WPAR, if the WPAR information is not available for a metric, then that metric is marked with the at sign (@), and the system-wide value is displayed for that metric.
4. If a metric is not supported, then a en dash sign (-) is displayed instead of a value.

kthr: Information about kernel thread states.

r Average number of runnable kernel threads over the sampling interval. Runnable threads consist of the threads that are ready but still waiting to run, and the threads that are already running.

b Average number of kernel threads placed in the Virtual Memory Manager (VMM) wait queue (awaiting resource, awaiting input/output) over the sampling interval.

Memory: Information about the usage of virtual and real memory. Virtual pages are considered active if they have been accessed. A page is 4096 bytes.

avm Active virtual pages.

fre Size of the free list.
Note: A large portion of real memory is utilized as a cache for file system data. It is not unusual for the size of the free list to remain small.

Page: Information about page faults and paging activity. These are averaged over the interval and given in units per second.

re Pager input/output list.
pi Pages paged in from paging space.
po Pages paged out to paging space.
fr Pages freed (page replacement).
sr Pages scanned by page-replacement algorithm.
cy Clock cycles by page-replacement algorithm.

Faults: Trap and interrupt rate averages per second over the sampling interval.

in Device interrupts.
sy System calls.
cs Kernel thread context switches.

CPU: Breakdown of percentage usage of processor time.

us User time.
sy System time.
id Processor idle time.

wa Processor idle time during which the system had outstanding disk/NFS I/O request(s). See detailed description above.

pc Number of physical processors consumed. Displayed only if the partition is running with shared processor.

ec The percentage of entitled capacity consumed. Displayed only if the partition is running with shared processor. Because the time base over which this data is computed can vary, the entitled capacity percentage can sometimes exceed 100%. This excess is noticeable only with small sampling intervals.

rc The percentage of processor resource that is consumed. This information is displayed only for the WPARs that have enforced processor resource limit.

Disk: Provides the number of transfers per second to the specified physical volumes that occurred in the sample interval. The *physicalvolume* parameter can be used to specify one to four names. Transfer statistics are given for each specified drive in the order specified. This count represents requests to the physical device. It does not imply an amount of data that was read or written. Several logical requests can be combined into one physical request. If the *physicalvolume* parameter is used, the physical volume names are printed at the beginning of command execution.

If the **-l** flag is specified, an I/O oriented view is presented with the following column changes.

kthr The column **p** will also be displayed besides columns **r** and **b**.

p Number of threads waiting on I/O to raw devices per second.

page New columns **fi** and **fo** will be displayed instead of **re** and **cy** columns.

fi File page-ins per second.

fo File page-outs per second.

If the **-c** flag is specified, then the Active Memory Expansion view is presented with the following column changes.

memory The columns **csz**, **cfr** and **dxm** will be displayed besides columns **avm** and **fre.**

csz Current compressed pool size, in 4K page units.

cfr Free pages available in compressed pool, in 4K page units

dxm Deficit in Expanded Memory Size, in 4K page units

page New columns **ci** and **co** will be displayed instead of **re** and **cy** columns.

ci Number of page-ins per second from compressed pool.

co Number of page-outs per second to compressed pool.

If, while the **vmstat** command is running, there is a change in system configuration that will affect the output, **vmstat** prints a warning message about the configuration change. It then continues the output, after printing the updated system configuration information and the header.

If the **-l** flag is specified, an additional ″large-page″ section is displayed with the following columns:

alp Indicates the number of large pages currently in use.

flp Indicates the number of large pages on the large page freelist.

If the **-p** option is specified, additional lines of VMM statistics are displayed for the specified page sizes. With **-l** and **-t** options, the **-p** option produces an additional line for the specified page size. This line contains the following VMM statistics relevant to the specified page size:

- **avm**
- **fre**
- **re**

- **fi**
- **fo**
- **pi**
- **po**
- **ci**
- **co**
- **fr**
- **sr**
- **cy**

Notes:

1. The display of the **re**, **fi**, **fo**, and **cy** options are affected by the **-l** option.
2. The display of the **re**, **ci**, **co**, and **cy** options are affected by the **-c** option.
3. If there is no resource control, then the **avm** and **fre** options are system-wide. Therefore with the **-@** option set, both the **avm** and **fre** options will be marked with the at sign (@).

These VMM statistics are preceded by a **psz** column and followed by an **siz** column. The description of these two columns follows:

psz Page size (for example, 4 KB, 64 KB).

siz Number of frames of the specified page size that exist on the system.

With the **-s** option, the **-p** option produces a separate stanza of output that contains only the statistics relevant to the specified page size. This additional stanza is preceded by a page size header.

The **-P** option produces the following report for the specified page size:

pgsz Indicates the page size (for example, 4 KB, 64 KB).

Memory
Indicates the memory statistics for the specified page sizes.

siz The number of frames of the specified page size that exist on the system.

avm Active virtual pages applicable to the specified page size.

fre Size of the free list for the specified page size.

Page Indicates the relevant page faults and paging activity for the specified page size. The page related columns **re**, **pi**, **po**, **fr**, **sr**, **cy**, **fi**, **fo**, **ci** and **co** are also applicable to this report.

Flags

Note: If the **-f** (or **-s**) flag is entered on the command line, then the system will only accept the **-f** (or **-s**) flag and will ignore other flags. If both the **-f** and **-s** flags are specified, the system will accept only the first flag and ignore the second flag.

-@ *wparname* Reports the Virtual Memory activity of a workload partition:

- The **-@ ALL** option indicates that the report pertains to the system and global environment, in addition to all of the workload partitions in the system.
  Note: The values that are system-wide statistics are marked with dash sign (-) against the WPAR section.
- The **-@** *wparname* flag indicates that the activity is only for that workload partition. In a workload partition, if you specify the **-@** flag, system-wide statistics and workload partition statistics are displayed. The system-wide statistics are marked with the at sign (@).
  Note: Do not use the **-@** flag with any combination of the **-i** flag.

-c Displays memory compression statistics with the new columns of output, **csz**, **cfr**, and **dxm** under the heading memory, and columns **ci** and **co** under the heading page instead of the columns **re** and **cy.**
Note: This option is available only when Active Memory Expansion is enabled.

-f Reports the number of forks since system startup.

-i Displays the number of interrupts taken by each device since system startup.
Note: The **-I**, **-t**, **-w**, and **-I** flags are ignored when they are specified with the **-i** flag.

-I Displays I/O oriented view with the new columns of output, **p** under heading `kthr`,and columns **fi** and **fo** under heading page instead of the columns **re** and **cy** in the page heading.

-l Displays an additional ″large-page″ section with the `alp` and `flp` columns.

-p *pagesize* Appends the VMM statistics for the specified page size to the regular **vmstat** output.

-P *pagesize* Displays only the VMM statistics which are relevant for the specified page size.

-s Writes to standard output the contents of the sum structure, which contains an absolute count of paging events since system initialization. The **-s** flag can only be used with the **-v** flag. These events are described as follows:

address translation faults
Incremented for each occurrence of an address translation page fault. I/O may or may not be required to resolve the page fault. Storage protection page faults (lock misses) are not included in this count.

-s *(Statistics displayed by* **-s***, continued)*:

backtracks
Incremented for each page fault that occurs while resolving a previous page fault. (The new page fault must be resolved first and then initial page faults can be *backtracked*.)

CPU context switches
Incremented for each processor context switch (dispatch of a new process).

decrementer interrupts
Incremented on each decrementer interrupt.

device interrupts
Incremented on each hardware interrupt.

executable-filled page faults
Incremented for each instruction page fault.

extend XPT waits
Incremented each time a process is waited by VMM due to a commit in progress for the segment being accessed.

free frame waits
Incremented each time a process requests a page frame, the free list is empty, and the process is forced to wait while the free list is replenished.

iodones
Incremented at the completion of each VMM I/O request.

mpc send interrupts
Incremented on each mpc send interrupt

mpc receive interrupts
Incremented on each mpc receive interrupt

page ins
Incremented for each page read in by the virtual memory manager. The count is incremented for page ins from page space and file space. Along with the page out statistic, this represents the total amount of real I/O initiated by the virtual memory manager.

page outs
Incremented for each page written out by the virtual memory manager. The count is incremented for page outs to page space and for page outs to file space. Along with the page in statistic, this represents the total amount of real I/O initiated by the virtual memory manager.

paging space page ins
Incremented for VMM initiated page ins from paging space only.

paging space page outs
Incremented for VMM initiated page outs to paging space only.

pages examined by the clock
VMM uses a clock-algorithm to implement a pseudo least recently used (lru) page replacement scheme. Pages are *aged* by being examined by the clock. This count is incremented for each page examined by the clock.

pages freed by the clock
Incremented for each page the clock algorithm selects to free from real memory.

pending I/O waits
Incremented each time a process is waited by VMM for a page-in I/O to complete.

-s *(Statistics displayed by* **-s***, continued)*:

phantom interrupts
Incremented on each phantom interrupt

revolutions of the clock hand
Incremented for each VMM clock revolution (that is, after each complete scan of memory).

start I/Os
Incremented for each read or write I/O request initiated by VMM.

syscalls
Incremented for each system call.

total reclaims
Incremented when an address translation fault can be satisfied without initiating a new I/O request. This can occur if the page has been previously requested by VMM, but the I/O has not yet completed; or if the page was pre-fetched by VMM's read-ahead algorithm, but was hidden from the faulting segment; or if the page has been put on the free list and has not yet been reused.

traps Not maintained by the operating system.

zero-filled page faults
Incremented if the page fault is to working storage and can be satisfied by assigning a frame and zero-filling it.

When the **-c** flag is specified along with the **-s** flag, the following additional metrics are displayed.

compressed pool page ins
Number of page-ins from Compressed Pool since system boot.

compressed pool page outs
Number of page-outs to Compressed Pool since system boot.

-s *(Statistics displayed by* **-s***, continued)*:

When used with the **-p** *pagesize* option, the **-s** option appends the sum structure for the specified page size to the system-wide sum structure. This additional stanza is preceded by a page size header (for example, 4K pages). The following details are not be displayed in this pagesize-based stanza as these statistics are not related to page sizes:

- Processor context switches
- Device interrupts
- Software interrupts
- Decrementer interrupts
- MPC-sent interrupts
- MPC-received interrupts
- Phantom interrupts
- Traps
- Syscalls

Notes:

1. When the **-s** flag is used with the **-@ ALL** option, the system-wide statistics are repeated in the workload partition section.
2. When the **-s** flag is used with the *wparname* option, all metrics are reported and the system-wide statistics are marked with the at sign (@).
3. When the **-s** flag is used with the **-l** flag, the **vmstat** command displays the following metric:

   large-page hi water count
   Specifies the maximum value of the large-page inuse count.

-S *power* Multiplies the statistics of the processor with a value of 10^{power}. The default value of the power is 0.

The following statistics are scaled:

- **us**
- **sy**
- **id**
- **wa**
- **pc**
- **ec**

Notes:

1. Do not use the **-S** flag with the **-f**, **-s**, **-i**, **-v**, or **-p** flags.
2. When the **-S** flag is specified, the **us**, **sy**, **id**, and **wa** statistics change. By default, the **us**, **sy**, **id**, and **wa** statistics are relative to the processor consumption of WPAR. When the **-S** flag is specified with a value of power that is not equal to zero, these statistics will be relative to system-wide processor consumption.
3. The value of power for **-S** flag can be only between 0 and 3.

-t Prints the time-stamp next to each line of output of **vmstat**. The time-stamp is displayed in the `HH:MM:SS` format.
Note: Time stamp will not be printed if **-f**, **-s**, or **-i** flags are specified.

-v Writes to standard output various statistics maintained by the Virtual Memory Manager. The **-v** flag can only be used with the **-s** and the **-h** flags.

If you specify the **-v** flag, the following statistics are displayed:

compressed percentage
Percentage of memory used by compressed pages.

client filesystem I/Os blocked with no fsbuf
Number of client filesystem I/O requests blocked because no fsbuf was available. NFS (Network File System) and VxFS (Veritas) are client filesystems. Fsbuf are pinned memory buffers used to hold I/O requests in the filesystem layer.

client pages
Number of client pages.

compressed pages
Number of compressed memory pages.

external pager filesystem I/Os blocked with no fsbuf
Number of external pager client filesystem I/O requests blocked because no fsbuf was available. JFS2 is an external pager client filesystem. Fsbuf are pinned memory buffers used to hold I/O requests in the filesystem layer.

file pages
Number of 4 KB pages currently used by the file cache.

free pages
Number of free 4 KB pages.

filesystem I/Os blocked with no fsbuf
Number of filesystem I/O requests blocked because no fsbuf was available. Fsbuf are pinned memory buffers used to hold I/O requests in the filesystem layer.

lruable pages
Number of 4 KB pages considered for replacement. This number excludes the pages used for VMM internal pages, and the pages used for the pinned part of the kernel text.

maxclient percentage
Tuning parameter (managed using vmo) specifying the maximum percentage of memory which can be used for client pages.

maxperm percentage
Tuning parameter (managed using **vmo**) in percentage of real memory. This specifies the point above which the page stealing algorithm steals only file pages.

maxpin percentage
Tuning parameter (managed using **vmo**) specifying the percentage of real memory which can be pinned.

memory pages
Size of real memory in number of 4 KB pages.

memory pools
Tuning parameter (managed using **vmo**) specifying the number of memory pools.

minperm percentage
Tuning parameter (managed using **vmo**) in percentage of real memory. This specifies the point below which file pages are protected from the re-page algorithm.

numclient percentage
Percentage of memory occupied by client pages.

numperm percentage
Percentage of memory currently used by the file cache.

-v *(Statistics displayed by* **-v***, continued)*:

paging space I/Os blocked with no psbuf
Number of paging space I/O requests blocked because no psbuf was available. Psbufs are pinned memory buffers used to hold I/O requests at the virtual memory manager layer.

pending disk I/Os blocked with no pbuf
Number of pending disk I/O requests blocked because no pbuf was available. Pbufs are pinned memory buffers used to hold I/O requests at the logical volume manager layer.

pinned pages
Number of pinned 4 KB pages.

remote pageouts scheduled
Number of pageouts scheduled for client file systems.

If you specify the **-h** flag with the **-v** flag, the following additional metrics are displayed:

Time resolving virtualized partition memory page faults
The total time that the virtual partition is blocked to wait for the resolution of its memory page fault. The time is measured in seconds, with millisecond granularity.

Virtualized partition memory page faults
The total number of virtual partition memory page faults that are recorded for the virtualized partition.

Number of 4 KB page frames loaned
The number of the 4 KB pages of partition's memory loaned to the hypervisor.

Percentage of partition memory loaned
The percentage of the partition's memory loaned to the hypervisor.

Notes:

1. When the **-v** flag is used with the **-@ ALL** option, the system-wide statistics are not repeated in the workload partition section.
2. When the **-s** flag is used with the *wparname* option, all metrics are reported and the system-wide statistics are marked with the at sign (@).

When the **-c** flag is specified along with the **-v** flag, the following additional metrics are displayed:

Compressed Pool Size
Size of Compressed Pool, in 4K page unit.

percentage of true memory used for compressed pool
Percentage of unexpanded memory used for compressed pool.

free pages in compressed pool (4K pages)
Number of free pages in compressed pool, in 4K page unit.

target memory expansion factor
Target memory expansion factor configured for the LPAR.

achieved memory expansion factor
Current memory expansion factor achieved.

-h Displays the **hypv-page** section that includes the hypervisor page information. The **hypv-page** section contains the following metrics:

hpi Number of hypervisor page-in per second.

hpit Average time spent in milliseconds per hypervisor page-in.

pmem Amount of physical memory that is backing the logical memory of partitions. The value is measured in gigabytes.

If you specify the **-h** flag with the **-v** flag, the following metrics are displayed in addition to the metrics that are displayed using the **-v** flag:

Time resolving virtualized partition memory page faults
The total time that the virtual partition is blocked to wait for the resolution of its memory page fault. The time is measured in seconds, with millisecond granularity.

Virtualized partition memory page faults
The total number of virtual partition memory page faults that are recorded for the virtualized partition.

Number of 4 KB page frames loaned
The number of the 4 KB pages of the memory that is loaned to the hypervisor in the partition.

Percentage of partition memory loaned
The percentage of the memory loaned to the hypervisor in the partition.

-w Displays the report in wide mode.

Notes:

1. If Active Memory Expansion is enabled, then the vmstat reports memory statistics in the expanded view. However, if the environment variable AME_MEMVIEW is set to TRUE, then the memory statistics will represent the true view.
2. The AME_MEMVIEW environment variable has no impact on memory statistics reported using the **-c** option.

Examples

1. To display a summary of the statistics since boot, enter the following command:
   ```
   vmstat
   ```
2. To display five summaries at 2-second intervals, enter the following command:
   ```
   vmstat 2 5
   ```
3. To display a summary of the statistics since boot including statistics for logical disks scdisk13 and scdisk14, enter the following command:
   ```
   vmstat scdisk13 scdisk14
   ```
4. To display fork statistics, enter the following command:
   ```
   vmstat -f
   ```
5. To display the count of various events, enter the following command:
   ```
   vmstat -s
   ```
6. To display time-stamp next to each column of output of **vmstat**, enter the following command:
   ```
   vmstat -t
   ```
7. To display the I/O oriented view with an alternative set of columns, enter the following command:
   ```
   vmstat -I
   ```
8. To display all the VMM statistics available, enter the following command:
   ```
   vmstat -vs
   ```
9. To display the large-page section with the `alp` and `flp` columns at 8-second intervals, enter the following command:

```
vmstat -l 8
```

10. To display the VMM statistics specific to a particular page size (in the example, 4 KB), enter the following command:

    ```
    vmstat -p 4K
    ```

11. To display the VMM statistics for all page sizes that are supported on the system, enter the following command:

    ```
    vmstat -p ALL
    ```

    Or enter the following command:

    ```
    vmstat -p all
    ```

12. To display only the VMM statistics for a particular page size (in this example, 4 KB), enter the following command:

    ```
    vmstat -P 4K
    ```

13. To display only the per-page breakdown of VMM statistics for all supported page sizes, enter the following command:

    ```
    vmstat -P ALL
    ```

    Or enter the following command:

    ```
    vmstat -P all
    ```

14. To display a summary of the statistics for all of the workload partitions after boot, enter the following command:

    ```
    vmstat -@ ALL
    ```

15. To display all of the VMM statistics available for all of the workload partitions, enter the following command:

    ```
    vmstat –vs -@ ALL
    ```

16. To display both WPAR and system-wide VMM statistics from a workload partition, enter the following command:

    ```
    vmstat -@
    ```

17. To multiply the processor values with 10 and display the results, enter the following command:

    ```
    vmstat -S 1
    ```

18. To display the statistics for the hypervisor page, enter the following command:

    ```
    vmstat –h
    ```

19. To display the information about pages that are loaned to the hypervisor, enter the following command:

    ```
    vmstat -vh
    ```

20. To display memory compression statistics (in an LPAR with Active Memory Expansion enabled), enter the following command:

    ```
    vmstat -c
    ```

21. To display memory compression statistics specific to per-pagesize (in an LPAR with Active Memory Expansion enabled), enter the following command:

    ```
    vmstat –c –P ALL
    ```

22. To append memory compression information to the statistics displayed by **-s** option (in an LPAR with Active Memory Expansion enabled), enter the following command:

    ```
    vmstat –s -c
    ```

23. To append memory compression information to the statistics displayed by **-v** option (in an LPAR with Active Memory Expansion enabled), enter the following command:

    ```
    vmstat –v -c
    ```

Files

/usr/bin/vmstat Contains the **vmstat** command.

Related Information

The **iostat** and **vmo** command.

Memory performance in *Performance management*.

---

vpdadd Command

Purpose

Adds entries to the product, lpp, history, and vendor databases.

Syntax

vpdadd { **-c** *Component* | **-p** *Product* | **-f** *Feature* } **-v** *v.r.m.f* [**-D** *Destdir*] [**-U** *Command*] [**-R** *Prereq*] [**-S** *Msg_Set*] [**-M** *Msg_Number*] [**-C** *Msg_Catalog*] [**-P** *Parent*] [**-I** *Description*]

Description

The **vpdadd** command is for use with or by installers that wish to be listed in Vital Product Database (VPD). The VPD consists of the product, lpp, and history databases. Entries to the inventory database must be added by the **sysck** command. A new vendor database is now included to track products that use destination directories and **non-installp** uninstallers.

The **vpdadd** command uses a tree structure of *Product* at the highest level, then *Feature*, and then *Component*.

The *Component* is the lowest installable unit, but in this hierarchy, a *Component* is not selectable for install or uninstall. Therefore, if an installer is using the **vpdadd** command to update the install database, they should look at their own tree representation and add entries based on their structure. If only adding one entry per install, then adding a *Product* type rather than *Component* type would allow that entry to be listed in the uninstall Web-based System Manager and SMIT interfaces. All the entries are made in the VPD, but *Components* and *Features* are filtered out in the default **lslpp** listings (**-Lc**).

Flags

-C *Msg_Catalog* Specifies the message catalog to search for a translated description of the *Component*. The default (English) description is specified with the **-I** flag. If the message catalog is not in the standard NLSPATH, then the full path name should be given.

-c *Component* Specifies the *Component* name to add to the VPD. An entry is only added if it is unique. Uniqueness is described as having a different destination directory. If the same instance of a *Component* is already in the database, then no entry is added, and an error is returned. This allows a force install (that is, reinstall).

-D *Destdir* Specifies the root (prefix) path that is added to all the files in a *Component* when being installed (and when being added to the inventory database by the **sysck** command). Files in a *Component* are listed with relative path names, so the root path is allowed to change. The default destination directory is **/opt**.

-f *Feature* Specifies the *Feature* name to add to the VPD. An entry is only added if it is unique. Uniqueness is described as having a different VRMF or destination directory. If the same instance of a *Feature* is already in the database, then no entry is added, and an error is not returned. This allows for a force install (that is, reinstall).

-I *Description*	Specifies the default description of the *Component*, *Feature* or *Product*. The description must be specified in double quotation marks. Single quotation marks are allowed inside the description, and double quotation marks must be prepended with a ****.
-M *Msg_Number*	Specifies the message number for the description.
-P *Parent*	Specifies the parent software unit. A *Component* specifies either a *Feature* or a *Product* as its parent, depending on where it was in the tree. This flag is used to allow tree listings in Web-based System Manager.
-p *Product*	Specifies the *Product* name to add to the VPD. An entry is only added if it is unique. Uniqueness is described as having a different VRMF or destination directory. If the same instance of a *Product* is already in the database, then no entry is added, and an error is not returned. This allows a force install (that is, reinstall).
-R *Prereq*	Specifies a *Component* (fileset) that is a requisite of the installing *Component*. The argument must be specified in quotation marks. This flag can be used more than once to specify multiple prerequisites. Although these are treated as prerequisites at install time (by the installer), they are listed as corequisites in the *Product* database to avoid creating circular requisite chains.
-S *Msg_Set*	Specifies the message set (if more than one in the catalog).
-U *Command*	Specifies the *Command* to launch the uninstaller for this *Component*. This may be just a command path name, or it may include parameters if there is a global uninstaller. The **geninstall** command calls this uninstaller, and **installp** does not deinstall a fileset if this value is set in the VPD.
-v *v.r.m.f*	The VRMF of the *Component*, *Feature* or *Product* being added.

Security

RBAC Disclaimer

Examples

1. The following example shows how the Registry service would call **vpdadd** to add a *Component* for the *Foo product*. This *Component* has two requisites, one that is specific to the operating system, and one that is listed as GUID.

   ```
   vpdadd -c EPL2890198489F -v 1.2.3.0 -R "bos.rte.odm 4.3.3.0" -R "8KDEOKY90245686 1.1.0.0"  \
   -U /usr/opt/foo/uninstaller.class -p KID892KYLIE25 -I "Foo Database Component"
   ```

2. To add a new product `devices.pci.cool.rte` to the VPD, enter:

   ```
   vpdadd -p devices.pci.cool.rte -v 5.1.0.0 -U /usr/sbin/udisetup
   ```

Files

/usr/sbin/vpdadd

Related Information

The **installp** command, **lslpp** command, **vpddel** command, and **geninstall** command.

RBAC in *Security*.

Trusted AIX in *Security*.

---

vpddel Command

Purpose

Removes entries from the product, lpp, history, and vendor databases.

Syntax

vpddel { **-c** *Component* | **-p** *Product* | **-f** *Feature* } **-v** *v.r.m.f* **-D** *Dest_dir*

Description

The **vpddel** command removes entries from the product, lpp, history, and vendor databases. The vrmf and destination directory must be specified so that the correct entries are removed.

Flags

-c *Component*	Removes the specified *Component*. The VRMF must also be included when removing a *Component*.
-D *Dest_dir*	Specifies the destination directory of the *Component* to remove. If a destination directory is not included, then the default **/opt** is used.
-f *Feature*	Specifies the *Feature* to remove from the vendor database.
-p *Product*	The *Product* to remove from the vendor database.
-v *V.R.M.F*	Specifies the version, release, modification and fix level of the *component* to delete from the VPD and vendor database.

Security

RBAC Disclaimer

Example

To remove the *Component* `EPL2890198489F` from the product, history, lpp, and vendor databases, type:

```
vpddel -c EPL2890198489F -v 1.2.3.0 -D /usr/lpp/Foo
```

Files

/usr/sbin/vpddel

Related Information

The vpdadd command and lslpp command.

RBAC in *Security*.

Trusted AIX in *Security*.

---

vsdatalst Command

Purpose

vsdatalst – Displays virtual shared disk subsystem information.

Syntax

vsdatalst {**–g** | **–n** | **–v** | **–c**}

Description

Use this command to display one of several kinds of information to standard output.

You can use the System Management Interface Tool (SMIT) to run the **vsdatalst** command. To use SMIT, enter:

```
smit list_vsd
```

and select the option for the kind of virtual shared disk SDR information you want to see.

Flags

Only one of the following flags can be specified with each invocation of **vsdatalst**:

–g Displays the following global volume group data:
global_group_name,
local_group_name,
primary_server_node,
secondary_server_node. (This is only enabled with the Recoverable virtual shared disk subsystem.)
eio_recovery
recovery
CVSD *server_list*

–n Displays the following Node data:
node_number,
host_name,
adapter_name,
min_buddy_buffer_size,
max_buddy_buffer_size,
max_buddy_buffers.

–v Displays the following definition data:
vsd_name,
logical_volume_name,
global_group_name,
minor_number.

–c Displays the following cluster information:
node_number
cluster_name

Parameters

None.

Security

You must have root authority to run this command.

Exit Status

0 Indicates the successful completion of the command.

nonzero Indicates that an error occurred.

Restrictions

You must issue this command from a node that is online in the peer domain. To bring a peer domain online, use the **startrpdomain** command. To bring a particular node online in an existing peer domain, use the **startrpnode** command. For more information on creating and administering an RSCT peer domain, refer to the *RSCT: Administration Guide*.

Standard Output

Current RVSD subsystem run level.

Examples

1. To display global volume group date, enter:

```
vsdatalst -g
```

The system displays a message similar to the following:

```
VSD Global Volume Group Information
                                                Server Node Numbers
Global Volume Group name   Local VG name   primary   backup eio_recovery  recovery  server_list  vsd_type
------------------------    ------------   -------   ------ ------------  --------  -----------  --------
gpfs0gvg                     gpfs0vg         1         2          0          0        0            VSD
gpfs1gvg                     gpfs1vg         2         1          0          0        0            VSD
gpfs3gvg                     gpfs3vg         1         0          0          0        1:2          CVSD
```

2. To display global volume group date, enter:

```
vsdatalst -n
```

The system displays a message similar to the following:

```
VSD Node Information
                                              Buddy Buffer
  node                 VSD      IP packet minimum maximum # maxbufs
number host_name       adapter    size     size    size
------ --------------- -------- --------- ------- ------- ---------
     1 host1            ml0       61440    4096   262144    128
     2 host2            ml0       61440    4096   262144    128
```

3. To display global volume group date, enter:

```
vsdatalst -v
```

The system displays a message similar to the following:

```
VSD Table
VSD name                       logical volume  Global Volume Group             minor# size_in_MB
------------------------------ --------------- ------------------------------- ------ ----------
gpfs0vsd                       gpfs0lv         gpfs0gvg                             3 4096
gpfs1vsd                       gpfs1lv         gpfs1gvg                             1 4096
gpfs3vsd                       gpfs3lv         gpfs3gvg                             4 4096
```

Location

/opt/rsct/vsd/bin/vsdatalst

Related Information

Commands: **lsvsd**, **updatevsdnode**, **vsdnode**

---

vsdchgserver Command

Purpose

vsdchgserver – Switches the server function for one or more virtual shared disks from the node that is currently acting as the server node to the other.

Syntax

vsdchgserver **–g** *vsd_global_volume_group_name* **–p** *primary_node*
[**–b** *secondary_node*] [**–o** *EIO_recovery*]

Description

The **vsdchgserver** command allows the serving function for a global volume group defined on a primary node to be taken over by the secondary node, or to be taken over by the primary node from the secondary node. This allows an application to continue to use virtual shared disks in situations where the cable or adapter between the physical disks and one of the attached nodes is not working.

The Recoverable virtual shared disk subsystem automatically updates the virtual shared disk devices if, and only if, the **vsdchgserver** command is used to flip the currently-defined primary node and secondary node in the global volume group specified in the **–g** flag.

Flags

–g Specifies the Global Volume Group name for the volume group that represents all the virtual shared disks defined on a particular node.

–p Specifies the primary server node number for the global volume group.

–b Specifies the secondary node number for the global volume group. If the **-b** flag is not specified, the secondary node definition will be removed.

–o Specified as **0**, for no recovery on an EIO error, or **1**, for recovery on an EIO error.

Parameters

None.

Security

You must have root authority to run this command.

Exit Status

0 Indicates the successful completion of the command.

nonzero Indicates that an error occurred.

Restrictions

You must issue this command from a node that is online in the peer domain. To bring a peer domain online, use the **startrpdomain** command. To bring a particular node online in an existing peer domain, use the **startrpnode** command. For more information on creating and administering an RSCT peer domain, refer to the *RSCT: Administration Guide*.

Standard Output

Current RVSD subsystem run level.

Examples

To change the primary server node for the global volume group node12vg to node 1 and the secondary node to node 2, with EIO recovery, enter:

```
vsdchgserver -g node12vg -p 1 -b 2 -o 1
```

Location

/opt/rsct/vsd/bin/vsdchgserver

Related Information

Refer to *RSCT: Managing Shared Disks* for information on how to use this command in writing applications.

---

vsdelnode Command

Purpose

Removes virtual shared disk information for a node or series of nodes.

Syntax

vsdelnode *node_number* ...

Description

This command is used to remove virtual shared disk data for a node or series of nodes.

The **vsdelnode** command makes the listed nodes no longer virtual shared disk nodes so that no virtual shared disks can be accessed from them. This command is unsuccessful for any nodes that are servers for any global volume groups.

You can use the System Management Interface Tool (SMIT) to run the **vsdelnode** command. To use SMIT, enter:

```
smit delete_vsd
```

and select the **Delete Virtual Shared Disk Node** Information option.

Flags

–g Specifies the Global Volume Group name for the volume group that represents all the virtual shared disks defined on a particular node.

–p Specifies the primary server node number for the global volume group.

–b Specifies the secondary node number for the global volume group. If the **-b** flag is not specified, the secondary node definition will be removed.

–o Specified as **0**, for no recovery on an EIO error, or **1**, for recovery on an EIO error.

Parameters

node_number Specifies the node number of the node whose virtual shared disk information you want to remove.

Security

You must have **root** authority to run this command.

Restrictions

The recoverable virtual shared disk subsystem must be stopped on the node(s) you are deleting. Otherwise, the results may be unpredictable. For more information, see *RSCT for AIX 5L Managing Shared Disks* .

You must issue this command from a node that is online in the peer domain. To bring a peer domain online, use the **startrpdomain** command. To bring a particular node online in an existing peer domain, use the **startrpnode** command. For more information on creating and administering an RSCT peer domain, refer to *RSCT Administration Guide*.

Examples

To delete virtual shared disk node information for nodes **3** and **6**, enter:

```
vsdelnode 3 6
```

Location

/opt/rsct/vsd/bin/vsdelnode

Related Information

Commands: **vsdatalst**, **vsdnode**

vsdelvg Command

Purpose

vsdelvg – Removes virtual shared disk global volume group information.

Syntax

vsdelvg [**–f**] *global_group_name* ...

Description

Use this command to remove virtual shared disk global volume group information. If any virtual shared disks are defined on a global volume group, the **vsdelvg** command is unsuccessful unless **–f** is specified. If **–f** is specified, any such virtual shared disks must be unconfigured and in the defined state on all the virtual shared disk nodes to be deleted.

You can use the System Management Interface Tool (SMIT) to run the **vsdelvg** command. To use SMIT, enter:

```
smit delete_vsd
```

and select the **Delete Virtual Shared Disk Global Volume Group Information** option.

Flags

–f Forces the removal of any virtual shared disks defined on this global volume group.

Parameters

global_group_name
Specifies the volume group that you no longer want to be global to the system.

Security

You must have root authority to run this command.

Exit Status

0 Indicates the successful completion of the command.

nonzero Indicates that an error occurred.

Restrictions

You must issue this command from a node that is online in the peer domain. To bring a peer domain online, use the **startrpdomain** command. To bring a particular node online in an existing peer domain, use the **startrpnode** command. For more information on creating and administering an RSCT peer domain, refer to the *RSCT: Administration Guide*.

Standard Output

Current RVSD subsystem run level.

Examples

To remove the global volume group **vg1n1**, enter:

```
vsdelvg vg1n1
```

Location

/opt/rsct/vsd/bin/vsdelvg

Related Information

Commands: **undefvsd**, **vsdatalst**, **vsdvg**

---

vsdnode Command

Purpose

Define virtual shared disk information for a node or series of nodes.

Syntax

vsdnode *node_number... adapter_name min_buddy_buffer_size*
max_buddy_buffer_size max_buddy_buffers
vsd_max_ip_msg_size [*cluster_name*]

Description

Use this command to make the specified nodes virtual shared disk nodes and to assign their virtual shared disk operational parameters. If this information is the same for all nodes, run this command once. If the information is different for the nodes, run this command once for each block of nodes that should have the same virtual shared disk information.

You can use the System Management Interface Tool (SMIT) to run the **vsdnode** command. To use SMIT, enter:

```
smit vsd_data
```

and select the **virtual shared disk Node Information** option.

Flags

–f Forces the removal of any virtual shared disks defined on this global volume group.

Parameters

node_number Specifies the node or nodes whose virtual shared disk information is to be set. The value you specify for *node_number* must match a valid RSCT remote peer domain node number.

adapter_name Specifies the adapter name to be used for virtual shared disk communications for the nodes specified. The adapter name must already be defined to the nodes. Note that the nodes involved in virtual shared disk support must be fully connected so that proper communications can take place. Use **ml0** to specify that the virtual shared disk device driver transmits data requests over the SP Switch. The **ml0** adapter will be used the next time the virtual shared disk device driver is loaded.

min_buddy_buffer_size
Specifies the smallest buddy buffer a server uses to satisfy a remote request to a virtual shared disk. This value must be a power of 2 and greater than or equal to 4096. The suggested value is 4096 (4 KB). For a 512 byte request, 4 KB is excessive. However, recall that a buddy buffer is only used for the short period of time while a remote request is being processed at the server node.

max_buddy_buffer_size
Specifies the largest buddy buffer a server uses to satisfy a remote noncached request. This value must be a power of 2 and greater than or equal to the *min_buddy_buffer_size*. The suggested value is 262144 (256 KB). This value depends on the I/O request size of applications using the virtual shared disks and the network used by the virtual shared disk software.

max_buddy_buffers
Specifies the number of *max_buddy_buffer_size* buffers to allocate. The virtual shared disk device driver will have an initial size when first loaded, and then will dynamically allocate and reclaim additional space as needed. The suggested value is 2000 256 KB buffers.

Buddy buffers are only used on the servers. On client nodes you may want to set *max_buddy_buffers* to 1.

Note: The **statvsd** command will indicate if remote requests are queueing waiting for buddy buffers.

vsd_max_ip_msg_size
Specifies the maximum message size in bytes for virtual shared disks. This value must not be greater than the maximum transmission unit (MTU) size of the network. The recommended values are:

- 61440 (60KB) for a switch
- 8192 (8KB) for jumbo frame Ethernet
- 1024 (1KB) for 1500-byte MTU Ethernet

cluster_name A cluster name must be specified for server nodes that will be serving concurrently accessed shared disks. The cluster name can be any user provided name. A node can only belong to one cluster. For example, when you have a concurrent access environment, the two servers for the CVSD must both specify the same cluster name.

Note: The *cluster_name* is required only for SSA (Serial Storage Architecture) disks.

Security
You must have **root** authority to run this command.

Restrictions
The node specified on this command must already belong to a peer domain, and you must issue this command from a node that is online in the peer domain. To bring a peer domain online, use the **startrpdomain** command. To bring a particular node online in an existing peer domain, use the **startrpnode** command. For more information on creating and administering an RSCT peer domain, refer to *RSCT Administration Guide*.

Examples
The following example defines information for a switch network and nodes 1 through 8.

```
vsdnode 1 2 3 4 5 6 7 8 ml0 4096 262144 128 61440
```

Location
/opt/rsct/vsd/bin/vsdnode

Related Information
Commands: **updatevsdnode**, **vsdatalst**, **vsdelnode**

vsdsklst Command

Purpose

Produces output that shows you the disk resources used by the virtual shared disk subsystem across a peer domain.

Syntax

vsdsklst [**–v**] [**–d**] {**–a** | **–n** *node_number*[, node_number2, ...]}

Description

Use this command to check disk utilization across a peer domain.

Flags

–v Displays only disk utilization information about volume groups and the virtual shared disks associated with them.

–d Displays only disk utilization information about volume groups and the physical disks associated with them.

–a Displays specified information for all nodes in the system or system partition.

–n *node_number*
Lists one or more node numbers for which information is to be displayed.

Parameters

node_number Specifies the node or nodes whose virtual shared disk information is to be set. The value you specify for *node_number* must match a valid RSCT remote peer domain node number.

adapter_name Specifies the adapter name to be used for virtual shared disk communications for the nodes specified. The adapter name must already be defined to the nodes. Note that the nodes involved in virtual shared disk support must be fully connected so that proper communications can take place. Use **ml0** to specify that the virtual shared disk device driver transmits data requests over the SP Switch. The **ml0** adapter will be used the next time the virtual shared disk device driver is loaded.

min_buddy_buffer_size
Specifies the smallest buddy buffer a server uses to satisfy a remote request to a virtual shared disk. This value must be a power of 2 and greater than or equal to 4096. The suggested value is 4096 (4 KB). For a 512 byte request, 4KB is excessive. However, recall that a buddy buffer is only used for the short period of time while a remote request is being processed at the server node.

max_buddy_buffer_size
Specifies the largest buddy buffer a server uses to satisfy a remote noncached request. This value must be a power of 2 and greater than or equal to the *min_buddy_buffer_size*. The suggested value is 262144 (256 KB). This value depends on the I/O request size of applications using the virtual shared disks and the network used by the virtual shared disk software.

max_buddy_buffers
Specifies the number of *max_buddy_buffer_size* buffers to allocate. The virtual shared disk device driver will have an initial size when first loaded, and then will dynamically allocate and reclaim additional space as needed. The suggested value is 2000 256KB buffers.

Buddy buffers are only used on the servers. On client nodes you may want to set *max_buddy_buffers* to 1.

Note: The **statvsd** command will indicate if remote requests are queueing waiting for buddy buffers.

vsd_max_ip_msg_size
Specifies the maximum message size in bytes for virtual shared disks. This value must not be greater than the maximum transmission unit (MTU) size of the network. The recommended values are:

- 61440 (60KB) for a switch
- 8192 (8KB) for jumbo frame Ethernet
- 1024 (1KB) for 1500-byte MTU Ethernet

cluster_name A cluster name must be specified for server nodes that will be serving concurrently accessed shared disks. The cluster name can be any user provided name. A node can only belong to one cluster. For example, when you have a concurrent access environment, the two servers for the CVSD must both specify the same cluster name.

Note: The *cluster_name* is required only for SSA (Serial Storage Architecture) disks.

Security

You must have **root** authority to run this command.

Restrictions

You must issue this command from a node that is online in the peer domain. To bring a peer domain online, use the **startrpdomain** command. To bring a particular node online in an existing peer domain, use the **startrpnode** command. For more information on creating and administering an RSCT peer domain, refer to *RSCT Administration Guide*.

Examples

This command:

```
vsdsklst -dv -a
```

displays the following information on a system that has volume groups and virtual shared disks defined on nodes 1 and 2.

```
c164n12.ppd.pok.ibm.com: Node Number:2; Node Name:c164n12.ppd.pok.ibm.com
c164n12.ppd.pok.ibm.com: Volume group:rootvg; Partition Size:32; Total:271; Free:168
c164n12.ppd.pok.ibm.com:         Physical Disk:hdisk0; Total:271; Free:168
c164n12.ppd.pok.ibm.com:     Volume group:testvg is not varied on.
c164n12.ppd.pok.ibm.com:         Physical Disk:hdisk5;
c164n12.ppd.pok.ibm.com: Volume group:test1vg; Partition Size:4; Total:537; Free:534
c164n12.ppd.pok.ibm.com:         Physical Disk:hdisk2; Total:537; Free:534
c164n12.ppd.pok.ibm.com:         VSD Name:vsd1n2[testnewlv21n2]; Size:1
c164n12.ppd.pok.ibm.com:         VSD Name:vsd2n2[testlv1n2]; Size:346112.25
c164n12.ppd.pok.ibm.com:         VSD Name:vsd3n2[testlv2n2]; Size:346112.25
c164n12.ppd.pok.ibm.com:     Volume group:vg1 is not varied on.
c164n12.ppd.pok.ibm.com:         Physical Disk:hdisk9;
c164n12.ppd.pok.ibm.com:     Volume group:sharkvg is not varied on.
c164n12.ppd.pok.ibm.com:         Physical Disk:hdisk7;
c164n12.ppd.pok.ibm.com:         Physical Disk:hdisk10;
c164n12.ppd.pok.ibm.com: Volume group:bdhclvg; Partition Size:32; Total:134; Free:102
c164n12.ppd.pok.ibm.com:         Physical Disk:hdisk13; Total:134; Free:102
c164n12.ppd.pok.ibm.com: Volume group:gpfs0vg; Partition Size:8; Total:536; Free:0
c164n12.ppd.pok.ibm.com:         Physical Disk:hdisk12; Total:536; Free:0
c164n12.ppd.pok.ibm.com:         VSD Name:gpfs0vsd[gpfs0lv]; Size:352256.75
c164n12.ppd.pok.ibm.com:     Not allocated physical disks:
c164n12.ppd.pok.ibm.com:         Physical disk:hdisk1
```

```
c164n12.ppd.pok.ibm.com:         Physical disk:hdisk3
c164n12.ppd.pok.ibm.com:         Physical disk:hdisk4
c164n12.ppd.pok.ibm.com:         Physical disk:hdisk6
c164n12.ppd.pok.ibm.com:         Physical disk:hdisk11
c164n12.ppd.pok.ibm.com:         Physical disk:hdisk15
c164n11.ppd.pok.ibm.com: Node Number:1; Node Name:c164n11.ppd.pok.ibm.com
c164n11.ppd.pok.ibm.com: Volume group:rootvg; Partition Size:32; Total:271; Free:172
c164n11.ppd.pok.ibm.com:         Physical Disk:hdisk0; Total:271; Free:172
c164n11.ppd.pok.ibm.com: Volume group:bdhclvg; Partition Size:32; Total:134; Free:102
c164n11.ppd.pok.ibm.com:         Physical Disk:hdisk9; Total:134; Free:102
c164n11.ppd.pok.ibm.com:         VSD Name:bdhcvsd1n1[lvbdhcvsd1n1]; Size:45056
c164n11.ppd.pok.ibm.com: Volume group:testvg; Partition Size:16; Total:134; Free:70
c164n11.ppd.pok.ibm.com:         Physical Disk:hdisk13; Total:134; Free:70
c164n11.ppd.pok.ibm.com:     Not allocated physical disks:
c164n11.ppd.pok.ibm.com:         Physical disk:hdisk1
c164n11.ppd.pok.ibm.com:         Physical disk:hdisk2
c164n11.ppd.pok.ibm.com:         Physical disk:hdisk3
```

Location

/opt/rsct/vsd/bin/vsdsklst

Related Information

Commands: **vsdatalst**

---

vsdvg Command

Purpose

Defines a virtual shared disk global volume group.

Syntax

vsdvg [**–g** *global_volume_group*] {**–l** *server_list local_group_name* | *local_group_name primary_node* [*secondary node* [*eio_recovery*]]}

Description

Use this command to define volume groups for use by the Virtual shared disk subsystem. This is done by specifying the local volume group name, the node on which it resides, and the name by which the volume group will be known throughout the cluster.

You can use the System Management Interface Tool (SMIT) to run the **vsdvg** command. To use SMIT, enter the following command and select the **Virtual Shared Disk Global Volume Group Information** option:

```
smit vsd_data
```

Flags

–g *global_volume_group*
: Specifies a unique name for the new global volume group. This name must be unique across the system partition. It should be unique across the SP, to avoid any naming conflicts during future system partitioning operations. The suggested naming convention is **vg***xx***n***yy*, where *yy* is the node number, and *xx* uniquely numbers the volume groups on that node. If this is not specified, the local group name is used for the global name. The length of the name must be less than or equal to 31 characters.

–l *server_list*
: Define the list of servers for CVSD. More than one server indicates the *global_volume_group* is a concurrent volume group.

Parameters

local_group_name
Specifies the name of a volume group that you want to indicate as being used for virtual shared disks. This name is local to the host upon which it resides. The length of the name must be less than or equal to 15 characters.

primary_node Specifies the primary server node number on which the volume group resides. The length of the name must be less than or equal to 31 characters.

secondary_node
Specifies the secondary server node number on which the volume group resides. The length of the name must be less than or equal to 31 characters.

eio_recovery

Specifies how the Recoverable virtual shared disk subsystem will respond to EIO errors. If *eio_recovery* is set to the value 1 (the default), an EIO error will cause the Recoverable virtual shared disk system to flip the current primary node and the secondary node and perform one more retry on the new primary node.

Security

You must have root authority to run this command.

Exit Status

0 Indicates the successful completion of the command.

nonzero Indicates that an error occurred.

Restrictions

You must issue this command from a node that is online in the peer domain. To bring a peer domain online, use the **startrpdomain** command. To bring a particular node online in an existing peer domain, use the **startrpnode** command. For more information on creating and administering an RSCT peer domain, refer to the *RSCT: Administration Guide*.

Standard Output

Current RVSD subsystem run level.

Examples

1. The following example defines **gpfs1gvg** as a virtual shared disk global volume group with the local volume group **gpfs1vg** accessed from **node1** as the *primary_node* and **node2** as the *secondary_node*.

   ```
   vsdvg -g gpfs1gvg gpfs1vg 1 2
   ```

2. The following example defines **gpfs3gvg** as a virtual shared disk global volume group with the local volume group **gpfs3vg** concurrently accessed from **node1** and **node2**.

   ```
   vsdvg -g gpfs3gvg -l 1:2 gpfs3vg
   ```

Location

/opt/rsct/vsd/bin/vsdvg

Related Information

Commands: **updatevsdvg**, **vsdelvg**

vsdvgts Command

Purpose

vsdvgts – Updates the timestamp used by the Recoverable virtual shared disk subsystem by reading the timestamp from the volume group descriptor area (VGDA) of the physical disks.

Syntax

vsdvgts [**–a**] [*volgrp*]

Description

Use this command to update the timestamp that the Recoverable virtual shared disk subsystem uses to determine if a twin-tailed volume group has changed. When the subsystem detects a change, the recovery scripts export the volume group and then import the volume group.

This command can be used to avoid exporting the volume group and then importing the volume group during recovery in situations where the export and import operations are not really necessary. This command should be used very carefully.

Flags

–a Specifies that the timestamps for this volume group for both primary and secondary nodes should be updated. If this flag is not specified, the timestamp is updated on the local node only.

Parameters

volgrp Specifies a volume group. If this operand is not specified, the timestamps for all the volume groups on this node are updated.

Security

You must have root authority to run this command.

Exit Status

0 Indicates the successful completion of the command.

1 Indicates that the program was unable to read one or more timestamps.

Restrictions

You must issue this command from a node that is online in the peer domain. To bring a peer domain online, use the **startrpdomain** command. To bring a particular node online in an existing peer domain, use the **startrpnode** command. For more information on creating and administering an RSCT peer domain, refer to the *RSCT: Administration Guide*.

Standard Output

Current RVSD subsystem run level.

Examples

To update the timestamp associated with the virtual shared disk volume group vsdvg1 for just this node, enter:

```
vsdvgts vsdvg1
```

Location

/usr/lpp/vsd/bin/vsdvgts

Related Information

Commands: **updatevsdvg**, **vsdelvg**

---

w Command

Purpose

Prints a summary of current system activity.

Syntax

w [**-h**] [**-u**] [**-w**] [**-l** | **-s** [**-X**] [**-@** [*WPAR*]][*User*]

Description

The **w** command prints a summary of the current activity on the system. The summary includes the following:

WPAR	Workload partition name.
User	Who is logged on.
tty	Name of the tty the user is on.
login@	Time of day the user logged on.
idle	Number of minutes since a program last attempted to read from the terminal.
JCPU	System unit time used by all processes and their children on that terminal.
PCPU	System unit time used by the currently active process.
What	Name and arguments of the current process.

The heading line of the summary shows the current time of day, how long the system has been up, the number of users logged into the system, and the load average. The load average is the number of runnable processes over the preceding 1-, 5-, 15-minute intervals.

The following examples show the different formats used for the login time field:

`10:25am`	The user logged in within the last 24 hours.
`Tue10am`	The user logged in between 24 hours and 7 days.
`12Mar91`	The user logged in more than 7 days ago.

If a user name is specified with the *User* parameter, the output is restricted to that user.

Flags

-@ Prints the System activity tagged with a workload partition name:

- providing the **-@** option without a WPAR name indicates the global environment in addition to all WPARs active in the system, and the heading line indicates values for the global environment only
- providing the **-@** option with a WPAR name indicates the activity, and the heading line indicates values for only that WPAR
- providing **-@** Global indicates the activity, and the heading line indicates values for the global environment only.

Note: Not providing the **-@** option indicates that the current WPAR or global environment, wherever the **w** command is running.

-h Suppresses the heading.

-l Prints the summary in long form. This is the default.

-s Prints the summary in short form. In the short form, the tty is abbreviated, and the login time, system unit time, and command arguments are omitted.

-u Prints the time of day, amount of time since last system startup, number of users logged on, and number of processes running. This is the default. Specifying the **-u** flag without specifying the **-w** or **-h** flag is equivalent to the **uptime** command.

-w The equivalent of specifying the **-u** and **-l** flags, which is the default.

-X Prints all available characters of each user name instead of truncating to the first 8 characters. The user name is also moved to the last column of the output.

Files

/etc/utmp Contains the list of users.

Related Information

The **who** command, **finger** command, **ps** command, **uptime** command.

---

wait Command

Purpose

Waits until the termination of a process ID.

Syntax

wait [*ProcessID* ...]

Description

The **wait** command waits (pauses execution) until the process ID specified by the *ProcessID* variable terminates. If the *ProcessID* variable is not specified, the **wait** command waits until all process IDs known to the invoking shell have terminated and exit with a 0 exit status. If a *ProcessID* variable represents an unknown process ID, the **wait** command treats them as known process IDs that exited with exit status 127. The **wait** command exits with the exitstatus of the last process ID specified by the *ProcessID* variable.

Flag

ProcessID Specifies an unsigned decimal integer process ID of a command, which the **wait** command waits on until termination.

Exit Status

If one or more operands were specified, all of the operands terminated or were not known by the invoking shell, and the status of the last operand specified is known, then the exit status of the **wait** command is the same as the exit status information of the command indicated by the last operand specified. If the process terminated abnormally due to the receipt of a signal, then the exit status is greater than 128 and distinct from the exit status information generated by other signals, although the exact status value is unspecified (see the **kill -l** command option). Otherwise, the **wait** command exits with one of the following values:

0 The **wait** command was invoked with no operands and all process IDs known by the invoking shell have terminated.

1-126 The **wait** command detected an error.

127 The command identified by the last *ProcessID* operand specified is unknown.

File

/usr/bin/wait Contains the **wait** command.

Related Information

The **shutdown** command, **sleep** command, **wall** command.

The **alarm** subroutine, **pause** subroutine, **sigaction** subroutine.

Shells in *Operating system and device management*.

---

wall Command

Purpose

Writes a message to all users that are logged in.

Syntax

wall [**-a**] [**-g** *Group*][*Message*]

Description

The **wall** command writes a message to all users that are logged in. If the *Message* parameter is not specified, the **wall** command reads the message from standard input until it reaches an end-of-file character. The message is then sent to all logged in users. The following heading precedes the message:

```
Broadcast message from
user@node

 (tty) at hh:mm:ss ...
```

`hh:mm:ss` represents the hours, minutes, and seconds when the message was sent.

To override any protections set up by other users, you must operate with root user authority. Typically, the root user uses the **wall** command to warn all other users of an impending system shutdown.

Note:

- The **wall** command only sends messages to the local node.
- Messages can contain multibyte characters.

Flags

-a	Performs the default operation. This flag is provided for System V compatibility. It broadcast messages to the console and pseudo-terminals.
-g *Group*	Broadcasts to a specified group only.

Files

/dev/tty Specifies a device.

Related Information

The **mesg** command, **su** command, **write** command.

---

wallevent Command

Purpose

Broadcasts an event or a rearm event to all users who are logged in.

Syntax

wallevent [**-c**] [**-h**]

Description

The **wallevent** script broadcasts a message on an event or a rearm event to all users who are currently logged in to the host when the event or the rearm event occurs. Event or rearm event information is captured and posted by the event response resource manager in environment variables that are generated by the event response resource manager when an event or a rearm event occurs. This script can be used as an action that is run by an event response resource. It can also be used as a template to create other user-defined actions. The language in which the messages of the **wallevent** script are returned depend on the locale settings.

Messages are displayed in this format at the consoles of all users who are logged in when an event or a rearm event occurs for which this script is a response action :

```
Broadcast message from user@host (tty) at hh:mm:ss...

severity event_type occurred for Condition condition_name
on the resource resource_name of resource_class_name at hh:mm:ss mm/dd/yy
The resource was monitored on node_name and resided on {node_names}.
```

Event information is returned about the ERRM environment variables, and also includes the following:

Local Time
: Time when the event or rearm event is observed. The actual environment variable supplied by ERRM is ERRM_TIME. This value is localized and converted to readable form before being displayed.

This script captures the environment variable values and uses the **wall** command to write a message to the currently logged-in user consoles.

Flags

–c Instructs **wallevent** to broadcast the **ERRM_VALUE** of an ERRM event. When the **-c** flag is specified, **wallevent** broadcasts the SNMP trap message.

–h Writes the script's usage statement to standard output.

Parameters

log_file
: Specifies the name of the file where event information is logged. An absolute path for the *log_file* parameter should be specified.

  The *log_file* is treated as a circular log and has a fixed size of 64KB. When *log_file* is full, new entries are written over the oldest existing entries.

  If *log_file* already exists, event information is appended to it. If *log_file* does not exist, it is created so that event information can be written to it.

Exit Status

0 Script has run successfully.

1 Error occurred when the script was run.

Restrictions

1. This script must be run on the node where the ERRM is running.
2. The **wall** command is used to write a message to currently logged-in user consoles. Refer to the **wall** man page for more information on the **wall** command.

Standard Output

When the **-h** flag is specified, the script's usage statement is written to standard output.

Examples

1. Suppose the **wallevent** script is a predefined action in the critical-notification response, which is associated with the **/var space used** condition on the resource **/var**. The threshold of the event expression defined for this condition is met, and an event occurs. The critical-notification response takes place, and **wallevent** is run. The following message is displayed on the consoles of all users who are logged in:

   ```
   Broadcast message from joe@neverland.com (pts/6) at 18:42:03...

   Critical event occurred for Condition /var space used
   on the resource /var of filesys of IBM.FileSystem at 18:41:50 03/28/02
   The resource was monitored on c174n05 and resided on {c174n05}.
   ```

2. When a rearm event occurs for the **/var space used** condition on the resource **/var**, the following message is displayed on the consoles of all users who are logged in:

   ```
   Broadcast message from joe@neverland.com (pts/6) at 18:42:03...

   Critical rearm event occurred for Condition /var space used
   on the resource /var of filesys of IBM.FileSystem at 18:41:50 03/28/02
   The resource was monitored on c174n05 and resided on {c174n05}.
   ```

Location

/usr/sbin/rsct/bin/wallevent

Related Information

Commands: **wall**, **ewallevent**

---

watch Command

Purpose

Observes a program that might be untrustworthy.

Syntax

watch [**-e** *Events*] [**-o** *File*] *Command* [*Parameter* ...]

Description

The **watch** command permits the root user or a member of the audit group to observe the actions of a program that is thought to be untrustworthy. The **watch** command executes the program you specify with the *Command* parameter, with or without any *Parameter* fields, and records all audit events or the audit events you specify with the **-e** flag.

The **watch** command observes all the processes that are created while the program runs, including any child process. The **watch** command continues until all processes exit, including the process it created, to observe all the events that occur.

The **watch** command formats the audit records and writes them to standard output or to a file you specify with the **-o** flag.

For the **watch** command to work, the auditing subsystem must not have been configured and enabled.

Flags

-e *Events*	Specifies the events to be audited. The *Events* parameter is a comma-separated list of audit events that are defined in the **/etc/security/audit/events** file. The default value is all events.
-o *File*	Specifies the path name of the output file. If the **-o** flag is not used, output is written to standard output.

Security

Access Control: This command should grant execute (x) access to the root user and members of the audit group. The command should be **setuid** to the root user so it can access other audit subsystem commands and files, and have the **trusted computing base** attribute.

Files Accessed:

Mode	File
r	**/dev/audit**
x	**/usr/sbin/auditstream**
x	**/usr/sbin/auditselect**
x	**/usr/sbin/auditpr**

RBAC Disclaimer

Examples

1. To watch all files opened by the **bar** command, enter:

   ```
   watch -e FILE_Open /usr/lpp/foo/bar -x
   ```

   This command opens the audit device and executes the **/usr/lpp/foo/bar** command. It then reads all records and selects and formats those with the event type of `FILE_Open`.
2. To watch the installation of the `xyzproduct` program, that might be untrustworthy, enter:

   ```
   watch /usr/sbin/installp xyzproduct
   ```

   This command opens the audit device and executes the **/usr/sbin/installp** command. It then reads all records and formats them.

Files

/usr/sbin/watch	Contains the **watch** command.
/dev/audit	Specifies the audit device from which the audit records are read.

Related Information

The **audit** command, **auditbin** daemon, **auditcat** command, **auditpr** command, **auditselect** command, **auditstream** command, **login** command, **logout** command, **su** command.

The **auditread** subroutine.

For more information about the identification and authentication of users, discretionary access control, the trusted computing base, and auditing, refer to *Security* .

For more information about auditing, refer to Auditing overview in *Security*.

RBAC in *Security*.

Trusted AIX in *Security*.

---

wc Command

Purpose

Counts the number of lines, words, bytes, or characters in a file.

Syntax

wc [**-c** | **-m**] [**-l**] [**-w**] [*File ...*]

wc -k [**-c**] [**-l**] [**-w**] [*File ...*]

Description

By default, the **wc** command counts the number of lines, words, and bytes in the files specified by the *File* parameter. The command writes the number of newline characters, words, and bytes to the standard output and keeps a total count for all named files.

When you use the *File* parameter, the **wc** command displays the file names as well as the requested counts. If you do not specify a file name for the *File* parameter, the **wc** command uses standard input.

The **wc** command is affected by the **LANG**, **LC_ALL**, **LC_CTYPE**, and **LC_MESSAGES** environment variables.

The **wc** command considers a word to be a string of characters of non-zero length which are delimited by a white space (for example SPACE , TAB).

Flags

-c Counts bytes unless the **-k** flag is specified. If the **-k** flag is specified, the **wc** command counts characters.

-k Counts characters. Specifying the **-k** flag is equivalent to specifying the **-klwc** flag. If you use the **-k** flag with other flags, then you must include the **-c** flag. Otherwise, the **-k** flag is ignored. For more information, see examples 4 and 5.

Note: This flag is to be withdrawn in a future release.

-l Counts lines.

-m Counts characters. This flag cannot be used with the **-c** flag.

-w Counts words. A word is defined as a string of characters delimited by spaces, tabs, or newline characters.

Note: If no flag is specified, **wc** by default counts the lines, words, bytes in a file or from standard input.

Exit Status

This command returns the following exit values:

0	The command ran successfully.
>0	An error occurred.

Examples

1. To display the line, word, and byte counts of a file, enter:

   ```
   wc chap1
   ```

   The **wc** command displays the number of lines, words, and bytes in the `chap1` file.
2. To display only byte and word counts, enter:

   ```
   wc -cw chap*
   ```

   The **wc** command displays the number of bytes and words in each file that begins with `chap`. The command also displays the total number of bytes and words in these files.
3. To display the line, word, and character counts of a file, enter:

   ```
   wc -k chap1
   ```

   The **wc** command displays the number of lines, words, and characters in the `chap1` file.
4. To display the word and character counts of a file, enter:

   ```
   wc -kcw chap1
   ```

   The **wc** command displays the number of characters and words in the `chap1` file.
5. To use the **wc** command on standard input, enter:

   ```
   wc -klw
   ```

   The **wc** command displays the number of lines and words in standard input. The **-k** flag is ignored.
6. To display the character counts of a file, enter:

   ```
   wc -m chap1
   ```

   The **wc** command displays the number of characters in the `chap1` file.
7. To use the **wc** command on standard input, enter:

   ```
   wc -mlw
   ```

   The **wc** command displays the number of lines, words, and characters in standard input.

Files

/usr/bin/wc, /bin/wc	Contains the **wc** command.
/usr/ucb/wc	Contains the symbolic link to the **wc** command.

Related Information

Files and Input and output redirection in *Operating system and device management*.

Understanding Locale Environment Variables in *AIX 5L Version 5.3 National Language Support Guide and Reference*.

what Command

Purpose

Displays identifying information in files.

Syntax

what [**-s**] *Pathname/File*.

Description

The **what** command searches specified files for all occurrences of the pattern that the **get** command substitutes for the **@(#)** keyletter (see the **get** or **prs** command for a description of identification keywords). By convention, the value substituted is ″@(#)″ (double quotation marks, at sign, left parenthesis, pound sign, right parenthesis, double quotation marks). If no file is specified, the **what** command reads from standard input.

The **what** command writes to standard output whatever follows the pattern, up to but not including the first double quotation mark (″), greater than symbol (>), new-line character, backslash (\), or null character.

The **what** command should be used in conjunction with the **get** command, which automatically inserts the identifying information. You can also use the **what** command on files where the information is inserted manually.

The **what** command accommodates the compiler inserted command line options in a binary file. The command line options saved in a binary file by AIX compilers may contain the backslash (\), the greater than symbol (>), or the double quotation mark (") within the macro definitions. The **what** command behaves in the following manner to write the command line options saved by the compiler.

At the start of a line, if the pattern @(#) is followed by "opt" and is with or without a blank space (" ") in between, then the **what** command writes the character till '\n' that is the end of a line.

For example,

- For C and FORTRAN AIX compilers, use @(#) opt (...).
- For C++ AIX compiler, use @(#) opt (...).

The whole line is printed after "@(#)".

Note: The **what** command may fail to find SCCS identification strings in executable files.

Flags

-s Searches for only the first occurrence of the **@(#)** pattern.

Exit Status

This command returns the following exit values:

0 Any matches were found.
1 Otherwise.

Examples

Suppose that the file `test.c` contains a C program that includes the line:

```
char ident[ ] = "@(#)Test Program";
```

If you compile `test.c` to produce `test.o`, then the command:

```
what test.c test.o
```

displays:

```
test.c:
Test Program
test.o:
Test Program
```

Note: The full file path names `usr/bin/test.c` and `user/bin/test.o` are required if the files are not in the current directory.

Files

/usr/bin/what	Contains the **what** command.

Related Information

The **get** command, **sccshelp** command.

The **sccsfile** file format.

List of SCCS Commands in *AIX 5L Version 5.3 General Programming Concepts: Writing and Debugging Programs*.

Source Code Control System (SCCS) Overview in *AIX 5L Version 5.3 General Programming Concepts: Writing and Debugging Programs*.

---

whatis Command

Purpose

Describes what function a command performs.

Syntax

whatis [**-M** *PathName*] *Command ...*

Description

The **whatis** command looks up a given command, system call, library function, or special file name, as specified by the *Command* parameter, from a database you create using the **catman -w** command. The **whatis** command displays the header line from the manual section. You can then issue the **man** command to obtain additional information.

The **whatis** command is equivalent to using the **man -f** command.

> **Note:** When the **/usr/share/man/whatis** database is built from the HTML library using the **catman -w** command, section 3 is equivalent to section 2 or 3. See the **man** command for further explanation of sections.

Flags

-M *PathName*	Specifies an alternative search path. The search path is specified by the *PathName* parameter, and is a colon-separated list of directories in which the **whatis** command expects to find the standard manual subdirectories.

Examples

To find out what the **ls** command does, enter:

```
whatis ls
```

This produces the following output:

```
ls(1)  -Displays the contents of a directory.
```

Files

/usr/share/man/whatis	Contains the **whatis** database.

Related Information

The **apropos** command, **catman** command, **ls** command, **man** command.

---

whatnow Command

Purpose

Starts a prompting interface for draft disposition.

Syntax

whatnow [{ **-draftfolder** **+***folder* | **-nodraftfolder** | *file* } { **-draftmessage** *message* | *file* }] [**-editor** *editor* | **-noedit**] [**-prompt** *string*]

Description

The **whatnow** command provides an interface for the disposition of messages. By default, the interface operates on the current draft message. When you enter the **whatnow** command, the system places you in the interface and returns the following prompt:

```
What now?
```

Within the interface you can manipulate message drafts using the **whatnow** subcommands. To see a listing of the subcommands, press the Enter key at the `What now?` prompt. To exit the interface, press `q`.

If you do not specify the **-draftfolder** flag or if the `Draft-Folder:` entry in the **$HOME/.mh_profile** file is undefined, the **whatnow** command searches your MH directory for a **draft** file. Specifying a message after the **-draftfolder** **+***folder* flag is the same as specifying the **-draftmessage** flag.

To change the default editor for the **whatnow** command, use the **-editor** flag or define the `Editor:` entry in the *UserMhDirectory***/.mh_profile** file.

Note: The **comp**, **dist**, **forw**, or **repl** commands use the same interface as the **whatnow** command.

Flags

-draftfolder **+***folder*	Specifies the folder containing the message. By default, the system uses the *UserMhDirectory***/draft** file. Specifying a message after the **-draftfolder** **+***folder* is the same as using the **-draftmessage** flag.
-draftmessage *message*	Specifies the draft message.
-editor *editor*	Specifies that the value of the *editor* variable is the initial editor for composing or revising the message.

-help Lists the command syntax, available switches (toggles), and version information.

Note: For MH, the name of this flag must be fully spelled out.

file User selected draft file.

message Specifies the message. Use the following references to specify messages:

Number
Number of the message.

cur or . (period)
Current message. This is the default.

first First message in a folder.

last Last message in a folder.

next Message following the current message.

prev Message preceding the current message.

-nodraftfolder
Places the draft in the *UserMhDirectory*/**draft** file.

-noedit Suppresses the initial edit.

-prompt *string*
Uses the specified string as the prompt. The default string is `What now?`.

whatnow Subcommands

The **whatnow** subcommands enable you to edit the message, direct the disposition of the message, or end the processing of the **whatnow** command.

display [*flags*] Displays the message being redistributed or replied to. You can specify any *flags* parameter that is valid for the listing program. (Use the `lproc:` entry in the **$HOME/.mh_profile** file to set a default listing program.) If you specify flags that are invalid for the listing program, the **whatnow** command does not pass the path name of the draft.

edit [*commandstring*] Specifies with the *commandstring* parameter an editor for the message. You can specify the editor and any valid flags to that editor. If you do not specify an editor, the **whatnow** command uses the editor specified by the `Editor:` entry in your *UserMhDirectory*/**.mh_profile** file. If your `Editor:` entry is undefined, the **whatnow** command starts the editor used in the previous editing session.

list [*flags*] Displays the draft. You can specify any *flags* parameter that is valid for the listing program. (To specify a default listing program, set a default `lproc:` entry in the **$HOME/.mh_profile** file.) If you specify any flags that are invalid for the listing program, the **whatnow** command does not pass the path name of the draft.

push [*flags*] Sends the message in the background. You can specify any valid flag for the **send** command.

quit [**-delete**] Ends the **whatnow** session. If you specify the **-delete** flag, the **whatnow** command deletes the draft. Otherwise, the **whatnow** command stores the draft.

refile [*flags*] **+***folder*
Files the draft in the specified folder and supplies a new draft having the previously specified form. You can specify any *flags* parameter that is valid for the command serving as the **fileproc**. (You can set a default `fileproc:` entry in the **$HOME/.mh_profile** file.)

send [*flags*] Sends the message. You can specify any valid flags for the **send** command.

whom [*flags*] Displays the addresses to which the message would be sent. You can specify any valid flags for the **whom** command.

Profile Entries

The following entries are entered in the *UserMhDirectory*/**.mh_profile** file:

`Draft-Folder:`	Sets the default folder for drafts.
`Editor:`	Sets the default editor.
`fileproc:`	Specifies the program used to refile messages.
`LastEditor-next:`	Specifies the editor used after exiting the editor specified by the *LastEditor* variable.
`lproc:`	Specifies the program used to list the contents of a message.
`Path:`	Specifies the *UserMhDirectory*.
`sendproc:`	Specifies the program used to send messages.
`whomproc:`	Specifies the program used to determine the users to whom a message would be sent.

Security

RBAC Disclaimer

Examples

1. To display the original message when you are replying to a message, enter the following at the `What now?` prompt:

   ```
   display
   ```

   The system displays the original message. If you enter the **display** subcommand from a command other than the **dist** or **repl** command, you will receive a system message stating that there is no alternate message to display.
2. To edit the draft message with the `vi` editor, enter the following at the `What now?` prompt:

   ```
   edit vi
   ```
3. To edit the draft message with the default editor specified in your **.mh_profile** file, enter the following at the `What now?` prompt:

   ```
   edit
   ```
4. To list the contents of the draft message you have composed, enter the following at the `What now?` prompt:

   ```
   list
   ```

   The draft message you are composing is displayed.
5. To send the draft message in the background and get a shell prompt immediately, enter the following at the `What now?` prompt:

   ```
   push
   ```

   The draft message is sent and you immediately receive the shell prompt.
6. To quit composing a draft message and save it to a file so that you can later finish composing the message, enter the following at the `What now?` prompt:

   ```
   quit
   ```

The system responds with a message similar to the following.

```
whatnow: draft left on /home/dale/Mail/draft
```

In this example, user `dale`'s draft message is saved to the **/home/dale/Mail/draft** file.

7. To quit composing a draft message and delete the message, enter the following at the `What now?` prompt:

   ```
   quit -delete
   ```

   The shell prompt is displayed when the draft message is deleted.
8. To file the draft message you are composing before you send it, enter the following at the `What now?` prompt:

   ```
   refile +tmp
   ```

   The system responds with a message similar to the following:

   ```
   Create folder "home/dale/Mail/tmp"?
   ```

   In this example, if you answer `yes`, the draft message is filed in user `dale`'s folder `tmp`.
9. To send the draft message you have composed, enter the following at the `What now?` prompt:

   ```
   send
   ```

   The shell prompt is displayed when the message is sent.
10. To verify that all addresses in the draft message are recognized by the mail delivery system, enter the following at the `What now?` prompt:

    ```
    whom
    ```

    The system responds with a message similar to the following:

    ```
    jeanne... User unknown
    dale@venus... deliverable
    ```

    In this example, the mail delivery system recognized `dale@venus` as a correct address, but did not recognize `jeanne` as a correct address.

Files

$HOME/.mh_profile	Specifies the MH user profile.
UserMhDirectory/**draft**	Contains the current message draft.
/usr/bin/whatnow	Contains the **whatnow** command.

Related Information

The **comp** command, **dist** command, **forw** command, **prompter** command, **refile** command, **repl** command, **rmm** command, **scan** command, **send** command, **whom** command.

The **mh_alias** file format, **mh_profile** file format.

Mail applications in *Networks and communication management*.

RBAC in *Security*.

Trusted AIX in *Security*.

whereis Command

Purpose

Locates source, binary, or manual for program.

Syntax

whereis [**-s**] [**-b**] [**-m**] [**-u**] [{ { **-S** | **-B** | **-M** } *Directory* ... }... **-f**] *File* ...

Description

The **whereis** command locates the source, binary, and manuals sections for specified files. The supplied names are first stripped of leading path name components and any (single) trailing extension of the form *.ext* (for example, **.c**). Prefixes of **s.** resulting from use of the Source Code Control System (see **SCCS**) are also dealt with. The command then attempts to find the desired program from a list of standard locations.

A usage message is returned if a bad option is entered. In other cases, no diagnostics are provided.

Flags

If any of the **-b, -s, -m** or **-u** flags are given, the **whereis** command searches only for binary, source, manual, or unusual sections respectively (or any two thereof).

-b Searches for binary sections of a file.
-m Searches for manual sections of a file.
-s Searches for source sections of a file.
-u Searches for unusual files. A file is said to be unusual if it does not have one entry of each requested type. Entering `whereis -m -u *` asks for those files in the current directory which have no documentation.

The **-B, -M,** and **-S** flags can be used to change or otherwise limit the places where the **whereis** command searches. Since the program uses the **chdir** subroutine to run faster, path names given with the **-M, -S** and **-B** flag directory list must be full; for example, they must begin with a **/** (slash).

-B Like **-b**, but adds a directory to search. Change or limit the places where the **whereis** command searches for binaries.
-M Like **-m**, but adds a directory to search. Change or limit the places where the **whereis** command searches for manual sections.
-S Like **-s**, but adds a directory to search. Change or limit the places where the **whereis** command searches for sources
-f Terminates the last **-M, -S** or **-B** directory list and signal the start of file names.

Examples

To find all of the files in the **/usr/ucb** directory that either are not documented in the **/usr/man/man1** directory or do not have source in the **/usr/src/cmd** directory, enter:

```
cd /usr/ucb
whereis -u -M /usr/man/man1 -S /usr/src/cmd -f *
```

Files

/usr/share/man/* Directories containing manual files.
/sbin, **/etc**, **/usr/**{**lib**,**bin**,**ucb**,**lpp**} Directories containing binary files.
/usr/src/* Directories containing source code files.

Related Information

The **chdir** subroutine.

---

which Command

Purpose

Locates a program file, including aliases and paths.

Syntax

which [*Name* ...]

Description

The **which** command takes a list of program names and looks for the files that run when these names are given as commands. The **which** command expands each argument, if it is aliased, and searches for it along the user's path. The aliases and paths are taken from the **.cshrc** file in the user's home directory. If the **.cshrc** file does not exist, or if the path is not defined in the **.cshrc** file, the **which** command uses the path defined in the user's environment.

A diagnostic is given if a name is aliased to more than a single word or if an executable file with the argument name is not found in the path.

In the Korn shell, you can use the **whence** command to produce a more verbose report. See ″Korn shell or POSIX shell built-in commands″ in *Operating system and device management* for more information on the **whence** command.

Examples

To find the executable file associated with a command name of `lookup`:

```
which lookup
```

Files

$HOME/.cshrc	Contains the source of aliases and path values.

Related Information

The **csh** command, **find** command, **file** command, **ksh** command, **sh** command, **whereis** command.

Shells in *Operating system and device management* describes shells, the different types, and how they affect the way commands are interpreted.

Commands in *Operating system and device management*.

---

which_fileset Command

Purpose

Searches the **/usr/lpp/bos/AIX_file_list** file for a specified file name or command. This command only applies to AIX 4.2.1 or later.

Syntax

which_fileset [*File*]

Description

The **which_fileset** command searches the **/usr/lpp/bos/AIX_file_list** file for a specified file name or command name, and prints out the name of the fileset that the file or command is shipped in.

The **/usr/lpp/bos/AIX_file_list** file is large and not installed automatically. You must install the **bos.content_list** fileset to receive this file.

The *File* parameter can be the command name, the full path name, or a regular expression search pattern.

Examples

1. To display which fileset the dbx command is shipped in, enter:

   ```
   which_fileset dbx
   ```

   The screen displays the following:

   ```
   /usr/bin/dbx > /usr/ccs/bin/dbx             bos.adt.debug 4.2.1.0
   /usr/ccs/bin/dbx                            bos.adt.debug 4.2.1.0
   ```

2. To display all commands and paths containing the *sendmail* string, enter:

   ```
   which_fileset sendmail.*
   ```

   T he screen displays the following:

   ```
   /usr/ucb/mailq > /usr/sbin/sendmail    bos.compat.links 4.2.0.0
   /usr/ucb/newaliases > /usr/sbin/sendmail bos.compat.links 4.2.0.0
   /usr/lib/nls/msg/Ca_ES/sendmail87.cat  bos.msg.Ca_Es.net.tcp.client 4.2.0.0
   /usr/lib/nls/msg/ca_ES/sendmail87.cat  bos.msg.ca_Es.net.tcp.client 4.2.0.0
   /usr/lib/nls/msg/cs_CZ/sendmail87.cat  bos.msg.cs_CZ.net.tcp.client 4.2.0.0
   /usr/lib/nls/msg/De_DE/sendmail87.cat  bos.msg.De_DE.net.tcp.client 4.2.0.0
   /usr/lib/nls/msg/de_DE/sendmail87.cat  bos.msg.de_DE.net.tcp.client 4.2.0.0
   /usr/lib/nls/msg/En_US/sendmail87.cat  bos.msg.En_US.net.tcp.client 4.2.0.0
   /usr/lib/nls/msg/en_US/sendmail87.cat  bos.msg.en_US.net.tcp.client 4.2.0.0
   /usr/lib/nls/msg/Es_ES/sendmail87.cat  bos.msg.Es_ES.net.tcp.client 4.2.0.0
   /usr/lib/nls/msg/es_ES/sendmail87.cat  bos.msg.es_ES.net.tcp.client 4.2.0.0
   /usr/lib/nls/msg/Fr_FR/sendmail87.cat  bos.msg.Fr_FR.net.tcp.client 4.2.0.0
   /usr/lib/nls/msg/fr_FR/sendmail87.cat  bos.msg.fr_FR.net.tcp.client 4.2.0.0
   /usr/lib/nls/msg/hu_HU/sendmail87.cat  bos.msg.hu_HU.net.tcp.client 4.2.0.0
   /usr/lib/nls/msg/It_IT/sendmail87.cat  bos.msg.It_IT.net.tcp.client 4.2.0.0
   /usr/lib/nls/msg/it_IT/sendmail87.cat  bos.msg.it_IT.net.tcp.client 4.2.0.0
   /usr/lib/nls/msg/Ja_JP/sendmail87.cat  bos.msg.Ja_JP.net.tcp.client 4.2.0.0
   /usr/lib/nls/msg/ja_JP/sendmail87.cat  bos.msg.ja_JP.net.tcp.client 4.2.0.0
   /usr/lib/nls/msg/ko_KR/sendmail87.cat  bos.msg.ko_KR.net.tcp.client 4.2.0.0
   /usr/lib/nls/msg/pl_PL/sendmail87.cat  bos.msg.pl_PL.net.tcp.client 4.2.0.0
   /usr/lib/nls/msg/ru_RU/sendmail87.cat  bos.msg.ru_RU.net.tcp.client 4.2.0.0
   /usr/lib/nls/msg/Sv_SE/sendmail87.cat  bos.msg.Sv_SE.net.tcp.client 4.2.0.0
   /usr/lib/nls/msg/sv_SE/sendmail87.cat  bos.msg.sv_SE.net.tcp.client 4.2.0.0
   /usr/lib/nls/msg/ZH_CN/sendmail87.cat  bos.msg.ZH_CN.net.tcp.client 4.2.0.0
   /usr/lib/nls/msg/zh_CN/sendmail87.cat  bos.msg.zh_CN.net.tcp.client 4.2.0.0
   /usr/lib/nls/msg/Zh_TW/sendmail87.cat  bos.msg.Zh_TW.net.tcp.client 4.2.0.0
   /usr/lib/nls/msg/zh_TW/sendmail87.cat  bos.msg.zh_TW.net.tcp.client 4.2.0.0
   /etc/sendmail.cf                       bos.net.tcp.client.4.2.1.0
   /usr/lib/sendmail > /usr/sbin/sendmail bos.net.tcp.client.4.2.1.0
   /usr/sbin/mailq > /usr/sbin/sendmail   bos.net.tcp.client.4.2.1.0
   /usr/sbin/newaliases > /usr/sbin/sendmail bos.net.tcp.client.4.2.1.0
   /usr/sbin/sendmail                     bos.net.tcp.client.4.2.1.0
   ```

3. To find where the **/usr/sbin/which_fileset** command is shipped, enter:

   ```
   which_fileset /usr/bin/which_fileset
   ```

   The screen displays:

   ```
   /usr/sbin/which_fileset      bos.rte.install 4.2.1.0
   ```

who Command

Purpose

Identifies the users currently logged in.

Syntax

who [**-a** | **-b -d -i -l -m -p -q -r -s -t -u -w -A -H -T -X**] [*File*]

who am { **i** | **I** }

Description

The **who** command displays information about all users currently on the local system. The following information is displayed: login name, tty, date and time of login. Typing `who am i` or `who am I` displays your login name, tty, date and time you logged in. If the user is logged in from a remote machine, then the host name of that machine is displayed as well.

The **who** command can also display the elapsed time since line activity occurred, the process ID of the command interpreter (shell), logins, logoffs, restarts, and changes to the system clock, as well as other processes generated by the initialization process.

The general output format of the **who** command is as follows:

```
Name [State] Line Time [Activity] [Pid] [Exit] (Hostname)
```

where:

`Name`	Identifies the user's login name.
`State`	Indicates whether the line is writable by everyone (see the **-T** flag).
`Line`	Identifies the line name as found in the **/dev** directory.
`Time`	Represents the time when the user logged in.
`Activity`	Represents the hours and minutes since activity last occurred on that user's line. A . (dot) here indicates line activity within the last minute. If the line has been quiet more than 24 hours or has not been used since the last system startup, the entry is marked as old.
`Pid`	Identifies the process ID of the user's login shell.
`Term`	Identifies the process termination status (see the **-d** flag). For more information on the termination values, refer to the **wait** subroutine or to the **/usr/include/sys/signal.h** file.
`Exit`	Identifies the exit status of ended processes (see the **-d** flag).
`Hostname`	Indicates the name of the machine the user is logged in from.

To obtain information, the **who** command usually examines the **/etc/utmp** file. If you specify another file with the *File* parameter, the **who** command examines that file instead. This new file is usually the **/var/adm/wtmp** or **/etc/security/failedlogin** file.

If the *File* parameter specifies more than one file name, only the last file name will be used.

Note: This command only identifies users on the local node.

Flags

-a	Processes the **/etc/utmp** file or the named file with all information. Equivalent to specifying the **-bdlprtTu** flags.
-b	Indicates the most recent system startup time and date.

-d Displays all processes that have expired without being regenerated by init. The exit field appears for dead processes and contains the termination and exit values (as returned by wait) of the dead process. (This flag is useful for determining why a process ended by looking at the error number returned by the application.)

-l Lists any login process.

-m Displays information about the current terminal only. The **who -m** command is equivalent to the **who am i** and **who am I** commands.

-p Lists any active process that is currently active and has been previously generated by init.

-q Prints a quick listing of users and the number of users on the local system.

-r Indicates the current run-level of the process.

-s Lists only the name, line, and time fields. This flag is the default; thus, the **who** and **who -s** commands are equivalent.

-t Indicates the last change to the system clock by the root user using the **date** command. If the **date** command has not been run since system installation, the **who -t** command produces no output.

-u or **-i** Displays the user name, tty, login time, line activity, and process ID of each current user.

-A Displays all accounting entries in the **/etc/utmp** file. These entries are generated through the **acctwtmp** command.

-H Displays a header (title).

-T or **-w** Displays the state of the tty and indicates who can write to that tty as follows:

+ Writable by anyone.

- Writable only by the root user or its owner.

? Bad line encountered.

-X Prints all available characters of each user name instead of truncating to the first 8 characters. The user name is also moved to the last column of the output.

Exit Status

This command returns the following exit values:

0 Successful completion.

>0 An error occurred.

Examples

1. 1. To display information about who is using the local system node, type:

   ```
   who
   ```

   Information similar to the following is displayed:

   ```
   pts/1      Nov  9 00:20   long_username_greater_than_eight_characters  (localhost)
   ```

2. To display your user name, type:

   ```
   who am i
   ```

   Information similar to the following is displayed:

   ```
   george lft/0 Jun 8 08:34
   ```

3. To display a history of logins, logouts, system startups, and system shutdowns, type:

   ```
   who /var/adm/wtmp
   ```

   Information similar to the following is displayed:

   ```
   hank   lft/0   Jun   8   08:34  (ausnix5)
   john   lft/0   Jun   8   08:34  (JIKey)
   mary   lft/0   Jun   8   08:22  (machine.austin.ibm)
   jan    pts4    Jun   8   09:19  (puff.wisc.edu)
   ```

4. To display the run-level of the local system node, type:

```
who  -r
```

Information similar to the following is displayed:

```
. run-level 2 Jun 8 04:15 2 0 s
```

5. To display any active process that is currently actively and has been previously generated by init, type:

```
who  -p
```

Information similar to the following is displayed:

```
srcmstr   .    Jun 8  04:15  old  2896
cron      .    Jun 8  04:15  old  4809
uprintfd  .    Jun 8  04:15  old  5158
```

6. To process the **/var/adm/wtmp** file with the **-bdlprtTu** flags specified, type:

```
who  -a /var/adm/wtmp
```

Information similar to the following is displayed:

```
   .        system boot Jun 19 10:13
   .        run-level 2 Jun 19 10:13
   .            .       Jun 19 10:14    old
   .            .       Jun 19 10:14    old
   .            .       Jun 19 10:14    old
rc        -     .       Jun 19 10:13    old
   .            .       Jun 19 10:16    old
   .            .       Jun 19 10:14    old
srcmstr   -     .       Jun 19 10:14    old
rctcpip   -     .       Jun 19 10:14    old
rcdce     -     .       Jun 19 10:14    old
rccm      -     .       Jun 19 10:15    old
dceupdt   -     .       Jun 19 10:15    old
rcnfs     -     .       Jun 19 10:15    old
cron      -     .       Jun 19 10:16    old
piobe     -     .       Jun 19 10:16    old
qdaemon   -     .       Jun 19 10:16    old
writesrv  -     .       Jun 19 10:16    old
uprintfd  -     .       Jun 19 10:16    old
   .            .       Jun 19 10:16    old
LOGIN     - lft0        Jun 19 10:16    old
   .            .       Jun 19 10:16    old
   .            .       Jun 19 10:16    old
```

Files

/etc/utmp	Contains user and accounting information.
/etc/security/failedlogin	Contains the history of all invalid logins.
/var/adm/wtmp	Contains the history of all logins since the file was last created.
/usr/include/sys/signal.h	Contains a list of termination values.

Related Information

The **date** command, **mesg** command, **whoami** command, **su** command.

The **wait** subroutine.

whoami Command

Purpose
Displays your login name.

Syntax
whoami

Description
The **whoami** command displays your login name. Unlike using the command **who** and specifying **am i**, the **whoami** command also works when you have root authority since it does not examine the **/etc/utmp** file.

Files

/etc/passwd Contains user IDs.

Related Information
The **who** command.

whodo Command

Purpose
Lists the jobs being performed by users on the system.

Syntax
whodo [**-h**] [**-l**] [**-X**] [*User*]

Description
Prints information on all processes for a terminal, as well as the child processes.

By default, the output generated by the command for each active logged user will contain name of the terminal, user ID, date login time. The output is headed by the date, time and machine name. This information is followed by a record of active processes associated with that user ID. Each record shows the terminal name, process-ID, CPU minutes and seconds used, and process name.

Flags

-h Suppress the heading that is printed on the output.

-l Produce a long form of output. A summary of the current activity on the system is printed. The summary includes the following:

User Who is logged on.

tty Name of the tty the user is on.

login@ Time of day the user logged on.

idle Number of minutes since a program last attempted to read from the terminal.

JCPU System unit time used by all processes and their children on that terminal.

PCPU System unit time used by the currently active process.

what Name and parameters of the current process.

The heading line of the summary shows the current time of day, how long the system has been up, the number of users logged into the system.

-X Prints all available characters of each user name instead of truncating to the first 8 characters. The user name is also moved to the last column of the output.

Parameters

User Limits output to all sessions pertaining to the user specified with *User*. More than one user name cannot be specified at a time.

Exit Status

0 The command completed successfully.

>0 An error occurred.

Examples

1. When the **whodo** command is invoked on host ″linguist″ without any flags or parameters, the output looks similar to the following:

```
Sun Jul 28 16:27:12 2002
linguist

lft0 jeffg 8:15
         ? 4136  0:00 dtlogin
         ? 3408  4:55 dtsession
         ? 2072  4:37 dtwm
         ? 17310  0:00 dtexec
         ? 20904  5:53 dtterm
     pts/0 22454  0:00 ksh
     pts/0 4360  0:07 ksh
     pts/0 25788  0:00 whodo
         ? 23672  0:00 dtexec
         ? 27536  0:00 dtterm
     pts/3 21508  0:00 ksh
         ? 23888  0:00 dtexec
         ? 24384  2:49 dtterm
     pts/2 24616  0:00 ksh
     pts/2 25002  0:04 ksh
     pts/2 26110  0:00 ksh
         ? 25276  0:00 dtexec
         ? 27090  0:31 dtterm
     pts/1 24232  0:00 ksh
     pts/1 23316  0:01 ksh
         ? 12566  4:23 dtfile
         ? 21458  1:35 dtfile
```

```
pts/0 jeffg 8:16
     pts/0 22454  0:00 ksh
     pts/0 4360  0:07 ksh
     pts/0 25788  0:00 whodo

pts/1 jeffg 17:8
     pts/1 24232  0:00 ksh
     pts/1 23316  0:01 ksh

pts/2 jeffg 17:20
     pts/2 24616  0:00 ksh
     pts/2 25002  0:04 ksh
     pts/2 26110  0:00 ksh

pts/3 root 16:26
     pts/3 21508  0:00 ksh
```

2. The command **whodo -l** on the host linguist produces the following output:

```
  04:33PM   up 20 day(s), 22 hr(s), 51 mins(s)  5 user(s)
User     tty          login@       idle      JCPU      PCPU what
jeffg    lft0        08Jul02   21day(s)                   /usr/sbin/getty /de
jeffg    pts/0       08Jul02                14:00         7 whodo -l
jeffg    pts/1       16Jul02   10day(s)        44         9 /usr/bin/ksh
jeffg    pts/2       12Jul02         11      8:39         4 /usr/bin/ksh
root     pts/3       04:26PM          7                     -ksh
```

3. The command **whodo -lX** on the host kq11 produces the following output:

```
 12:50AM   up 3 day(s),  1 hr(s), 41 mins(s)  4 user(s)
tty          login@       idle      JCPU      PCPU what            User
tty0        Wed11PM    2day(s)                    -ksh            root
pts/0       12:12AM                               tn 0            root
pts/1       12:20AM                               whodo -lX       long_username_greater_than_eight_characters
pts/2       Fri05AM    2day(s)                    -ksh            root
```

Files

/usr/sbin/whodo	Contains the **whodo** command.
/etc/utmp	Contains the list of users.

Related Information

The **ps** command, **who** command.

---

whois Command

Purpose

Identifies a user by user ID or alias.

Syntax

whois [**-h** *HostName*] [**.** | **!**] [*****] *Name* [**. . .**]

whois ?

Description

The **/usr/bin/whois** command searches a user name directory and displays information about the user ID or nickname specified in the *Name* parameter. The **whois** command tries to reach ARPANET host `internic.net` where it examines a user-name database to obtain information. The **whois** command should be used only by users on ARPANET. Refer to RFC 812 for more complete information and recent changes to the **whois** command.

Note: If your network is on a national network, such as ARPANET, the host name is hard-coded as `internic.net`.

The *Name* [**. . .**] parameter represents the user ID, host name, network address, or nickname on which to perform a directory search. The **whois** command performs a wildcard search for any name that matches the string preceding the optional ... (three periods).

Flags

.	Forces a name-only search for the name specified in the *Name* parameter.
!	Displays help information for the nickname or handle ID specified in the *Name* parameter.
*****	Displays the entire membership list of a group or organization. If there are many members, this can take some time.
?	Requests help from the ARPANET host.
-h *HostName*	Specifies an alternative host name. The default host name on the ARPANET is `internic.net`. You can contact the other major ARPANET user-name database, `nic.ddn.mil`, by specifying the **-h** *HostName* flag.

Examples

1. To display information about ARPANET registered users by the name of Smith, enter:

   ```
   whois Smith
   ```

2. To display information about ARPANET registered users that use the handle Hobo, enter:

   ```
   whois  !Hobo
   ```

3. To display information about ARPANET registered users with the name of John Smith, enter:

   ```
   whois  .Smith, John
   ```

4. To display information about ARPANET registered users whose names or handles begin with the letters HEN, enter:

   ```
   whois HEN  ...
   ```

5. To get help information for the **whois** command, enter:

   ```
   whois  ?
   ```

Related Information

The **who** command.

The **named.conf** file format.

Communications and networks in *Networks and communication management*.

---

whom Command

Purpose

Manipulates Message Handler (MH) addresses.

Syntax

whom [**-alias** *File* ...] [**-nocheck** | **-check**] [{ **-draftfolder** +*Folder* | **-nodraftholder** | *File* }{ **-draftmessage** *Message* | **-draft***File* }]

Description

The **whom** command does the following:

- Expands the headers of a message into a set of addresses.
- Lists the addresses of the proposed recipients of a message.
- Verifies that the addresses are deliverable to the transport service.

> **Note:** The **whom** command does not guarantee that addresses listed as being deliverable will actually be delivered.

A message can reside in a draft folder or in a file. To specify where a message resides, use the **-draft**, **-draftfolder**, or **-draftmessage** flag.

If you do not specify the **-draftfolder** flag or if the `Draft-Folder:` entry in the **$HOME/.mh_profile** file is undefined, the **whom** command searches your MH directory for a **draft** file. Specifying a message after the **-draftfolder +***Folder* flag is the same as specifying the **-draftmessage** flag.

Flags

-alias *File*
: Specifies a file to search for mail aliases. By default, the system searches the **/etc/mh/MailAliases** file.

-draft
: Uses the header information in the *UserMhDirectory***/draft** file if it exists.

-draftfolder +*Folder*
: Uses the header information from the draft message in the specified folder. If you specify a draft folder that doesn't exist, the system creates one for you.

-draftmessage *Message*
: Uses the header information from the specified draft message.

-help
: Lists the command syntax, available switches (toggles), and version information.

> **Note:** For MH, the name of this flag must be fully spelled out.

Message
: Specifies the message draft. Use the following to specify messages:

  Number
  : Number of the message.

  cur or . (period)
  : Current message. This is the default.

  first
  : First message in a folder.

  last
  : Last message in a folder.

  next
  : Message following the current message.

  prev
  : Message preceding the current message.

-nodraftfolder
: Undoes the last occurrence of the **-draftfolder +***Folder* flag.

> **Note:** Two other flags, **-check** and **-nocheck**, are also available. These flags have no effect on how the **whom** command performs verification. The **-check** and **-nocheck** flags are provided for compatibility only.

Profile Entries

The following entries are entered in the *UserMhDirectory***/.mh_profile** file:

`Draft-Folder:`
: Sets your default folder for drafts.

`postproc:`
: Specifies the program used to post messages.

Security

RBAC Disclaimer

Examples

To list and verify the addresses of the proposed recipients of a message, enter the addressees and subject of the message at the respective prompt, as follows:

```
To: d77@nostromo
Subject: a test
```

When prompted again, enter the text of the message:

```
-------Enter initial text
test
-------
```

After the `whatnow` prompt, enter the **whom** command:

```
whatnow>>> whom
```

The address of the proposed recipients of the message is then displayed:

```
lance...
d77@nostromo... deliverable
```

Files

$HOME/.mh_profile	Specifies the MH user profile.
/usr/bin/whom	Contains the **whom** command.

Related Information

The **ali** command**, post** command**, whatnow** command.

The **mh_alias** file format, **mh_profile** file format.

Mail applications in *Networks and communication management*.

RBAC in *Security*.

Trusted AIX in *Security*.

---

wlmassign command

Purpose

Manually assigns processes to a Workload Management class or cancels prior manual assignments for processes.

Syntax

wlmassign [**-s** | **-S**] [**-u** | *Class_Name*] [*PID_List*] [**-g** *Pgid_List*]

Description

The **wlmassign** command:

- Assigns a set of processes specified by a list of process identifiers (pids) and/or process group identifiers (pgids) to a specified superclass or subclass or both, thus overriding the automatic class assignment or a prior manual assignment.

- Cancels a previous manual assignment for the processes specified in *pid_list* or *pgid_list*.

The **wlmassign** command allows to specify processes using a list of PIDs, a list of pgids, or both. The format of these lists is:

pid[**,pid**[**,pid**[**...**]]]

or

pgid[**,pgid**[**,pgid**[**...**]]]

The name of a valid superclass or subclass must be specified to manually assign the target processes to a class. If the target class is a superclass, each process is assigned to one of the subclasses of the specified superclass according to the assignment rules for the subclasses of this superclass.

A manual assignment remains in effect (and a process remains in its manually assigned class) until:

- The process terminates
- Workload Management (WLM) is stopped. When WLM is restarted, the manual assignments in effect when WLM was stopped are lost.
- The class the process has been assigned to is deleted
- A new manual assignment overrides a prior one.
- The manual assignment for the process is canceled using the **-u** flag.
- The process calls the **exec()** routine.

The name of a valid superclass or subclass must be specified to manually assign the target processes to a class. The assignment can be done or canceled at the superclass level, the subclass level or both. When a manual assignment is canceled for a process, or the process calls **exec()**, the process is then subject to automatic classification; if inheritance is enabled for the class that the process is in, it will remain in that class, otherwise the process will be reclassified according to the assignment rules. The interactions between automatic assignment (inheritance and rules), inheritance and manual assignment are detailed in the Workload management in *Operating system and device management*.

For a manual assignment:

- If the *Class_Name* is the name of a superclass, the processes in the list are assigned to the superclass. The subclass is then determined, for each process, using the assignment rules for the subclasses of the target superclass.
- If the class name is a subclass name (supername.subname), the processes by default are assigned to both the superclass and the subclass. The processes can be assigned to the superclass only by specifying the **-S** flag or the subclass only by specifying the **-s** flag.

```
wlmassign   super1.sub2  -S  pid1
```

is equivalent to:

```
wlmassign super1 pid1
```

To assign a process to a class or cancel a prior manual assignment, the user must have authority both on the process and on the target class. These constraints translate into the following:

- The root user can assign any process to any class.
- A user with administration privileges on the subclasses of a given superclass (that is, the user or group name matches the user or group names specified in the attributes **adminuser** and **admingroup** of the superclass) can manually reassign any process from one of the subclasses of this superclass to another subclass of the superclass.
- Users can manually assign their own processes (same real or effective user ID) to a class, for which they have manual assignment privileges (that is, the user or group name matches the user or group names specified in the attributes **authuser** and **authgroup** of the superclass or subclass).

This defines 3 levels of privilege among the persons who can manually assign processes to classes, root being the highest. For a user to modify or terminate a manual assignment, they must have at least the same level of privilege as the person who issued the last manual assignment.

Note: The **wlmassign** command works with currently loaded WLM configuration. If the current configuration is a set, and the assignment is made to a class which does not exist in all configurations in the set, the assignment will be lost when a configuration that does not contain the class becomes active (class is deleted).

Flags

-g *Pgid_list*	Indicates that the following list is a list of pgids.
-S	Specifies that the assignment is to be done or canceled at the superclass level only. This flag is used with a subclass name of the form *supername.subname*.
-s	Specifies that the assignment is to be done or canceled at the subclass level only. This flag is used with a subclass name of the form *supername.subname*.
-u	Cancel any manual assignment in effect for the processes in the pid_list or the pgid_list. If none of the **-s** or **-S** flags are used, this cancels the manual assignments for both the superclass and the subclass level.

Related Information

The **chclass** command, **lsclass** command, **mkclass** command, and **rmclass** command.

The concept article about Workload management in *Operating system and device management*.

---

wlmcheck command

Purpose

Check automatic assignment rules and/or determines the Workload Manager class a process with a specified set of attributes would be classified in.

Syntax

wlmcheck [**-d** *Config*] [**-a** *Attributes*] [**-q**]

Description

The **wlmcheck** command with no arguments, gives the status of Workload Manager (WLM) and makes some coherency checks:

- Displays the current status of WLM (running/non running, active/passive, rsets bindings active, total limits enabled).
- Displays the status files that report the last loading errors, if any. If 'current' configuration is a set, this applies to all configurations in the set, and messages logged by the WLM daemon are reported.
- Checks the coherency of the attributes and assignment rules file(s) (such as, the existence of the classes, validity of user and group names, existence of application file names, etc).

If the **-d** *Config* flag is not specified, the checks are performed on the 'current' configuration.

The **wlmcheck** command can apply to a configuration set. In this case, the checks mentioned above are performed on all configurations of the set, after checking the set itself. Superclass names are reported in the form 'config/superclass' to indicate the regular configuration which they belong to.

Specifying a configuration with **-d** *Config* performs the checks on the *Config* configuration or set instead of 'current'. This does not change the reporting of status files and of the WLM daemon log, which only applies to the active configuration.

With the **-a** flag, **wlmcheck** displays the class that the process with attributes specified by *Attributes* would be assigned to, according to the rules for the current or specified configuration or configuration set. The format of the *Attributes* string is similar to an entry in the *rules* file, with the following differences:

- The class field is omitted (it is actually an output of **wlmcheck**)
- Each field can have at most one value. Exclusion (!), attribute groupings ($), comma separated lists, and wild cards are not allowed. For the *type* field, the AND operator ″+″ is allowed, since a process can have several of the possible values for the type attribute at the same time. For instance a process can be a 32 bit process and call plock, or be a 64 bit fixed priority process.
- At least one field must be specified (have a value different from a hyphen (-).

In addition, the first 2 fields are mandatory. The other fields, if not present default to a hyphen (-) which mean that any value in the corresponding field of an assignment rule is a match. When one or more of the fields in the attribute string are either not present or specified as a hyphen (-), the string is likely to match more than one rule. In this case, **wlmcheck** displays all the classes corresponding to all the possible matches.

Example of valid attribute strings:

```
$ wlmcheck -a "- root system /usr/lib/frame/framemaker - -"
$ wlmcheck -a "- - staff - 32bit+fixed"
$ wlmcheck -a "- bob"
```

Flags

-a *Attributes*	Passes a set of values for the classification attributes of the process in order to determine which class the process would be put into. This is a way to check that the assignment rules are correct and classify processes as expected.
-d *Config*	Uses the WLM property files in **/etc/wlm/Config** (which may indicate a set of time-based configurations) instead of **/etc/wlm/current**.
-q	Suppresses the output of the status of the latest activation/update of WLM and of messages logged by the WLM daemon (quiet mode).

Security

RBAC Disclaimer

Files

classes	Contains the names and definitions of the classes.
limits	Contains the resource limits enforced on the classes.
rules	Contains the automatic assignment rules.
shares	Contains the resource shares allocated to the classes.

Related Information

The **chclass** command, **lsclass** command, **mkclass** command, **rmclass** command.

The **rules** file.

RBAC in *Security*.

Trusted AIX in *Security*.

wlmcntrl Command

Purpose

Starts or stops the Workload Manager.

Syntax

wlmcntrl [[**-a** | **-c** | **-p**] [**-T** [**class** | **proc**] [**-g**] [**-d** *Config_Dir*] [**-o** | **-q**]

wlmcntrl -u [**-S** *Superclass* | **-d** *Config_Dir*]

Description

The **wlmcntrl** command stops, starts, updates or queries the state of Workload Manager (WLM). When starting or updating WLM, the WLM property files for the target configuration are pre-processed, and the data is loaded into the kernel. WLM can be started in two different modes:

- An active mode where WLM monitors and regulates the processor, memory and disk I/O utilization of the processes in the various classes.
- A passive mode where WLM only monitors the resource utilization without interfering with the standard operating system resource allocation mechanisms.

The active mode is the usual operating mode of WLM.

The classes, their limits and shares are described respectively in the **classes**, **limits**, and **shares** files. The automatic assignment rules are taken from the **rules** file. The class properties files for the superclasses of the WLM configuration **Config** are located in the subdirectory **/etc/wlm/Config**. The class properties files for the subclasses of the superclass Super of the configuration Config are located in **/etc/wlm/Config/Super**. The standard configuration shipped with the operating system is in **/etc/wlm/standard**. The current configuration is the one in the directory pointed to by the symbolic link **/etc/wlm/current**.

When the **-d** *Config_dir* flag is not used, **wlmcntrl** uses the configuration files in the directory pointed to by the symbolic link **/etc/wlm/current**.

When the **-d** *Config_dir* flag is used, **wlmcntrl** uses the configuration files in **/etc/wlm/Config_dir** and updates the **/etc/wlm/current** symbolic link to point to **/etc/wlm/Config_dir**, making **/etc/wlm/Config_dir** the current configuration. This is the recommended way to make **/etc/wlm/Config_dir** the current configuration.

When updating WLM using the **-u** flag, an empty string can be passed as *Config_dir* with the **-d** flag:

```
wlmcntrl -u -d ""
```

will simply refresh (reload) the assignment rules of the current configuration into the kernel without reloading the class definitions. This can be useful when a prior activation of WLM detected that some application files could not be accessed. After the system sdministrator has fixed the problems with either the rules or the files, this command can be used to reload only the rules.

The WLM configuration **Config** may also be a set of time-based configurations, in which case the subdirectory **/etc/wlm/Config** does not contain the properties files, but a list of configurations and the times of the week when they apply. The properties files are still in the subdirectory of each regular configuration of the set. When WLM is started or updated which such a set, a daemon is responsible for switching regular configurations of the set when the applicable one changes.

Note: This command is not supported when executed within a workload partition.

Flags

-a	Starts WLM in active mode or switches from passive to active mode. This is the default when no flag other than **-d**, **-g**, or **-T** is specified.
-c	Starts WLM in processor-only mode or switches from any mode to processor-only mode. In this mode, the WLM accounts for all resources, but only processor resource is regulated.
-d *Config_dir*	Uses **/etc/wlm/Config_dir** as an alternate directory for the WLM configuration (containing the classes, limits, shares and rules files) or configuration set (containing the list of configurations and the time tanges when they apply). This makes **/etc/wlm/Config_dir** the current configuration. This flag is effective when starting the WLM in active, processor-only or passive mode, or when updating the WLM. This flag cannot be used in conjunction with the **-o** and **-q** flags or when switching from a mode (among active, processor-only and passive) to another.
-g	Instructs WLM to ignore any potential resource set bindings. This means that all classes have access to the whole resource set of the system, regardless of whether or not they use a restricted resource set.
-o	Stops Workload Manager.
-p	Start WLM in passive mode or switches from any mode to passive mode. In this mode, the WLM accounts for all resources, but no resource is regulated.

-q Queries the current state of WLM. Returns:

0	WLM is running in active mode.
1	WLM is not started.
2	WLM is running in passive mode.
3	WLM is running in active mode with no rset bindings.
4	WLM is running in passive mode with no rset bindings.
5	WLM is running in active mode for processor only
6	WLM is running in active mode for processor only with no rset bindings.
16	WLM is running in active mode, process total accounting is off.
18	WLM is running in passive mode, process total accounting is off.
19	WLM is running in active mode with no rset bindings, process total accounting is off.
20	WLM is running in passive mode with no rset bindings, process total accounting is off.
21	WLM is running in active mode for processor only, process total accounting is off.
22	WLM is running in active mode for processor only with no rset bindings, process total accounting is off.
32	WLM is running in active mode, class total accounting is off.
34	WLM is running in passive mode, class total accounting is off.
35	WLM is running in active mode with no rset bindings, class total accounting is off.
36	WLM is running in passive mode with no rset bindings, class total accounting is off.
37	WLM is running in active mode for processor only, class total accounting is off.
38	WLM is running in active mode for processor only with no rset bindings, class total accounting is off.
48	WLM is running in active mode, class and process total accounting are off.
50	WLM is running in passive mode, class and process total accounting are off.
51	WLM is running in active mode with no rset bindings, class and process total accounting are off.
52	WLM is running in passive mode with no rset bindings, class and process total accounting are off.
53	WLM is running in active mode for processor only, class and process total accounting are off.
54	WLM is running in active mode for processor only with no rset bindings, class and process total accounting are off.

A message indicating the current state of WLM is printed on STDOUT.

-S *Superclass* Requests an update of WLM that is limited to the subclasses of the Superclass. Use this flag with the **-u** flag. If the running configuration is a set of time-based configurations, Superclass must be given in the form ″config/Superclass″ where ″config″ is the regular configuration of the set which the Superclass belongs to. If ″config″ is the currently active configuration of the set, the changes will take effect immediately, else they will take effect at the next time ″config″ will be made active.

-T Disables both class and process total limits accounting and regulation.

-T class Disables only class total limits accounting and regulation.

-T proc Disables only process total limits accounting and regulation.

-u Updates the WLM. A single update operation can change the attributes, limits and shares of existing classes and/or add or remove classes. If the running configuration is a set, this operation refreshes the set description along with the content of all configurations of the set. Update can be used by a user with root authority to switch to an alternate configuration or configuration set. Update can also be used by a superclass administrator to update only the subclasses of the superclass he has administrative access to (using the **-S** flag).

Security

Access Control: Starting, stopping, switching from one mode to another, and updating superclasses or a configuration set requires root privileges. Updating the subclasses of a given superclass requires only admin user or admin group privileges (superclass administrator). Any user can query the state of WLM.

RBAC Disclaimer

Files

classes	Contains the names and definitions of the classes.
limits	Contains the resource limits enforced on the classes.
rules	Contains the automatic assignment rules.
shares	Contains the resource shares allocated to the classes.
description	Contains the description text for each configuration.
groupings	Contains attribute value groupings for the configuration

Related Information

The **chclass** command, **confsetcntrl** command, **lsclass** command, **lswlmconf** command, **mkclass** command, and **rmclass** command.

The Workload Management in *Operating system and device management*.

RBAC in *Security*.

Trusted AIX in *Security*.

---

wlmmon and wlmperf Commands

Purpose

The **wlmmon** and **wlmperf** commands provide graphical views of Workload Manager (WLM) resource activities by class.

Syntax

wlmmon

wlmperf

Description

The **wlmmon** and **wlmperf** commands generate resource usage reports of system WLM activity. The **wlmperf** command, which is part of the Performance Toolbox (PTX), can generate reports from trend recordings made by the PTX daemons for periods covering minutes, hours, days, weeks, or months. The **wlmmon** command, which ships with the base AIX, generates reports only for the latest 24-hour period and has no usage options. Three types of visual reports can be generated:

- Snapshot Display

- Detailed Display
- Tabulation Display

The type of report can be customized to cover specified WLM classes over specific time periods. In addition, the WLM activity from two different time periods can be compared (trended) for any chosen display type.

These reports are generated from data that is collected using the same mechanism as the **wlmstat** command. However, the **wlmmon** and **wlmperf** commands use recordings made by a daemon that must operate at all times to collect WLM data. For the **wlmmon** command, this daemon is called **xmwlm**, and ships with the base AIX. For the **wlmperf** command, this daemon is called **xmtrend** and ships with the PTX.

Analysis Overview

While the **wlmstat** command provides a per-second view of WLM activity, it is not suited for the long-term analysis. To supplement the **wlmstat** command, the **wlmmon** and **wlmperf** commands provide reports of WLM activity over much longer time periods, with minimal system impact. The reports generated by this tool are based off samplings made by the associated recording daemon. These daemons sample the WLM and system statistics at a very high rate (measured in seconds), but only record supersampled values at low rate (measured in minutes). These values represent the minimum, maximum, mean, and standard deviation values for each collected statistic over the recording period.

WLM Report Browser

Upon startup, the Report Browser displays. The browser shows a collection of reports. The type of display, which is user configurable, is based off the properties chosen to generate the report.

Report Browser menu options:

New	Create report
Close	Exit browser
Open	Display a selected report
Properties	Allow the properties of a report to be viewed and edited
Delete	Delete a selected report

Report Properties Panel

The Report Properties Panel allows the user to define the attributes that control the actual graphical representation of the WLM data. There are three tabbed panes in this panel:

- General Menu
- Tier/Class Menu
- Advanced Menu

Report Name A user-editable field for naming the report. Reports should end with the **.rpt** extension

General Menu: The first tabbed pane allows the user to edit the general properties of a display as follows:

Trend Box Indicates that a trend report of the selected type will be generated. Trend reports allow the comparison of two different time periods on the same display. Selecting this box enables the **End of first interval** field for editing.

Resource Allows selections for the WLM resources to be displayed (such as CPU or memory). Refer to the WLM user's guide and documentation for information about the resources that can be managed.

Width of interval Represents the period of time covered by any display type measuring either from the latest values available in the recording, or from user-input time selections. Interval widths are selected from this menu. The available selections vary, depending upon the tool being used:

wlmmon
Multiple selections for minutes and hours

wlmperf
Multiple selections for minutes, hours, days, weeks, and months

End of first interval Represents the end time of a period of interest for generating a trend report. The first interval always represents a time period ending earlier than the last interval. This field can only be edited if the Trend Box is selected.

End of last interval Represents the end time of a period of interest for trend and non-trend reports The last interval always represents the latest time frame to be used in generating a display report. There are two exclusive selection options for this field:

Latest Uses the latest time available in the recording as the end time for the report.

Selected Time
Allows the user to input the end time of the last interval.

Tier/Class Menu: The second tabbed pane allows users to define the set of WLM tiers and classes to be included in a report.

Scope Allows the user to select a tier or class-based scope for the display. This display will vary, as tier and class concepts vary between the AIX releases (AIX 4.3 classes versus AIX 5.1 superclass and subclass definitions).

Selection Allows selection of including and excluding the WLM tiers or classes available in the recording.

Advanced Menu: The third tabbed pane of the Report Properties Panel provides advanced options, primarily for the snapshot display. For snapshots, exclusive methods for coloring the display are provided for user selection. Option 1 ignores the minimum and maximum settings defined in the configuration of the WLM environment. Option 2 uses the minimum and maximum settings.

Report Displays

There are three types of report displays:

- Snapshot Display
- Detailed Display
- Tabulation Display

Each of these displays has the following common elements:

WLM Console Selections for printing or closing the display.

Time Period Displays the time period defined in the Report Properties Panel. For trend reports comparing two time periods, two time displays are shown.

Tier Column Displays the tier number associated with a class. For AIX 5.1, the column has two entries, for superclass tier (left) and subclass tier (right).

Class Column Displays the class name.

Resource Columns Displays the resource information based off of the type of graphical report selection chosen. These are described below.

Status Area Displays a set of global system performance metrics that are also recorded to aid in analysis. The set displayed may vary between AIX releases, but will include metrics such as run, queue, swap queue, and CPU busy.

Snapshot Display: This display is a quick ″Am I OK?″ overview. The display focuses on showing class resource relationships based off user-specified variation from the defined target shares. To select or adjust the variation parameters for this display, use the Report Properties Panel Advanced Menu.

If the snapshot display is trended, the earlier (first) analysis period is shown by an arrow pointing from the earlier measurement to the later (second) measurement. If there has been no change between the periods, no arrow is shown.

Detailed Display: In this display, the resource columns are displayed in bar-graph style, along with the percentage of measured resource activity over the time period specified. The percentage is calculated based off the total system resources defined by the WLM subsystem. If the detailed display is trended, the later (second) measurement is shown above the earlier (first) measurement interval.

Tabulation Display: The third type of display report is a tabulation report. In this report, the following fields are provided:

Number Sampled	Number of recorded samples for this period
Share Value	Computed share value target by WLM
Mean Value	Calculated average over the sample period
Standard Deviation	Computed standard deviation
Defined Min	Class minimum defined in WLM limits
Observed Min	Actual observed minimum across time period
Defined Soft Max	Class soft maximum defined in WLM limits
Defined Hard Max	Class hard maximum defined in WLM limits
Observed Max	Actual observed minimum across time period

Daemon Recording and Configuration

The daemons create recordings in the **/etc/perf/wlm** directory. For the base AIX tool **wlmmon**, these recordings are limited to the last 24–hour period.

For the Performance Toolbox tool **wlmperf**, these recordings are limited to 1 year. For the PTX, the **xmtrend** daemon is used, and uses a configuration file for recording preferences. A sample of this configuration file for WLM— related recordings is located at **/usr/lpp/perfagent.server/xmtrend_wlm.cf**. Recording customization, startup, and operation are the same as those described for the **xmtrend** daemon.

For the base AIX, the **xmwlm** daemon is used and cannot be customized.

For recordings to be created, adequate disk allocations must be made for the **/etc/perf/wlm** directory, allowing at least 10 MB of disk space. Additionally, the daemon should be started from an **/etc/inittab** entry so that recordings can automatically restart after system reboots. The daemon will operate whether the WLM subsystem is in active, passive, or disabled (off) modes. However, recording activity is limited when WLM is off.

Prerequisites

Java™ 1.3 **perfagent.tools**

Exit Status

A warning message is issued by the tool if no WLM recordings are located.

Files

/usr/bin/wlmmon	base AIX
/usr/bin/xmwlm	base AIX
/usr/bin/wlmperf	Performance Toolbox
/usr/lpp/perfagent.server/xmtrend.cf	Performance Toolbox
wlmmon and **xmwlm**	Located in the **perfagent.tools** fileset.
wlmperf and **xmtrend**	Available only with the Performance Toolbox product media.

Related Information

The **wlmstat**, **wlmcntrl**, and **topas** commands.

---

wlmstat Command

Purpose

Shows Workload Manager (WLM) per class resource utilization statistics.

Syntax

wlmstat [**-l** *Class* | **-t** *Tier*] [**-S** | **-s**] [**-@**] [**-c**] [**-m**] [**-b**] [**-B** *Device*] [**-T**] [**-a**] [**-w**] [**-v**] [*Interval*] [*Count*]

wlmstat [**-l** *Class* | **-t** *Tier*] [**-@**] [**-c**] [**-m**] [**-b**] [**-u**] [*Interval*] [*Count*]

wlmstat [**-l** *Class* | **-t** *Tier*] [**-@**] [**-M**] [**-S** | **-s**] [**-w**] [**-v**] [*Interval*] [*Count*]

Description

The **wlmstat** command symbolically displays the contents of WLM data structures retrieved from the kernel. If a *Count* is specified, **wlmstat** loops *Count* times and sleeps *Interval* seconds after each block is displayed. If *Interval* and *Count* are not specified, one output report is produced. If *Interval* is specified but no *Count* is given, **wlmstat** outputs results continuously at the given interval until stopped by a signal (SIGINTR, SIGQUIT, and SIGKILL). By default, **wlmstat** displays the statistics for all the resources for every superclass and subclass. You can specify flags to narrow the focus of the statistics to a type of resource, tier, superclass, or subclass and alter the output format.

Note: The following should be considered when viewing the **wlmstat** output:

1. Starting with AIX 5.3, the WLM processor usage values and process priority adjustments are updated 10 times per second by default.
2. The value displayed for processor usage is not the current instantaneous usage from the last second, but is instead an average of the last *N* readings (starting with AIX 5.3, the default value for *N* is 15).
3. The Unmanaged class is used to report system interrupt time and for tracking memory usage for all of the pinned pages in the system that are not managed by the WLM. No processes are assigned to this class.

It is possible for a process with a hard limit of 50 percent to use more than 50 percent of the processor between two consecutive WLM usage updates. Each tenth of a second, every process is assigned a priority, and the scheduler then schedules all processes based on their assigned priorities. A process might receive more of the processor resources than the process hard limit between WLM updates.

By default, each instantaneous value of processor usage from each update is kept for the following 15 readings and is averaged with the other 14 readings before being displayed by **wlmstat**. This can potentially result in a value of greater than 50 percent due to a single instance of more than 50 percent usage between WLM updates.

The priority of a process will be greatly reduced and the process will be unable to run if the process consistently reaches or exceeds its hard limit. Over the long term, the resource utilization of the process must be at or under the process hard maximum. Over a short time interval, **wlmstat** may show the process using more than the process hard limit. The **/usr/samples/kernel/wlmtune** command that is available in the bos.adt.samples PTF can be used to modify the behavior of WLM in such an instance. The related tunables are:

schedhz
: The frequency at which the WLM scheduler recalculates class consumption and priority for processor. The default is 10. Modifying this value changes the responsiveness of WLM. Increasing this value causes WLM to update more frequently, thereby reducing the possibility of a process exceeding its hard limit during a short time interval. The trade-off for this is increased overhead, since more WLM processing occurs. This can potentially affect overall system performance.

cpuhist
: The number of consecutive processor consumption values used in the average calculation. The default is 15. Increasing this value further smooths the reported processor usage values by averaging over a longer period.

To make WLM more responsive so that classes do not exceed their maximums over long periods, it is recommended that you first try modifying **schedhz** until the **wlmstat** output displays the desired results. You may want to also modify **cpuhist** so that **wlmstat** averages over the same time interval. For example, if **schedhz** is 20 and **cpuhist** is 15, **wlmstat** will average over a period of 0.75 seconds (15/20), so you may want to change **cpuhist** to 30 so that **wlmstat** still averages over 1.5 seconds.

On systems with no contention for processor, an *Interval* of 5 for **wlmstat** is recommended in order to adhere to WLM limits.

Flags

-@	Displays workload partition resource information.
-a	Displays subclass consumption in absolute terms. By default, the subclass consumption percentages are shown relative to the superclass consumption. With this option, subclass consumption is displayed relative to the total amount of resource available on the system (as is done for superclasses). All values are displayed with 1% precision. For instance, if a superclass has a processor target of 20% and the processor percentage shown by **wlmstat** without **-a** for a subclass is 10%, **wlmstat** with **-a** shows the processor percentage for the subclass as 2%.
-b	Displays only disk I/O statistics.
-B *Device*	Displays disk I/O device statistics. Passing an empty string (-B ″″) displays the statistics for all the disks accessed by the class.
-c	Shows only processor statistics.
-l *Class*	Displays statistics for *Class* name. If not specified, all classes display along with a summary for appropriate fields.
-m	Shows only physical memory statistics.

-M Displays the Real/Virtual Memory statistics. Use of the **-M** option adds the following columns in the output:

RMSIZ Utilized real memory size for the class

VMSIZ Utilized virtual memory size for the class

RMLIM Real memory limit for the class

VMLIM Virtual memory limit for the class

LGPGSIZ
Utilized large pages in the class

LGPGLIM
Large page limit for the class

Note: A - will be displayed for the **RMLIM**, **VMLIM**, and **LGPGLIM** fields if the limit is undefined. When the **-M** and **-w** options are used together, **RMSIZ** and **VMSIZ** fields contain the high watermarks for these attributes instead of the actual utilized values. In addition, the **LGPGSIZ** and **LGPGLIM** fields is turned off.

-s Displays only subclass statistics.

-S Displays only superclasses statistics.

-t *Tier* Displays statistics only for the specified *Tier*.

-T Displays the total numbers for resource utilization since WLM was started or the class was created, whichever is the latter. The units are:

CPU The total processor time, in milliseconds, consumed by a class

MEM Unused

DKIO The total number of 512 byte blocks sent/received by a class for all the disk devices accessed.

-v Specifies verbose mode. This flag, intended for trouble shooting, also displays some class attributes, resource shares and limits and other WLM parameters, including internal parameter values intended for AIX support personnel. The following information can be of interest for users:

Column Header	Description
CLASS	Class name.
tr	tier number (0 to 9)
i	Value of the inheritance attribute: 0 = no, 1 = yes.
#pr	Number of processes in the class. If a class has no (0) process assigned to it, the values shown in the other columns might not be significant.
CPU	Processor utilization of the class (%).
MEM	Physical memory utilization of the class (%).
DKIO	Disk IO bandwidth utilization for the classs (%).
sha	Number of shares ('-' is represented as -1)
min	Resource minimum limit (%)
smx	Resource soft maximum limit (%)
hmx	Resource hard maximum limit (%)
des	(desired): percentage goal (target) calculated by WLM using the shares numbers (%)
npg	Number of memory pages owned by the class.

The other columns are for internal use only and bear no meaning for administrators and end users. This format is better used with a resource selector (**-c**, **-m**, or **-b**), otherwise the lines might be too long to fit into a line of a display terminal.

-w Displays the memory *high water mark*, that is the maximum number of pages that a class had in memory at any given time since WLM was started or the class was created (whichever happened last).

-u Displays per-tier and total unused resources.

Display

Results are tabulated, with the following fields:

Name	Class name
CPU	Percentage of total processor time consumed by the class.
MEM	Percentage of physical memory consumed by the class.
DKIO	Percentage of the disk IO bandwidth consumed by the class. This number is the average of the disk bandwidth on all the disk devices accessed by the class, and is usually not significant. For instance if a class consumes 80% of the bandwidth of one disk and 5% of the bandwidth of two other disks, the DKIO column shows 30%. For details on the per device utilization, use the -B device option.

Examples

1. To get a printout of WLM activity right now, enter:

   ```
   wlmstat
   ```

   This produces the following output:

   ```
                    CLASS CPU MEM DKIO
             Unclassified   0   0   0
                Unmanaged   0   0   0
   ```

```
                Default   0   0   0
                 Shared   0   0   0
                 System   0   0   0
                 class1  12   0   0
    class1.Default     4   0   0
     class1.Shared     0   0   0
  class1.subclass1     4   0   0
  class1.subclass2     4   0   0
                 class2  12   0   0
    class2.Default     4   0   0
     class2.Shared     0   0   0
  class2.subclass1     4   0   0
  class2.subclass2     4   0   0
```

2. To get a report for superclass **class1**, enter:

   ```
   wlmstat -l class1
   ```

   This produces the following output:

```
                  CLASS CPU MEM DKIO

                class1  12   0   0
    class1.Default     4   0   0
     class1.Shared     0   0   0
  class1.subclass1     4   0   0
  class1.subclass2     4   0   0
```

3. To get a report for subclass **sclass1.subclass2** updated every 10 seconds, for one minute, enter:

   ```
   wlmstat -l class1.subclass2 10 6
   ```

   This produces the following output:

```
                  CLASS CPU MEM DKIO
  class1.subclass2     4   0   0
  class1.subclass2     4   0   0
  class1.subclass2     4   0   0
  class1.subclass2     4   0   0
  class1.subclass2     4   0   0
  class1.subclass2     4   0   0
```

4. To display virtual/real memory statistics, enter:

   ```
   wlmstat -M
   ```

   This produces the following output:

```
CLASS          RMSIZ    RMLIM   VMSIZ    VMLIM  LGPGSIZ LGPGLIM
Unmanaged       1024     4096    4096     8192     0       -
Default         0          -      0       -       0       -
Shared          0          -      0       -       0       -
System          23567    50000  819234   1000000  0       -
```

5. To display the memory high water mark, enter:

   ```
   wlmstat -M -w
   ```

   This produces the following output:

```
CLASS      RMSIZ RMLIM  VMSIZ   VMLIM
Unmanaged  1024  4096   4096    8192
Default    0     -      0       -
Shared     0     -      0       -
System     23567 50000  819234  1000000
```

Errors

A warning message is issued by **wlmstat** if WLM is not started.

Related Information

The **wlmcntrl** command.

wol command

Purpose

Wakes up one or more hosts that are connected to a network in suspend mode by sending a Magic Packet.

Syntax

To send a Magic Packet to a subnet-directed broadcast address:

wol { [**-m** *MACAddress* [[**-h** *Host* **-s** *SubnetMask*] | **-i** *Interface*] | **-f** *File* } [**-v**]

To send a Magic Packet to a multicast address:

wol { **-m** *MACAddress* **-M** *MulticastAddress* [**-p** *Port*] [**-i** *Interface*] | **-f** *File*} [**-v**]

Description

The **wol** command wakes up one or more hosts that are connected to a network in suspend mode by sending a Magic Packet to the specified address or addresses on the specified subnet.

If the user doesn't specify either the **-h**, nor **-s** flag, the **wol** manager will broadcast the Magic Packet as follows:

- If the user specifies the interface name (**-i** *Interface*), the Magic Packet will be broadcast from the specified interface.
- If the user doesn't specify the interface name, then the **wol** manager will loop through each network interface installed on the machine. If an interface is up, it will broadcast the Magic Packet from that interface, and then continue to the next interface until it goes through the entire interface list on the machine.

The file specified with **-f** *File* contains the list of hosts which need to be awakened. This file consists of one or more lines, each line containing the following information in this format:

MacAddress; Hostname/IPaddress; SubnetMask; Multicast; Port; Interface

For example, the file might look like this:

```
00:20:35:7a:7:89a;        9.41.86.19;      255.255.255.0  ; ; ;
00:04:ac:17:c0:9f ;        obiwan.aoot.austin.ibm.com;       255.255.255. 224; ; ;
00:07:be:4a:2:394; ; ; ; ; en0
00:06:38:6b:7e:8f ;         ; ; 234.5.6.7;   12345 ;
```

A line starting with a ″#″ character is a comment and is ignored. Each line contains 6 tokens separated by ″;″ character. The MAC address is mandatory. The other tokens are optional, but the ″;″ character must be used to separate unused tokens.

Flags

-i *Interface*	Specifies the interface to use on the host where the **wol** command is being run
-f *File*	Specifies the name of a file containing a group list. This allows the user to wake a specified group of hosts.
-h *Host*	Specifies a host to wake, either as a hostname or as an IPv4 address in dot string representation (for example, 10.0.0.3).
-m *MACAddress*	Specifies the a 48 bits MAC address of the host in hex representation (for example, 00:20:35:7a:78:9a).

-M *MulticastAddress*	Specifies an IPv4 multicast address.
-p *Port*	Specifies the port to use on the multicast machine.
-s *SubnetMask*	Specifies an IPv4 subnet mask in dot string representation (for example, 255.255.255.0).
-v	Specifies verbose mode.

Exit Status

0	The command completed successfully.
>0	An error occurred.

Security

RBAC Disclaimer

Location

/usr/sbin/wol

Related information

RBAC in *Security*.

Trusted AIX in *Security*.

---

wparexec Command

Purpose

Creates an application workload partition or specification file.

Syntax

wparexec [**-a**] [**-c** [**-1**]][**-F**] [**-h** *hostname*] [**-i**] [**-l** *attribute=value* ...] ... [**-M** *attribute=value*...] ... [**-N** *attribute=value*...] ... [**-R** *attribute=value*...] [**-u** *userscript*] { **-n** *wparname* [**-e** *existingwpar* | **-f** *infile*] [**-o** *outfile* [**-w**]] | **-f** *infile* [**-n** *wparname*] [**-o** *outfile* [**-w**]] | **-w -o** *outfile* [**-n** *wparname*] [**-e** *existingwpar* | **-f** *infile*] } [[--] [*var=value* ...] */path/to/command* [*arg* ...]]

Description

The **wparexec** command builds and starts an application workload partition, or creates a specification file to simplify the creation of future application workload partitions.

An application workload partition is an isolated execution environment that might have its own network configuration and resource control profile. Although the partition shares the global file system space, the processes running therein are only visible to other processes in the same partition. This isolated environment allows process monitoring, gathering of resource, accounting, and auditing data for a predetermined cluster of applications.

The **wparexec** command invokes and monitors a single application within this isolated environment. The **wparexec** command returns synchronously with the return code of this tracked process only when all of the processes in the workload partition terminate. For example, if the tracked process creates a daemon and exits with the `0` return code, the **wparexec** command will block until the daemon and all of its children terminate, and then exit with the `0` return code, regardless of the return code of the daemon or its children.

Flags

-1 Creates the configuration only. Causes the **wparexec** command to stop after creating the configuration of the application WPAR. The **startwpar** command must then be used to start the WPAR. Only advanced users can use the **-1** option.

-a Automatically resolves conflicting static settings if required. Resolvable settings are name, hostname, and network configuration.

-c Enables this workload partition to be checkpointed. This option is only valid when additional checkpoint-restart software is installed and configured. When this option is used, any file systems associated with this workload partition (for example, with the **-M** option) must be remote (for example, **vfs**=**nfs**).

-e *existingwpar* Uses an existing application workload partition as the source for specification data. Do not use the **-e** flag with the **-f** flag. Any values specified by other **wparexec** flags override those values from the existing workload partition.

-f *infile* Indicates the specification file to read default values from. Do not use with the **-e** flag. Any values specified by other **wparexec** flags override those values from the loaded specification file.

-F Suppresses or overrides most error conditions. With the **-F** flag, the **wparexec** command continues with a warning.

-i Enables WPAR-specific routing for the workload partition. By default, outgoing network traffic from a workload partition is routed like it is being sent from the global environment, notably in the following ways:

- Traffic between addresses that were hosted on the same global system is sent through the loopback interface.
- Routing table entries that are configured in the global system, including the default route, are used to transmit workload partition traffic.

If you enable WPAR-specific routing by specifying the **-i** flag, the workload partition creates and uses its own routing table for outgoing traffic. Routing entries are created automatically for each of the network addresses of the workload partition to accommodate broadcast, loopback, and subnet routes. For more information about the network attributes, see the **-N** flag. You can create explicit additions to the routing table of the workload partition using the **-l** flag. In particular, you can use the **-l** flag to configure the default route, as no default route is created automatically.

-l *attribute*=*value* ... Adds routing table entries to those that are automatically created when WPAR-specific routing is in effect. You can specify more than one **-l** flag to configure multiple routes. Using the **-l** flag automatically enables WPAR-specific routing as described under the **-i** flag.

You can specify the following attributes with the **-l** flag. The **rtdest** attribute and the **rtgateway** attribute are required to be specified.

rtdest=*destination* Identifies the host or network to which you are directing the route. You can specify the value using either a symbolic name or a numeric address. You can use the keyword **default** to specify a default route. For more information about the route **rtdest** attribute, see the *Destination* parameter of the **route** command.

rtgateway=*gateway* Identifies the gateway to which packets are addressed. You can specify the value using either a symbolic name or a numeric address.

rtnetmask=*A.B.C.D* Specifies the network mask to the destination address.

rtprefixlen=*n* Specifies the length of a destination prefix, which is the number of bits in the netmask. The value must be a positive integer.

rttype={*net*|*host*} Forces the **rtdest** attribute to be interpreted as the specified type.

rtinterface=*if* Specifies the interface, for example, `en0`, to associate with the route so that packets are sent using the interface when the route is chosen.

-h *hostname*
Specifies a host name for this workload partition. If not specified, the **wparexec** command uses the workload partition name as host name.

-M directory=*dir*
[**vfs**=*type*]
[**dev**=*devicepath*]
[**host**=*remotehost*]
[**mountopts**=*mountopts*]
Specifies file system dependencies only. Attributes must be space-separated. By default, an application workload partition has the same level of access to all of the global file systems and mounts as the user who created the workload partition. Use the **-M** flag with the directory attribute that is set to the file system name to specify additional file systems. More than one **-M** flag can be specified.
Note: All of the mounts and all of the directories are created and available at global level. File systems that are based on disk, such as the **vfs**=*jfs* and the **vfs**=*jfs2*, will not be created for application workload partitions.

A local file system dependency can be added by defining only the directory attribute. However, the directory specified must already exist in the **/etc/filesystems**.

If an error occurs during the process of creating the workload partition, any file systems mounted by the **wparexec** command are unmounted. After the creation succeeds, the file systems are not unmounted, regardless of the return status of the user application.

The following are the valid values for the *vfs* attribute for application workload partitions:

nfs The directory specified by the **dev** attribute on the system specified by the host attribute is mounted at the location that is specified by the directory attribute. If the mount point does not exist, it will be created. The only other attributes that are applicable to an **nfs** mount are the **mountopt** attributes, corresponding to the **-o** option of the **mount** command or the **options** attribute in an **/etc/filesystems** stanza. If not specified, no mount options are used by default. Acceptable option values correspond to the **-o** options to the **mount** command. For more information, see the mount Command in *AIX 5L Version 5.3 Commands Reference, Volume 3*.

namefs
The global directory specified by the **dev** attribute is mounted over the directory specified by the **directory** attribute.

The only other attributes that are applicable to a **namefs** mount are the **mountopt** attributes.

directory
The global directory specified by the **directory** attribute is created if it does not exist. No mounting is performed.

-n *wparname*
Specifies the name for the workload partition to be created. If no name is supplied, a name is generated.

-N *attribute*=*value*

Allows specification of the following network configuration attributes:

- **interface**= *if* or **interface**=*namemappedif*
- **address**=*A.B.C.D*
- **netmask**=*A.B.C.D*
- **broadcast**=*A.B.C.D*
- **address6**=*S*:*T*:*U*:*V*:*W*:*X*:*Y*:*Z*
- **prefixlen**=*n*

The name-mapped interface is defined in the **/etc/wpars/devmap** file. You can specify the mapping between the name-mapped interface and the system interface as follows:

```
# The comments start with '#'
# Each line contains a pair of name-mapped interface
# and real interface separated by tab or blank spaces.
foo en0
goo en1
soo en2
```

The *attribute*=*value* pairs must be separated by spaces. More than one **-N** flag can be used to configure multiple IP addresses. At minimum, the **address** or the **address6** attribute must be specified. The **wparexec** command collects any other values that are not specified from the global system's settings. If no **-N** flag is specified, the **wparexec** command will attempt to discover an appropriate IP address for the workload partition by running the **gethostbyname** subroutine on the workload partition name specified with the **-n** flag. If an address is found on the same subnet as any global interface, the settings of that interface will be used with the resolved IP address to create the default network entry.

To define an IPv6 network configuration, specify the **-N** flag with the **address6** attribute, the **prefixlen** attribute, and the **interface** attribute:

- The **address6** attribute is a 128-bit address. The address is represented by eight 16-bit integers that are separated by colons. Each integer is represented by four hex digits. Leading zeros can be skipped, and consecutive null 16-bit integers can be replaced by two colons (one time per address).
- The **prefixlen** attribute is the number of high-order bits that are used to mask the IPv6 address and to comprise the prefix. The value of the **prefixlen** attribute ranges from 0 through 128. Each **-N** flag can accept either IPv4 attributes, or IPv6 attributes, but not both.

-o *outfile*

Indicates an output path and file name to write specification data to. This specification file can be used to create an application workload partition later with the **-f** flag.

-R *attribute=value*

Allows specification of resource control attributes. Only one **-R** flag can be specified. Most resource controls are similar to those supported by the Workload Manager (WLM). See the listed WLM pages for descriptions of these attributes. Valid attributes are as follows:

active Allows resource control definitions to be retained, but rendered inactive. This attribute can take the *yes* or *no* values.

rset Configures this workload partition to use a resource set created by the **mkrset** command. For more information about the **mkrset** command, see mkrset Command in *AIX 5L Version 5.3 Commands Reference, Volume 3*.

shares_CPU
The number of processor shares available to this workload partition. For more information about processor shares, see Workload Manager shares File in in *AIX 5L Version 5.3 Files Reference*.

CPU The percentage of processor limits for this workload partition's processes. This attribute uses the following format to definite the limits values:

```
CPU=<m>%-<SM>%,<HM>%
```

The *m* value represents the minimum limit. The *SM* value represents the soft maximum limit. The *HM* value represents the hard maximum limit. For more information about limited values, see the Workload Manager limits File in in *AIX 5L Version 5.3 Files Reference*.

shares_memory
The number of memory shares available to this workload partition. For more information about memory shares, see Workload Manager shares File in in *AIX 5L Version 5.3 Files Reference*.

memory
The percentage of memory limits for this workload partition's processes. For more information about memory limit, see the Workload Manager limits File in in *AIX 5L Version 5.3 Files Reference*.

procVirtMem
The maximum amount of virtual memory that a single process can consume. Processes that exceed the specified limit are terminated. The valid units are megabytes (M or MB), gigabytes (G or GB), and terabytes (T or TB). The minimum limit allowed is 1M. The maximum limit that can be specified is 8796093022207M, 8589934591G, or 8388607T. If the value is set to -1 (no units), the limit is disabled. For more information about limit values, see Workload Manager limits File in in *AIX 5L Version 5.3 Files Reference*.

totalVirtMem
The maximum amount of virtual memory that can be consumed by the WPAR as a whole. Processes that cause the specified limit to be exceeded will be terminated. The valid range and units are the same as for procVirtMem. If the value is set to '-1' (no units), the limit is disabled. See Workload Manager limits File in *AIX 5L Version 5.3 Files Reference*.

totalProcesses
The total number of processes that are allowed in this workload partition. For more information about allowed processor number, see Workload Manager limits File in in *AIX 5L Version 5.3 Files Reference*.

(*Attributes for the* **-R** *flag*, *continued*):

totalPTYs=*n*
The total number of pseudo terminals that are allowed in the workload partition. For more information about the allowed pseudo terminals, see pty Special File.

totalLargePages=*n*
The number of large pages that can be allowed for the workload partition. For more information about the allowed large pages, see Large Pages.

pct_msgIDs=*n***%**
The percentage of the maximum number of message queue IDs of the system that are allowed in the workload partition. For more information about the allowed number of message queue IDs, see Message Queue Kernel Services.

pct_semIDs=*n***%**
The percentage of the maximum number of semaphore IDs of the system that are allowed in the workload partition.

pct_shmIDs=*n***%**
The percentage of the maximum number of shared memory IDs of the system that are allowed in the workload partition. For more information about the allowed number of shared memory IDs, see Shared Memory.

pct_pinMem=*n***%**
The percentage of the maximum pinned memory of the system that can be allocated to the workload partition. For more information about pinned memory, see Support for pinned memory.

totalThreads
The total number of threads that are allowed in this workload partition. For more information about allowed number of threads, see Workload Manager limits File in in *AIX 5L Version 5.3 Files Reference*.

-u *userscript*
Specifies the path to a user script to be run by the workload partition commands at various administration points. The parameter of the **-u** flag can be a quoted string including additional arguments to be passed to the script. In all cases, the first component of the parameter of the **-u** flag must be an absolute path to an existing executable file. The script is invoked as follows:

/path/to/userScript<*action*><*WPAR*>

The *action* argument indicates the administrative action being performed, as follows:

- **WPAR_LOAD**: A script runs in the global environment after kernel configuration, before the tracked process is created. If the script returns a value of non-zero, the workload partition will not be started.
- **WPAR_START**: A script runs in the global environment after the workload partition becomes active. For application workload partitions, the script runs once the tracked process is started.
  Note: This code path can be run asynchronously by a dissociated process with its standard I/O streams closed or redirected. Internal messaging must be handled accordingly, and the script must account for the fact that short-lived workload partitions might be stopped or stopping at any point during the execution of the script.
  If the script returns a value of nonzero, a warning is logged, but no other behavior changes.
- **WPAR_STOP**: A script runs in the global environment after all of the workload partition processes finish, and before the kernel is unconfigured.

Note: This code path can be executed by a dissociated process with its standard I/O streams closed. Internal messaging should be handled accordingly.

If the script returns a value of non-zero, a warning will be logged, but no other behavior will change.

-w Writes the specification file. When it is used with the **-o** flag, the **-w** flag causes the **wparexec** command to quit after writing the new specification file without actually creating the workload partition.

[--] [*var*=*value*] */path/to/command* [arg ...] Specifies the application (tracked process) to be run within the workload partition, along with any necessary environment variable settings and arguments.

The command is required, either by this command line syntax or the **general.application** attribute in the specification file, unless the command is only creating a specification file (with **-w** flag). When it is invoked, the command line provided is always shell-expanded within the workload partition. When using the command line, shell metacharacters should be escaped appropriately to prevent premature expansion.

The special double-minus separator (--) is used to signify that all subsequent command line arguments comprise the tracked process. For example, use this separator to remove ambiguity between attributes to the **-N** flag and assignment of environment variables to the tracked process.

Only one tracked process per workload partition is supported, but this application might create other processes. The workload partition is automatically stopped and removed when all of the processes therein terminate. A workload partition might be stopped and removed prematurely by the **stopwpar** command.

Security

Access Control: Only the root user can run this command.

RBAC Disclaimer

Examples

1. To create an application workload partition that is running a benchmark program, enter:
   ```
   wparexec -n tpcc -N address=192.168.0.51 /u/tpcc/benchmark -f /tmp/logfile
   ```
   Note: The **-f** flag is passed to the **/u/tpcc/benchmark** file and is not processed by the **wparexec** command as a flag.
2. To create a workload partition based on an existing specification file, enter:
   ```
   wparexec -f /tmp/wparexec1.spec
   ```
3. To override the default minimal *PATH* variable provided by the **wparexec** command, enter:
   ```
   wparexec PATH=/usr/opt/bin:/usr/bin:/usr/sbin /home/joe/runapp
   ```

Files

/usr/samples/wpars/sample.spec An annotated workload partition specification file.

Related Information

The **chwpar**, **clogin**, **lswpar**, **mkwpar**, **rc.wpars**, **rebootwpar**, **rmwpar**, **route**, **startwpar**, **stopwpar**, **syncwpar**, and the **syncroot** commands.

The **devexports File**, **Workload Manager Shares File** and the **Workload Manager limits File** in *AIX 5L Version 5.3 Files Reference*.

RBAC in *Security*.

Trusted AIX in *Security*.

write Command

Purpose

Opens a line of communication to send messages to other users on the system in real time.

Syntax

To query all messages awaiting replies from users on a host and display them with their handles, type the following:

write -q [**-n** *Host*]

To Reply to a Message Sent by a Utility or a Shell Script, or Redisplay the Message Associated with a Given handle, type the following:

write -h*Handle,* { ok | cancel | query } [**-n** *Host*]

To send messages to a user, optionally on another host or a particular device, type the following:

write [**-r**] { [**-n** *Host*] *User* | *User@Host* } [*Line*]

Description

The **write** command enables message sending over the system in real time. It provides conversation-like communication with another logged-in user. Each user alternately sends and receives short messages from the other workstation. Long messages can be sent by putting the complete message in a file and then redirecting that file as input to the **write** command.

For another user (as specified by the *User* parameter) to receive a message, that user must be logged in and must not have refused message permission. When a message is sent to a user who is not logged in, the message `user not logged in` appears. If the message is sent to a user who has refused message permission by setting the **mesg** command to no, the message `write: permission denied` appears.

When the **write** command is issued, it immediately sends the following message, along with an attention-getting sound (the ASCII BEL character) to the message recipient or target:

```
Message from SenderID on SenderHostname (ttynn) [Date] ...
```

With a successful connection, the **write** command sends two ASCII BEL characters to both workstations. The beep alerts the sender that the message can begin and it alerts the receiving user that a message is coming.

Sending occurs one line at a time as the Enter key is pressed. The communication link from the sender to the receiver remains open and sending continues until the Ctrl-D key sequence ends the sending link. Then an end-of-text character (<EOT>) is sent to the target workstation and the **write** command mode is terminated.

The receiving or target user can respond by sending a **write** command to the originating user. This opens a line of communication from the receiver back to the sender, enabling message responses in return. For this type of exchange, the following convention is useful: When you first write to others, wait for a response before sending any text. End a message with a signal such as `o` (over) to alert the other person to reply. Use `oo` (over and out) when the conversation is finished.

If the character `!` (exclamation point) is found at the beginning of a line, the **write** command calls the shell to execute the rest of the line as a command.

When you write to a user who is logged in at more than one workstation or multi-using more than one process, the **write** command uses the first login instance found in the **/etc/utmp** file as the message delivery point (usually the login or console shell), and you get the message:

```
UserID is logged on more than one place.
You are connected to "Workstation".
Other locations are:
Workstation
```

When this message is received, if you wish to send the message to a location other than the initial login location, the target user can be contacted at a different location by specifying the *Line* of the location (tty00, for example).

Permission to write to another user is granted or denied by the individual user with the **mesg** command. Some commands deny message permission while they are running to prevent interference with their output. A user with root user authority can write to any workstation regardless of the workstation's message permission.

You can use the **write** command to converse with users on other hosts. You can identify a user on a remote host by using the **-n***HostName* flag or the *User*@*Host* parameter. In order to write to a user on a remote host, the **writesrv** daemon must be running on both the current host and the remote host.

The **write** command is also used by the **qdaemon** daemon to send messages to users on other hosts and to wait for replies. There are only three valid replies:

`ok`	The original write exits with a status of 0.
`cancel`	The original write exits with a status of 1.
`query`	The message associated with the given handle is displayed.

Parameters

User	Specifies the user ID of the person to receive the message text.
User@*Host*	Specifies the user ID and remote host of the person to receive the message text.
Line	Contacts the target user at another location (`tty00`, for example).

Flags

-h *Handle*,*Reply*	Replies to a message sent by a utility or shell script using write with the reply option. The value to be used for the *Handle* variable is generated internally and supplied to the user in the text of the original message. The reply can be `ok`, `cancel`, or `query`.
-n*Host*	Specifies a remote host. The *Host* variable may be a nickname or an internet address.
-q	Queries all messages awaiting replies from users on a host and displays them with their handles.
-r	Generates a message handle, places it in the message header, sends the message, and waits for a reply. This flag is used by the **qdaemon** daemon for operator messages and can be put in shell scripts. It is not used for interactive conversations. An exit status of 0 indicates that the reply was `ok`, a status of 1 indicates that the reply was `cancel`, and an exit status of 2 indicates that the user could not be contacted.

Requirements:

- The **writesrv** daemon must be running on the target host in order for any of the flags to work. If you are not using either the **-n** flag or **@***Host*, but using **-h**, **-q**, or **-r**, the **writesrv** daemon must be running on your host.
- If TCP/IP is not installed on your machine but the *HostName* is set, in order to converse with users on the local host using the **write** command with the **-h**, **-q**, or **-r**

flag, you must append your host name to the end of the `loopback` entry in the **/etc/hosts** file. The original entry should read:

```
127.0.0.1 loopback LocalHostName
```

The new entry should read:

```
127.0.0.1 loopback LocalHostName HostName
```

Exit Status

This command returns the following exit values:

0 Successful completion.
>0 The addressed user either is not logged on or denies permission.

Examples

1. To write a message to a user who is logged in, enter:

   ```
   write june
   ```

   Press the Enter key and type,

   ```
   I need to see you! Meet me in the computer room at 12:30.
   ```

   Then press the Ctrl-D key sequence to terminate the **write** command mode.

   If your user ID is `karen` and you are using workstation `tty3`, `june`'s workstation displays:

   ```
   Message from karen on trek tty3 Aug 17 11:55:24  ...
   I need to see you!  Meet me in the computer room at 12:30.
   <EOT>
   ```

2. To hold a conversation, enter:

   ```
   write june
   ```

   Press the Enter key and type,

   ```
   Meet me in the computer room at 12:30.
   o
   ```

   This starts the conversation. The `o` at the beginning of the next line means the message is over. It tells June that you are waiting for a response. Do not press Ctrl-D if you wish to continue.

   Now June replies by typing:

   ```
   write karen
   ```

   Presses the Enter key and types,

   ```
   I'm running tests at 12:30. Can we meet at 3?
   o
   ```

   And you might respond:

   ```
   OK--the computer room at 3.
   oo
   ```

   The `oo` means *over and out*, telling June that you have nothing more to say. If June is also finished `oo`, then you both press Ctrl-D to end the conversation.

3. To write someone a prepared message, enter:

   ```
   write june < message.text
   ```

   This writes the contents of the **message.text** file to `june`'s workstation.

4. To write to the person using a certain workstation, enter:

```
write -n console
```

Press the Enter key and type,

```
The printer in building 998 has jammed.
Please send help.
```

Then press the Ctrl-D key sequence.

This writes the message to the person logged in at the workstation `/dev/console`.

5. To send a message to user `spuds` at host `partya`, enter:

```
write -n partya spuds
```

Press the Enter key and type,

```
Your new tape has just arrived,
come see me to pick it up.
Thanks!
```

Then press the Ctrl-D key sequence.

OR

```
write spuds@partya
```

Press the Enter key and type,

```
Your new tape has just arrived,
come see me to pick it up.
Thanks!
```

Then press the Ctrl-D key sequence.

6. Here is an example of a message sent by the **qdaemon** daemon:

```
Message from mary on trek (tty10) Aug 17 10:03:34 ...
Use "write -h 6398492,reply" to reply
Please insert tape number 5 into rmt0.
<EOT>
```

To reply in the affirmative, enter:

```
write -h 6398492,ok
```

Then press the Ctrl-D key sequence.

To reply in the negative, enter:

```
write -h 6398492,cancel
```

Then press the Ctrl-D key sequence.

With the **-h** flag, there is no need to supply the host name or user ID. This information is tracked with the handle.

Files

/etc/hosts	Contains TCP/IP host information.
/etc/utmp	Contains user and accounting information for the **who**, **write**, and **login** commands.

Related Information

The **mesg** command, **wall** command, **who** command, **writesrv** command.

Shells in *Operating system and device management*.

writesrv Daemon

Purpose
Allows users to send messages to and receive messages from a remote system.

Syntax
writesrv

Description
The **writesrv** daemon allows users to send messages to users on a remote system and receive responses from users on a remote system with the **write** command.

The **writesrv** utility receives incoming requests from a **write** command and creates a server process to handle the request. This server process communicates with the client process (**write**) and provides whatever services are requested.

To perform these services, the **writesrv** daemon creates a socket that is attached to the port defined in the **/etc/services** file. All requests for service are sent as messages to this socket.

> **Note:** If the **writesrv** daemon terminates abnormally (such as a system crash, power failure, or the **kill -9** command), the **/var/spool/writesrv** directory must be manually cleaned out to remove any files left behind by the **writesrv** daemon.

Examples
1. To start the **writesrv** daemon from the **/etc/rc** script, enter:
   ```
   /usr/sbin/writesrv
   ```
   The **writesrv** daemon is started from the **/etc/rc** script. This is the usual way the daemon is started.
2. To start the **writesrv** daemon using the System Resource Controller (SRC), enter:
   ```
   startsrc -s writesrv &
   ```
   The **writesrv** daemon is started using SRC.

Files
/etc/services	Contains the Network Services directory.

Related Information
The **kill** command, **write** command

Printing administration, and Remote Printing Overview in *Printers and printing*.

System Resource Controller in *Operating system and device management*.

wsm Command

Purpose
Starts a Web-based System Manager client session.

Syntax

/usr/websm/bin/wsm -host *managing host*

/usr/websm/bin/wsm -lang *language*

/usr/websm/bin/wsm -port *port number*

/usr/websm/bin/wsm -profile *pathname of preference file*

/usr/websm/bin/wsm -user *username*

/usr/websm/bin/wsm -DdefaultTurners=*value*

/usr/websm/bin/wsm -DdrawTreeLine=*value*

/usr/websm/bin/wsm -Ddatadir=*path*

Description

The **wsm** command is used to start a Web-based System Manager client session.

Note: The full pathname of this command, **/usr/websm/bin/wsm**, must be specified.

Flags

-host *managing host*	Forces Web-based System Manager to initially connect to the specified host. Even though you can easily manage other hosts while running Web-based System Manager, this option allows you to start Web-based System Manager with the preferences you set up on the specified host machine.
-lang *language*	Specifies language in which messages are displayed. If the **sysmgt.msg.***Language***.websm.apps** fileset is not installed, messages will be displayed in English.
-port *port number*	Causes Web-based System Manager to connect to any other hosts using the specified port. This port number used must match the port number on the managed machines for the **wsmserver** service specified in the **/etc/services** file.
-profile *pathname of preference file*	Specifies an alternate preference file. The default preference file will be a file named **WebSM.pref** found in the user's home directory. Using this option enables the user to use a different preference file. This can be useful if the user manages different sets of machines for different clients. **Note:** The preference file is read from either the local machine, or from the machine specified in the **-host** argument.
-user *username*	Causes Web-based System Manager to run as the given user name. You will be prompted for the user's password.
-DdefaultTurners=*value*	When the value is `true`, Java Look and Feel turners are used instead of Windows® turners for parent tree nodes in the Navigation Area and the Contents Area. No angled lines are drawn between tree objects.
-DdrawTreeLine=*value*	When value is `true` and **-DdefaultTurners**=`true`, causes angled lines to be drawn between tree objects in the Navigation Area and the Contents Area.
-Ddatadir=*path*	Specifies an alternate directory to look for configuration files normally found in **/var/websm/config/user_settings**.

Examples

1. To specify an alternate preference filer, enter:

   ```
   /usr/websm/bin/wsm -profile pathname of preference file
   ```

2. To specify an alternate configuration file, enter:

```
/usr/websm/bin/wsm -Ddatadir=pathname
```

Related Information

For information on installing the Web-based System Manager, see Chapter 2: Installation and System Requirements in *AIX 5L Version 5.3 Web-based System Manager Administration Guide*.

---

wsmaccess Command

Purpose

Wrapper around **wsm** command to enable Accessibility features.

Syntax

/usr/websm/bin/wsmaccess -host *managing host*

/usr/websm/bin/wsmaccess -lang *language*

/usr/websm/bin/wsmaccess -port *port number*

/usr/websm/bin/wsmaccess -profile *pathname of preference file*

/usr/websm/bin/wsmaccess -user *username*

/usr/websm/bin/wsmaccess -DdefaultTurners=*value*

/usr/websm/bin/wsmaccess -DdrawTreeLine=*value*

/usr/websm/bin/wsmaccess -Ddatadir=*path*

Description

Wrapper around **wsm** command to enable Accessibility features.

Note: The full pathname of this command, **/usr/websm/bin/wsmaccess**, must be specified.

Flags

-host *managing host*	Forces Web-based System Manager to initially connect to the specified host. Even though you can easily manage other hosts while running Web-based System Manager, this option allows you to start Web-based System Manager with the preferences you set up on the specified host machine.
-lang *language*	Specifies language in which messages are displayed. If the **sysmgt.msg.***Language***.websm.apps** fileset is not installed, messages will be displayed in English.
-port *port number*	Causes Web-based System Manager to connect to any other hosts using the specified port. This port number used must match the port number on the managed machines for the **wsmserver** service specified in the **/etc/services** file.
-profile *pathname of preference file*	Specifies an alternate preference file. The default preference file will be a file named **WebSM.pref** found in the user's home directory. Using this option enables the user to use a different preference file. This can be useful if the user manages different sets of machines for different clients. **Note:** The preference file is read from either the local machine, or from the machine specified in the **-host** argument.
-user *username*	Causes Web-based System Manager to run as the given user name. You will be prompted for the user's password.

-DdefaultTurners=*value* When the value is `true`, Java Look and Feel turners are used instead of Windows turners for parent tree nodes in the Navigation Area and the Contents Area. No angled lines are drawn between tree objects.

-DdrawTreeLine=*value* When value is `true` and **-DdefaultTurners**=`true`, causes angled lines to be drawn between tree objects in the Navigation Area and the Contents Area.

-Ddatadir=*path* Specifies an alternate directory to look for configuration files normally found in **/var/websm/config/user_settings**.

Examples

1. To specify an alternate preference filer, enter:

   ```
   /usr/websm/bin/wsmaccess -profile pathname of preference file
   ```

2. To specifiy an alternate configuration file, enter:

   ```
   /usr/websm/bin/wsmaccess -Ddatadir=pathname
   ```

Related Information

For information on installing the Web-based System Manager, see Chapter 2: Installation and System Requirements in *AIX 5L Version 5.3 Web-based System Manager Administration Guide*.

---

wsmserver Command

Purpose

Configures the functionality of the Web-based System Manager servers.

Syntax

/usr/websm/bin/wsmserver -enable

/usr/websm/bin/wsmserver -disable

/usr/websm/bin/wsmserver -start

/usr/websm/bin/wsmserver -enablehttps [*port_number*]

/usr/websm/bin/wsmserver -disablehttps

/usr/websm/bin/wsmserver -starthttps

/usr/websm/bin/wsmserver -sslalways

/usr/websm/bin/wsmserver -ssloptional

Description

The **wsmserver** command is used to control the server processes used by the Web-based System Manager. The servers are used to enable applet and client-server modes of execution. In addition, if the security functionality is installed, the SMGate utility can be configured.

Note: The full pathname of this command, **/usr/websm/bin/wsmserver**, must be specified.

Flags

-enable Enables the applet and client-server modes.

-disable Disable the applet and client-server modes

-start Start a session of the Web-based System Manager server.

This is normally only used by inetd.

The following flags can only be used if the security functionality has been installed:

-enablehttps [*port_number*] Starts the SMGate utility. An optional *port_number* for the SMGate server can be specified. If specified, the SMGate server listens on that port instead of the default of 9092.

-disablehttps Disables the SMGate utility.

-starthttps Starts the SMGate utility. This is normally started by the **init** process.

-sslalways Allows only secure connections. This flag is for a system with security configured.

-ssloptional Allows both secure and non-secure connections to the Web-based System Manager.

Examples

1. To enable Web-based System Manager for applet and client-server mode, enter:
   ```
   /usr/websm/bin/wsmserver -enable
   ```
2. To enable the SMGate utility, enter:
   ```
   /usr/websm/bin/wsmserver -enablehttps
   ```

Related Information

For information on installing the Web-based System Manager, see Chapter 2: Installation and System Requirements in *AIX 5L Version 5.3 Web-based System Manager Administration Guide*.

---

wtmpfix Command

Purpose

Manipulates connect-time accounting records by correcting date and time stamp inconsistencies.

Syntax

/usr/sbin/acct/wtmpfix [*File ...*]

Description

The **wtmpfix** command is called by the **runacct** procedure to examine standard input or *File*s that contain records in **wtmp** format, and correct problems that could make the **acctcon1** or **acctcon2** commands fail. The **wtmpfix** command corrects date and time stamp inconsistencies, and writes the corrected records to standard output. If the date and time stamps are not consistent when the **acctcon1** command runs, the **acctcon1** command generates an error and stops.

The **wtmpfix** command also checks the validity of the name field to ensure that it consists only of alphanumeric characters, a $ (dollar sign), or spaces. If the name is invalid, the **wtmpfix** command changes the login name to **INVALID** and writes a diagnostic message to standard error. In this way, the **wtmpfix** command reduces the chance that the **acctcon2** command will fail.

Each time the date is set (on system startup or with the **date** command), a pair of date change records is written to the **/var/adm/wtmp** file. The first record is the old date, denoted by the *old time* string. The *old time* string is placed in the line field and the **OLD_TIME** flag is placed in the type field. The second record

is the new date, denoted by the string *new time.* The *new time* string is placed in the line field and the **NEW_TIME** flag is placed in the type field. The **wtmpfix** command uses these records to synchronize all date and time stamps in the file.

Flags

None.

Parameters

File Specifies the file to examine that contains records in **wtmp** format.

Security

Access Control: These commands should grant execute (x) access only to members of the **adm** group.

Examples

1. To convert a binary record in **wtmp** format to an ASCII record called `dummy.file`, enter:

   ```
   /usr/sbin/acct/fwtmp < /var/adm/wtmp > dummy.file
   ```

   The content of a binary **wtmp** file is redirected to a dummy ASCII file.
2. To convert an ASCII `dummy.file` to a binary file in **wtmp** format called `/var/adm/wtmp`, enter the `fwtmp` command with the `-ic` switch:

   ```
   /usr/sbin/acct/fwtmp -ic < dummy.file  > /var/adm/wtmp
   ```

   The dummy ASCII file is redirected to a binary **wtmp** file.

Files

/usr/sbin/acct/wtmpfix	Contains the **wtmpfix** command.
/var/adm/wtmp	Contains records of date changes that include an old date and a new date.
/usr/include/utmp.h	Contains history records that include a reason, date, and time.

Related Information

The **acctcon1** or **acctcon2** command, **acctmerg** command, **acctwtmp** command, **fwtmp** command, **runacct** command.

System accounting in *Operating system and device management* describes the steps you must take to establish an accounting system.

See the Accounting commands in *Operating system and device management* for a list of accounting commands that can be run automatically or entered from the keyboard and about the preparation of daily and monthly reports, and the accounting files.

---

wump Command

Purpose

Starts the hunt the wumpus game.

Syntax

wump

Description

A wumpus is a creature living in a cave with many rooms interconnected by tunnels. You move among the rooms trying to shoot the wumpus with an arrow and trying to avoid being eaten by the wumpus or falling into bottomless pits. There are also super bats that may pick you up and drop you in some randomly selected room. For moving among the rooms and shooting arrows, the game prompts you with appropriate questions and follows your instructions. For example:

```
You are in room 14.
I feel a draft.
There are tunnels to 1 13 18.
Move or shoot? (m-s) m
Which room? 1
You are in room 1.
I feel a draft.
There are tunnels to 14 17 18.
Move or shoot? (m-s) m
Which room? 17
You are in room 17.
You fell into a pit!
Another game? (y-n)
```

In the above example, you start out in room 14. The computer displays `I feel a draft.` This is the hint that a pit is nearby. You choose to move to room 1. Again you are warned of the pit. You then choose to move to room 17 where you fall into a pit and die.

At the beginning of the game, you are prompted `Instructions? (y-n)`. Choosing `y` provides an explanation of the warnings, how to move, and how to shoot.

The game ends and you are prompted `Another game? (y-n)` if:

- You kill the wumpus.
- The wumpus eats you.
- You fall into a bottomless pit.
- You run out of arrows.

To quit the game at any time, press the interrupt (Ctrl-C) key sequence.

Files

/usr/games	Contains the location of the system's games.

Related Information

The **arithmetic** command, **back** command, **bj** command, **craps** command, **fish** command, **fortune** command, **hangman** command, **moo** command, **number** command, **quiz** command, **ttt** command, **turnoff** command, **turnon** command.

X Command

Purpose

Starts the X Server.

Syntax

X [**-a** *Number*] [**-auth** *FileName*] [**-bc** | **+bc**] [**-bp** *Color*] [**-broadcast**] [**-bs** | **-nobs**] [**-c** *Volume*] [**-cc** *VisualType* [**:***Display*]] [**-class** *DisplayClass*] [**-co** *File*] [**-cookie** *XDMAuthenticationBit*] [**-D** *File*] [**-d** *Depth* [**:***Display*]] **-displayID** *DisplayID*] [**-damage** | **+damage**] [**-evie** | **+evie**] [**-f** *Number*] [**-fc** *Font*] [**-fixes** | **+fixes**] [**-fn** *Font*] [**-fp** *Font*] [**-help**] [**-I**] [**-indirect** *HostName*] [**-layer** *#* [**:***Display*]] [**-logo** | **nologo**] [**-n :***Number*] [**-once**] [**-P** *RowColumn Display* }] [**-pbuffer** *level* [**:display** *name* | **:display** *number*]] [**-p** *Number*] [**-port** *PortNumber*] [**-query** *HostName*] [**-r** | **r**] [**+render** | **-render**][**-s** *Number*] [**-secIP** [*PermissionCode*]] [**-secLocal** [*PermissionCode*]] [**-secSMT** [*PermissionCode*]] [**-stereo** [**:***Display*]] [**-su**] [**-T**] [**-t** *Number*] [**-to** *Number*] [**-v**] [**-vfb**][**-wm**] [**-wp** *Color*] [**-wrap** | [**-wrapx**] [**-wrapy**]] [**-x** *ExtensionName*] [**-xkbdir** *Directory*] [**-xkbmap** *FileName*] [[**+**|**-**]**accessx**] [**-ar1** *Milliseconds*] [**-ar2** *Milliseconds*] [**-sp** *FileName*] [**+/- xinerama** *FileName*]

Description

The **X** command starts the X Server, a display server that runs on bitmapped terminals. The X Server distributes input and output requests to or from programs located on either the host system or systems connected to it through a network.

End an Enhanced X-Windows session by using the Ctrl+Alt+Backspace key sequence.

You can specify one or more display devices. If none are specified, the default is all. The default configuration order is determined by the adapter slot order. The adapter in the first slot is initialized as the left most screen, the adapter in the second slot is the next screen to the right. To rearrange the layout of the screens, use the **-P** flag. The **-P** flag associates the row and column of the device with the device name. You can determine the device name by using the **lsdisp** command.

The two displays are arranged either vertically or horizontally. The following example shows **-P** flags specifying a horizontal arrangement:

```
-P11 ppr0 -P12 ppr1
```

The 2 in the right position of the second **-P** flag indicates that the second monitor view is along the x-axis. This produces the horizontal arrangement:

```
        Display                       Display
                1                                2
```

To see two monitors in a vertical arrangement, the **-P** flags should read:

```
-P11 ppr0 -P21 ppr1
```

The 2 in the first position indicates that the monitors are in a vertical configuration along the y-axis:

```
        Display
                1
        Display
                2
```

In the horizontal configuration, when a mouse is traveling from left to right in Display 1 and reaches the border of Display 1 and 2, the cursor continues into Display 2 at the same y-axis position. When it reaches the edge of Display 2 and the **-wrapx** flag is set, it appears at the leftmost edge of Display 1 in the same y-axis position. If the **-wrapx** flag is not set, the mouse stops at the far edge of Display 2.

In a vertical configuration, when the mouse is traveling from top to bottom in Display 1 and reaches the border of Display 1 and Display 2, the cursor continues into Display 2 at the same x-axis position. When the cursor reaches the bottom of the display 2 and the **-wrapy** flag is set, the cursor appears at the top edge of Display 1 in the same x-axis position. If the **-wrapy** flag is not set, the mouse stops at the bottom of Display 2.

In addition, information and error messages (for example, a message indicating that an extension not able to load) are listed in the **/tmp/xlogfile** file. This file can provide useful information in cases when the X Server encounters a problem. This file is re-written every time the X Server is instantiated. This file provides additional error and non-error information but is not a complete error log for the X Server.

Flags

-a *Number*
Specifies the acceleration multiplier for mouse movement. For example, a value of 5 causes the cursor to move five times as fast as the mouse. The default is 4 pixels; any value specified must be a positive value greater than 0.

-auth *FileName*
Specifies to X the file from which to read the MIT (Massachusetts Institute of Technology) magic cookie.

-bc
Turns off backward compatibility with Enhanced X-Windows version 1.1.

+bc
Turns on backward compatibility with Enhanced X-Windows version 1.1. This is the default.

-bp *Color*
Specifies a black pixel color for the display. The default is display dependent.

-bs
Enables backing store support on all screens. Backing store support is disabled by default.

-c *Volume*
Specifies key click volume.

-cc *VisualType* [**:***Display*]
Specifies the type of visual to use for the root window of the screen specified by the display name. Not all visual types are available on all adapters at all depths. The **:***Display* parameter is optional, but useful when using the multihead option. The **:***Display* parameter is the name of the display as shown in the **lsdisp** command. If no display number or name is supplied, the specified visual is selected for all screens.

To specify the visual type and depth for the default visual, use the **-cc** and **-d** flags, respectively.

Values for the *VisualType* parameter are specified as a string or a number as follows:

String	Numeric equivalent
StaticGray	0
GrayScale	1
StaticColor	2
PseudoColor	3
TrueColor	4
DirectColor	5

-co *File*
Sets the name of the red, green, and blue (RGB) color database. This is the default flag for the color database.

-D *File*
Specifies the full path name of the color definition database file. The default is **/usr/lib/X11/rgb**.

-d *Depth*[**:***Display*]
Specifies the root depth for the screen specified by the display name. Not all visual types will be available on all adapters at all depths.

The **:***Display* parameter is optional, but useful when using the multihead option and must correspond to the values passed with the **-P** flag. The **:***Display* parameter is the name of the display as shown in the **lsdisp** command. In the absence of the **:***Display* parameter, the specified depth is selected for all the selected displays in the multihead option, as specified in the **-P** flag.

-damage
Disables the X Damage extension.

+damage
Enables the X Damage extension.

-evie Disables the X Event Interception extension.

+evie Enables the X Event Interception extension.

-f *Number* Specifies the beep volume. The default is -1 or medium. The supported values are as follows:

Value	Setting
0	Off
1-33	Low
-1 or **34-66**	Medium
67-100	High

-fc *Font* Specifies the cursor font for cursor glyphs and cursor masks. The default depends on the operating system and the display.

-fixes Disables the X Fixes extension.

+fixes Enables the X Fixes extension.

-fn *Font* Specifies the default text font. The default depends on the operating system and the display.

-fp *Font* Specifies the font path.

-I Causes all remaining command line arguments to be ignored. (Uppercase i)

-help Prints a usage message.

-layer *#*[**:***Display*] Specifies that the default visual should be in the *#* layer. The **:***Display* parameter is the name of the display as shown in the **lsdisp** command. Specifying this flag for an adapter that does not have overlays, or has less than 8 bits of overlay, has no effect. Specifying this flag with a *#* higher than the number of supported layers results in the default visual residing in the default layer of the screen (as if no **-layer** flag had been used).

-logo Turns on the X Window System logo display in the screen saver. There is currently no way to change this from a client.

-n :*Number* Specifies the connection number. Valid values for the *Number* parameter are 0 to 255. The default is the next available number. The *Number* parameter is used by programs to communicate with a specific X Server. For example, the command:

```
X -n :18
```

specifies that communication to the activated X Server takes place by unix:18 or by *Hostname*:18.

-nobs Disables backing store support on all screens. This is the default.

nologo Turns off the X Window System logo display in the screen saver. There is currently no way to change this from a client.

-once Instructs the server to exit after the first session ends. Normally, the server starts sessions automatically.

-P*RowColumn Display* Specifies the physical positioning of the displays in a multihead configuration. The *Row* parameter indicates the row in which the display is located. The *Column* parameter indicates the column in which the display is located.

The *Display* parameter is the device name of the display as shown in the first column of output from the **lsdisp** command. The first **-P***RowColumn Display* occurrence on the command line describes screen 0 to the X Server, the second describes screen 1, and so on.

The **-P** flag is for use with multiple head support.

-pbuffer *level* [**:display** *name* | **:display** *number*]
Specifies the **pbuffer** memory allocation level for the screen specified by **:display**. This flag is only useful when used in conjunction with the GLX extension.

The *level* parameter indicates the relative amount of frame buffer memory to be reserved for pbuffers. Specified values must be in the range of [0..2]. A value of 0 indicates that no memory should be reserved for pbuffers. A value of 1 indicates that a low amount of memory should be reserved. A value of 2 indicates that a high amount of memory should be reserved. Not all adapters support pbuffers. For those that do, not all screen configurations support pbuffers. The actual amount of frame buffer memory reserved for pbuffers is device-dependent, and may be influenced by other factors, such as screen resolution or default pixel depth.

The **:display** parameter is optional, but useful when using the multihead option. The **:display** parameter is the name of the display as shown in the **lsdisp** command. If no display *number* or *name* is supplied, the specified **pbuffer** width is selected for all screens.

-p *Number*
Specifies the time interval, in minutes, between changes of the X Window System logo position. This flag is used with the **-s** (screen saver timeout) flag to control the blanking of the screen.

-r
Disables autorepeat. The default is autorepeat enabled.

r
Turns on autorepeat.

+render
Enables the X Render extension. By default, the X Render extension is disabled.
Note: Use the **+render** flag with the **-vfb** flag. Currently, only Virtual Frame Buffer (VFB) is enabled for the X Render extension.

-render
Disables the X Render extension.

-s *Number*
Specifies the number of minutes to wait before blanking the screen. The default is 10 minutes. If this value is set to 0, the screen-saver is disabled.

-secIP [*PermissionCode*]
Sets local access control on the internet socket. The *PermissionCode* is 3 octal digits which can set read, write, and execute bits. If no *PermissionCode* is specified after a security flag, then permission is defaulted to 0 for that socket.

-secLocal [*PermissionCode*]
Sets access control on the unix socket. The *PermissionCode* is 3 octal digits which can set read, write, and execute bits. If no *PermissionCode* is specified after a security flag, then permission is defaulted to 0 for that socket.

-secSMT [*PermissionCode*]
Sets access control on the shared memory transport socket. The *PermissionCode* is 3 octal digits which can set read, write, and execute bits. If no *PermissionCode* is specified after a security flag, then permission is defaulted to 0 for that socket.

-stereo [**:***Display*]
Configures the graphics adapter for optimum stereo support for the screen specified by *Display*.

Supported screens will configure the adapter to provide the best available support for stereo. This may decrease other resources such as texture memory. The actual amount of memory affected is device-dependent, and may be influenced by other factors, such as screen resolution or default pixel depth.

The *Display* parameter is optional, but useful when using the multihead option. The *Display* parameter is the name of the display as shown in the **lsdisp** command. If no display number or name is supplied, the **-stereo** flag pertains to all supported screens.

Unsupported screens will ignore the **-stereo** flag.

-su
Disables save under support on all screens.

-T
Disables the Ctrl+Alt+Backspace key sequence that, by default, ends the AIXwindows session and all windows opened from it.

-t *Number*
Specifies the mouse threshold. The default is 2 pixels. Acceleration takes effect only if the mouse is moved beyond the mouse threshold in one time interval and only applies to the amount beyond the threshold.

-to *Number*
Specifies the number of minutes to elapse between connection checks. The default is 60 minutes. A specified value must be greater than 0.

-v
Specifies that the display be replaced with the current background color after the time specified by the **-s** flag expires. By default, if the **-v** flag is not used, the entire display is painted with the background tile after the time specified by the **-s** flag expires.

-vfb
Starts the X Server with Virtual Frame Buffer (VFB), without initializing any graphics adaptor.

-wm
Forces the default backing store of all windows to have the **WhenMapped** value. This is a convenient way of applying backing store to all windows.

-wp *Color*
Specifies a white pixel display color. The default depends on the display.

-wrap

Specifies the behavior of the mouse when its hotspot reaches the left or right border or the top or bottom of any root window. If this flag is set and the hotspot of the mouse reaches the left border of the leftmost root window, the mouse is automatically positioned at the right border of the rightmost root window at the same y position.

Conversely, if this flag is set and the hotspot of the mouse reaches the right border of the rightmost root window, the mouse is automatically positioned at the left border of the leftmost root window at the same y position. If this flag is not set, the mouse stops at the left or right border of any root window.

If this flag is set and the hotspot of the mouse reaches the top border of the topmost root window, the mouse is positioned at the bottom border of the bottommost root window at the same x position.

Conversely, if this flag is set and the hotspot of the mouse reaches the bottom border of the bottommost root window, the mouse is positioned at the top border of the topmost root window at the same x position.

The **-wrap** flag is for use with multiple head support.

-wrapx

Specifies the behavior of the mouse when its hotspot reaches the left or right border of any root window. If this flag is set and the hotspot of the mouse reaches the left border of the leftmost root window, the mouse is positioned at the right border of the rightmost root window at the same y position. Conversely, if this flag is set and the hotspot of the mouse reaches the right border of the rightmost root window, the mouse is positioned at the left border of the leftmost root window at the same y position. If this flag is not set, the mouse stops at the left or right border of any root window.

The **-wrapx** flag is for use with multiple head support.

-wrapy

Specifies the behavior of the mouse when its hotspot reaches the top or bottom border of any root window. If this flag is set and the hotspot of the mouse reaches the top border of the topmost root window, the mouse is positioned at the bottom border of the bottommost root window at the same x position.

Conversely, if this flag is set and the hotspot of the mouse reaches the bottom border of the bottommost root window, the mouse is positioned at the top border of the topmost root window at the same x position. If this flag is not set, the mouse stops at the top or bottom border of any root window.

The **-wrapy** flag is for use with multiple head support.

-x *ExtensionName*

Specifies that the extension name should be loaded when the server is initialized. This is particularly useful for large extensions, such as the Display PostScript® Level 2 (**dps**). This flag can be specified more than once with multiple extension names.

-query *HostName*
Enables Enhanced X-Windows Display Manager Control Protocol (**XDMCP**) and sends a **Query** packet to the specified host.

The **-query** flag is for use with **XDMCP**.

-broadcast
Enables **XDMCP** and broadcasts **BroadcastQuery** packets to the network. The first responding display manager is chosen for the session.

The **-broadcast** flag is for use with **XDMCP**.

-indirect *HostName*
Enables **XDMCP** and sends **IndirectQuery** packets to the specified host.

The **-indirect** flag is for use with **XDMCP**.

-port *PortNumber*
Specifies an alternative port number for **XDMCP**. This flag must be specified before any **-query**, **-broadcast**, or **-indirect** flags. Normally, the server starts sessions one after another. This flag causes the server to exit after the first session ends.

The **-port** flag is for use with **XDMCP**.

-class *DisplayClass*
Sets the value for an additional display qualifier used by **XDMCP** in resource lookup for display-specific options.

The **-class** flag is for use with **XDMCP**.

-cookie *XDMAuthenticationBits*
Specifies a private key to be shared between the server and the manager when testing XDM-AUTHENTICATION-1.

The **-cookie** flag is for use with **XDMCP**.

-displayID *DisplayID*
Allows the display manager to identify each display so that it can locate the shared key specified by the **-cookie** flag.

The **-displayID** flag is for use with **XDMCP**.

+/- xinerama
Enable/Disable panoramic screen or Virtual Large Screen (VLS). Allows users to treat all heads in a multihead environment as a large screen.

Xkeyboard Flags

-xkbdir *Directory*
Specifies the base directory for the keyboard layout files.

-xkbmap *FileName*
Specifies the keyboard description to load on startup.

[**+**|**-**]**accessx**
Enables (**+**) or disables (**-**) AccessX key sequences.

-ar1 *Milliseconds*
Sets the length of time in milliseconds that a key must be pressed before autorepeat starts.

-ar2 *Milliseconds*
Sets the length of time in milliseconds that should elapse between autorepeat generated keystrokes.

Security Extension Flags

-sp *FileName* Causes the server to attempt to read and interpret *FileName* as a security policy file with the format described below. The file is read at server startup and reread at each server reset.

The syntax of the security policy file is as follows. Notation: ″*″ means zero or more occurrences of the preceding element, and ″+″ means one or more occurrences. To interpret *foo*/*bar*, ignore the text after the /; it is used to distinguish between instances of *foo* in the next section.

```
policy file ::= version line other line*
version line ::= string/v '\n'
other line  ::= comment | access rule | site policy | blank line
comment ::= # not newline* '\n'
blank line ::= space '\n'
site policy ::= sitepolicy string/sp '\n'
access rule ::= property property/ar window perms '\n'
property ::= string
window ::= any | root | required property
required property ::= property/rp | property with value
property with value ::= property/rpv = string/rv
perms ::= [ operation | action | space ]*
operation ::= r | w | d
action ::= a | i | e
string ::= dbl quoted string | single quoted string | unqouted string
dbl quoted string ::= space " not dqoute* " space
single quoted string ::= space ' not squote* ' space
unquoted string ::= space not space+ space
space ::= [ ' ' | '\t' ]*
```

Character sets:

```
not newline ::= any character except '\n'
not dqoute  ::= any character except "
not squote  ::= any character except '
not space   ::= any character except those in space
```

The semantics associated with the previously described syntax are as follows.

version line
The first line in the file, specifies the file format version. If the server does not recognize the version *string***/v**, it ignores the rest of the file. The version string for the file format described here is `version-1`.

Once past the *version line*, lines that do not match the above syntax are ignored.

comment
Lines are ignored.

sitepolicy
Lines are currently ignored. They are intended to specify the site policies used by the XC-QUERY-SECURITY-1 authorization method.

access rule
Lines specify how the server should react to untrusted client requests that affect the X Window property named *property***/ar**. The rest of this section describes the interpretation of an *access rule*.

For an *access rule* to apply to a given instance of *property***/ar**, *property***/ar** must be on a window that is in the set of windows specified by *window*. If *window* is **any**, the rule applies to *property***/ar** on any window. If *window* is **root**, the rule applies to *property***/ar** only on root windows.

If *window* is *required property*, the following apply. If *required property* is a *property***/rp**, the rule applies when the window also has that *property***/rp**, regardless of its value. If *required property* is a *property with value*, *property***/rpv** must also have the value specified by *string***/rv**. In this case, the property must have type STRING and format 8, and should contain one or more null- terminated strings. If any of the strings match *string***/rv**, the rule applies.

The definition of string matching is simple case-sensitive string comparison with one elaboration: the occurrence of the character '*' in *string***/rv** is a wildcard meaning ″any string.″ A *string***/rv** can contain multiple wildcards anywhere in the string. For example, `x*` matches strings that begin with **x**, `*x` matches strings that end with **x**, `*x*` matches strings containing **x**, and `x*y*` matches strings that start with **x** and subsequently contain **y**.

There may be multiple *access rule* lines for a given *property***/ar**. The rules are tested in the order that they appear in the file. The first rule that applies is used.

perms Specify operations that untrusted clients may attempt, and the actions that the server should take in response to those operations.

operation
Can be **r** (read), **w** (write), or **d** (delete). The following information shows how X Protocol property requests map to these operations in the X Consortium server implementation.

GetProperty
r, or **r** and **d** if delete = True

ChangeProperty
w

RotateProperties
r and **w**

DeleteProperty
d

ListProperties
none, untrusted clients can always list all properties

action Can be **a** (allow), **i** (ignore), or **e** (error).

Allow Executes the request as if it had been issued by a trusted client.

Ignore Treats the request as a no-op. In the case of GetProperty, ignore means return an empty property value if the property exists, regardless of its actual value.

Error Specifies not to execute the request and return a `BadAtom` error with the `atom` set to the property name. Error is the default action for all properties, including those not listed in the security policy file.

An *action* applies to all *operation*s that follow it, until the next *action*> is encountered. Thus, `irwad` means ignore read and write, allow delete.

GetProperty and RotateProperties might do multiple operations (**r** and **d**, or **r** and **w**). If different actions apply to the operations, the most severe action is applied to the whole request; there is no partial request execution. The severity ordering is: allow < ignore < error. Thus, if the *perms* for a property are `ired` (ignore read, error delete), and an untrusted client attempts GetProperty on that property with `delete = True`, an error is returned, but the property value is not. Similarly, if any of the properties in a RotateProperties do not allow both read and write, an error is returned without changing any property values.

An example a security policy file follows:

```
version-1

# Allow reading of application resources, but not writing.
property RESOURCE_MANAGER     root      ar iw
property SCREEN_RESOURCES     root      ar iw

# Ignore attempts to use cut buffers.  Giving errors causes apps to crash,
# and allowing access may give away too much information.
property CUT_BUFFER0          root      irw
property CUT_BUFFER1          root      irw
property CUT_BUFFER2          root      irw
property CUT_BUFFER3          root      irw
property CUT_BUFFER4          root      irw
property CUT_BUFFER5          root      irw
property CUT_BUFFER6          root      irw
property CUT_BUFFER7          root      irw

# If you are using Motif, you probably want these.

property _MOTIF_DEFAULT_BINDINGS        rootar iw
property _MOTIF_DRAG_WINDOW   root      ar iw
property _MOTIF_DRAG_TARGETS  any       ar iw
property _MOTIF_DRAG_ATOMS    any       ar iw
property _MOTIF_DRAG_ATOM_PAIRS         any ar iw

# The next two rules let xwininfo -tree work when untrusted.
property WM_NAME              any       ar




# Allow read of WM_CLASS, but only for windows with WM_NAME.
# This might be more restrictive than necessary, but demonstrates
# the required property facility, and is also an attempt to
# say "top level windows only."
property WM_CLASS             WM_NAME   ar

# These next three let xlsclients work untrusted.  Think carefully
# before including these; giving away the client machine name and command
# may be exposing too much.
property WM_STATE             WM_NAME   ar
property WM_CLIENT_MACHINE    WM_NAME   ar
property WM_COMMAND           WM_NAME   ar

# To let untrusted clients use the standard colormaps created by
# xstdcmap, include these lines.
property RGB_DEFAULT_MAP      root      ar
property RGB_BEST_MAP         root      ar
property RGB_RED_MAP          root      ar
property RGB_GREEN_MAP        root      ar
property RGB_BLUE_MAP         root      ar
property RGB_GRAY_MAP         root      ar
```

```
# To let untrusted clients use the color management database created
# by xcmsdb, include these lines.
property XDCCC_LINEAR_RGB_CORRECTION    rootar
property XDCCC_LINEAR_RGB_MATRICES      rootar
property XDCCC_GRAY_SCREENWHITEPOINT    rootar
property XDCCC_GRAY_CORRECTION          rootar


# oddball property names and explicit specification of error conditions
property "property with spaces"         'property with "'aw er ed

# Allow deletion of Woo-Hoo if window also has property OhBoy with value
# ending in "son".  Reads and writes will cause an error.
property Woo-Hoo              OhBoy = "*son"ad
```

Example

To start the X Server with X Render extension, enter the following command:

```
$X -T -force :0 -vfb -d 32 +render
```

In this example, the X Server will use the Virtual Frame Buffer (VFB) for rendering instead of using the physical graphics adaptor.

Related Information

The **aixterm** command, **xclock** command, **xhost** command, **xinit** command, **xlsfonts** command, **xwd** command, **xwud** command.

The **lsdisp** shell command.

---

x_add_fs_fpe Command

Purpose

Adds a network font server to a font path.

Syntax

x_add_fs_fpe *Host Port Position TypeName*

Description

The **x_add_fs_fpe** command adds a font path element to the font path of the selected network type name for a font server to access fonts.

Host	Specifies the name of the system where the font server resides.
Port	Specifies the number of the font server port. This number must be in the **/etc/services** file and specified in decimal.
Position	Specifies where to insert this element in the font path.
TypeName	Specifies the name of the network type. Each network type has a font path consisting of one or more font path elements. Specify the name of the network type to which the font path element will be added, or choose to have it added to all network type names by specifying `All`. If a font path element is added to `All` network types, will be placed at the end of each font path.

Security

Access Control: Only the root user should have execute (x) access to this command.

Example

To add the font server to the start of the font path for network type `x_st_mgr.ether`, enter:

```
x_add_fs_fpe winter 7500 1 x_st_mgr.ether
```

In this example, the font server on host `winter` has been added to the start of the font path for network type `x_st_mgr.ether`. The font server port is `7500`.

Files

/usr/lpp/x_st_mgr/bin/x_add_fs_fpe	Contains the **x_add_fs_fpe** command.
/etc/x_st_mgr/ether.cf	Contains the network type **x_st_mgr.ether** configuration file (sample).

Related Information

The **aixterm** command, **bootpd** daemon, **login** command, **x_add_nfs_fpe** command, **x_rm_fpe** command.

---

x_add_nfs_fpe Command

Purpose

Adds a NFS/TFTP accessed font directory to a font path.

Syntax

x_add_nfs_fpe *Host Directory Method Position TypeName*

Description

The **x_add_nfs_fpe** command adds a font path element to the font path of the selected network type name. This font directory will be accessed using Network File System (NFS) or Trivial File Transfer Protocol (TFTP).

Host	Specifies the system name to access for the font directory.
Directory	Specifies the complete path to the directory that contains the fonts.
Method	Specifies either `nfs` or `tftp` to be used to access the fonts.
Position	Specifies where to insert this element in the font path.
TypeName	Specifies the name of the network type. Each network type has a font path consisting of one or more font path elements. Specify the name of the network type to which the font path element will be added, or choose to have it added to all network type names by specifying `All`. If a font path element is added to `All` network types, it will be placed at the end of each font path.

Security

Access Control: Only the root user should have execute (x) access to this command.

Example

To add the fonts in `/usr/lib/X11/fonts/100dpi` to the network type `x_st_mgr.ether`, enter:

```
x_add_nfs_fpe cedar /usr/lib/X11/fonts/100dpi nfs Last \ x_st_mgr.ether
```

In this the font path element /usr/lib/X11/fonts/100dpi is added to the end of the font path for network type x_st_mgr.ether. The font directory is on the host cedar, which is accessed using NFS.

Files

/usr/lpp/x_st_mgr/bin/x_add_nfs_fpe	Contains the **x_add_nfs_fpe** command.
/etc/x_st_mgr/ether.cf	Contains the network type **x_st_mgr.ether** configuration file (sample).

Related Information

The **aixterm** command, **bootpd** daemon, **login** command, **x_add_fs_fpe** command, **x_rm_fpe** command.

---

x_rm_fpe Command

Purpose

Removes a font path element from a font path.

Syntax

x_rm_fpe *TypeName Position Method Host Post Directory*

Description

The **x_rm_fpe** command removes a font path element from the font path of the selected network type name.

TypeName	Specifies from which network type name the element is to be removed.
Position	Specifies where the element is in the font path.
Method	Specifies the method used to access the font path element. The valid options are: **tcp** for Network Font Server; **default** for initial default font path element; **nfs** for NFS; and **tftp** for TFTP.
Host	Specifies the name of the system specified in the font path element. For elements using the default method, specify **None**.
Port	Specifies the number of the server port specified in the font path element. For elements using the nfs or tftp method, specify **None**.
Directory	Specifies the complete path to the directory that contains the fonts. For a Network Font Server element, specify **None**.

Security

Access Control: Only the root user should have execute (x) access to this command.

Examples

To remove the font element /usr/lib/X11/fonts/100dpi from the font path for network type x_st_mgr.ether, enter:

```
x_rm_fpe x_st_mgr.ether 3 nfs waco None /usr/lib/X11/fonts/100dpi
```

In this example, the font path element /usr/lib/X11/fonts/100dpi that is accessed on host waco using NFS has been removed from the third position of the font path for network type x_st_mgr.ether. Because a port number is not used for NFS, this parameter was set to None.

Files

/usr/lpp/x_st_mgr/bin/x_rm_fpe	Contains the **x_rm_fpe** command.

/etc/x_st_mgr/ether.cf Contains the network type **x_st_mgr.ether** configuration file (sample).

Related Information

The **aixterm** command, **bootpd** daemon, **login** command, **x_add_nfs_fpe** command, **x_rm_fpe** command.

---

xargs Command

Purpose

Constructs parameter lists and runs commands.

Syntax

xargs [**-p t x**] [**-e** [*EOFString*]] [**-E** *EOFString*] [**-i** [*ReplaceString*]] [**-I** *ReplaceString* | **-L** *Number* | [**-n** *Number*] [**-l** [*Number*]] [**-s** *Size*] [*Command* [*Argument*...]]

Note: Do not put a blank space between the lowercase flags and the parameter.

Description

The generated command line length is the sum of the size, in bytes, of the *Command* and each *Argument* treated as strings, including a null byte terminator for each of these strings. The **xargs** command limits the command line length. When the constructed command line runs, the combined *Argument* and environment lists can not exceed **ARG_MAX** bytes. Within this constraint, if you do not specify the **-n** or the **-s** flags, the default command line length is at least the value specified by **LINE_MAX**.

Flags

-e[*EOFString*] Obsolete flag. Use the **-E** flag.

Uses the *EOFString* parameter as the logical EOF string. If you do not specify the **-e** or the **-E** flags, underscore (_) is assumed for the logical EOF string. If you do not specify the *EOFString* parameter, the logical EOF string capability is disabled, and underscores are taken literally. The **xargs** command reads from standard input until either EOF or the specified string is reached.

-E *EOFString* Specifies a logical EOF string to replace the default underscore(_). The **xargs** command reads standard input until either EOF or the specified string is reached.

-i[*ReplaceString*] Obsolete flag. Use the **-I** (Uppercase i) flag.

If you do not specify the *ReplaceString* parameter, the string ″{}″ is used.

Note: The **-I** (Uppercase i), and the **-i** flags are mutually exclusive; the last flag specified takes effect.

-I *ReplaceString* (Uppercase i). Inserts each line of standard input as an argument for the *Command* parameter, inserting it in *Argument* for each occurrence of *ReplaceString*. *ReplaceStrings* can not be used in more than 5 arguments. Blank characters at the beginning of each standard input line are ignored. Each *Argument* can contain one or more *ReplaceStrings,* but may not be larger than 255 bytes. The **-I** flag also turns on the **-x** flag.

Note: The **-I** (Uppercase i), and the **-i** flags are mutually exclusive; the last flag specified takes effect.

-l[*Number*] (Lowercase L). Obsolete flag. Use the **-L** flag.

If you do not specify the *Number* parameter, a value of 1 is used. The **-l** flag also turns on the **-x** flag.

> **Note:** The **-L**, **-l** (Lowercase L), and **-n** flags are mutually exclusive; the last flag specified takes effect.

-L *Number* Runs the *Command* parameter with the specified number of nonempty parameter lines read from standard input. The last invocation of the *Command* parameter can have fewer parameter lines if fewer than the specified *Number* remain. A line ends with the first new-line character unless the last character of the line is a space or a tab. A trailing space indicates a continuation through the next nonempty line.

> **Note:** The **-L**, **-l** (Lowercase L), and **-n** flags are mutually exclusive; the last flag specified takes effect.

-n *Number* Runs the *Command* parameter using as many standard input arguments as possible, up to the maximum specified by the *Number* parameter. The **xargs** command uses fewer arguments if:

1. If the accumulated command line length exceeds the bytes specified by the **-s** *Size* flag.
2. The last iteration has fewer than *Number*, but not zero, arguments remaining.

> **Note:** The **-L**, **-l** (Lowercase L), and **-n** flags are mutually exclusive; the last flag specified takes effect.

-p Asks whether to run the *Command* parameter. It displays the constructed command line, followed by a `?...` (question mark, ellipsis) prompt. Enter an affirmative response specific to the locale to run the *Command* parameter. Any other response causes the **xargs** command to skip that particular invocation of the parameter. You are asked about each invocation. The **-p** flag also turns on the **-t** flag.

-s *Size* Sets the maximum total size of the constructed *Command* line. The *Size* parameter must be a positive integer. Fewer arguments are used if:

1. The total number of arguments exceeds those specified by the **-n** flag.
2. The total number of lines exceeds those specified by the **-L** or **-l** (Lowercase L) flags.
3. EOF is reached before the number of bytes specified by the *Size* parameter are accumulated.

-t Enables the trace mode and echoes the constructed *Command* line to standard error before running.

-x Stops running the **xargs** command if any *Command* line is greater than the number of bytes specified by the **-s** *Size* flag. This **-x** flag is turned on if you specify either the **-l** (Uppercase i) or **-l** (Lowercase L) flag. If you do not specify the **-i**, **-l** (Uppercase i), **-l** (Lowercase L), **-L**, or**-n** flag, the total length of the *Command* line must be within the limit specified by the -s *Size* flag.

Exit Status

This command returns the following exit values:

0 All invocations of the *Command* parameter returned exit status 0.

1-125 A command line meeting the specified requirements could not be assembled, one or more of the invocations of the *Command* parameter returned a non-zero exit status, or some other error occurred.

126 *Command* was found but could not be invoked.

127 *Command* could not be found.

If a command line meeting the specified requirements cannot be assembled, the command cannot be invoked, an invocation of the command is terminated by a signal, or an invocation of the command exits with exit status 255. The **xargs** command will write a diagnostic message and exit without processing any remaining input.

Examples

1. To use a command on files whose names are listed in a file, type:

   ```
   xargs lint -a <cfiles
   ```

   If the `cfiles` file contains the following text:

   ```
   main.c readit.c
   gettoken.c
   putobj.c
   ```

   the **xargs** command constructs and runs the following command:

   ```
   lint -a main.c readit.c gettoken.c putobj.c
   ```

   If the `cfiles` file contains more file names than fit on a single shell command line (up to **LINE_MAX**), the **xargs** command runs the **lint** command with the file names that fit. It then constructs and runs another **lint** command using the remaining file names. Depending on the names listed in the `cfiles` file, the commands might look like the following:

   ```
   lint -a main.c readit.c gettoken.c . . .
   lint -a getisx.c getprp.c getpid.c . . .
   lint -a fltadd.c fltmult.c fltdiv.c . . .
   ```

   This command sequence is not quite the same as running the **lint** command once with all the file names. The **lint** command checks cross-references between files. However, in this example, it cannot check between the `main.c` and the `fltadd.c` files, or between any two files listed on separate command lines.

   For this reason you may want to run the command only if all the file names fit on one line. To specify this to the **xargs** command use the **-x** flag by typing:

   ```
   xargs  -x lint -a <cfiles
   ```

   If all the file names in the `cfiles` file do not fit on one command line, the **xargs** command displays an error message.

2. To construct commands that contain a certain number of file names, type:

   ```
   xargs  -t  -n 2 diff <<EOF
   starting chap1 concepts chap2 writing
   chap3
   EOF
   ```

   This command sequence constructs and runs **diff** commands that contain two file names each (**-n 2**):

   ```
   diff starting chap1
   diff concepts chap2
   diff writing chap3
   ```

   The **-t** flag causes the **xargs** command to display each command before running it, so you can see what is happening. The `<<EOF` and `EOF` pattern-matching characters define a *here document*, which uses the text entered before the end line as standard input for the **xargs** command.

3. To insert file names into the middle of command lines, type:

   ```
   ls | xargs  -t  -I {} mv {} {}.old
   ```

This command sequence renames all files in the current directory by adding `.old` to the end of each name. The **-I** flag tells the **xargs** command to insert each line of the **ls** directory listing where {} (braces) appear. If the current directory contains the files `chap1`, `chap2`, and `chap3`, this constructs the following commands:

```
mv chap1 chap1.old
mv chap2 chap2.old
mv chap3 chap3.old
```

4. To run a command on files that you select individually, type:

   ```
   ls | xargs  -p  -n 1 ar r lib.a
   ```

   This command sequence allows you to select files to add to the `lib.a` library. The **-p** flag tells the **xargs** command to display each **ar** command it constructs and to ask if you want to run it. Type `y` to run the command. Press the any other key if you do not want to run the command.

   Something similar to the following displays:

   ```
   ar r lib.a chap1 ?...
   ar r lib.a chap2 ?...
   ar r lib.a chap3 ?...
   ```

5. To construct a command that contains a specific number of arguments and to insert those arguments into the middle of a command line, type:

   ```
   ls | xargs -n6 | xargs -I{} echo {} - some files in the directory
   ```

   If the current directory contains files chap1 through chap10, the output constructed will be the following:

   ```
   chap1 chap2 chap3 chap4 chap5 chap6 - some files in the directory
   chap7 chap8 chap9 chap10 - some file in the directory
   ```

File

/usr/bin/xargs	Contains the **xargs** command.

Related Information

The **ar** command, **diff** command, **echo** command, **ksh** command, **lint** command, **ls** command, **mv** command.

Shells and Commands in *Operating system and device management*.

Input and Output Handling Programmer's Overview in *AIX 5L Version 5.3 General Programming Concepts: Writing and Debugging Programs*.

---

xauth Command

Purpose

Edits and displays the authorization information used in connecting to the X server.

Syntax

xauth [**-f** *AuthFile*] [**-v** | **-q**] [**-i**] [**-b**] [*CommandArgument ...*]

Description

The **xauth** command is usually used to edit and display the authorization information used in connecting to the X server. This program extracts authorization records from one machine and merge them into another (for example, when using remote logins or granting access to other users).

The following commands can be entered interactively, on the **xauth** command line, or in scripts. Note that this program does not contact the X server.

add *DisplayName ProtocolName Hexkey*

An authorization entry is added to the authorization file for the indicated display using the given protocol and key data. The data is specified as an even-length string of hexadecimal digits, each pair representing one octet. The first digit of each pair gives the most significant 4 bits of the octet, and the second digit of the pair gives the least significant 4 bits. For example, a 32-character hexkey would represent a 128-bit value. A protocol name consisting of just a single period is treated as an abbreviation for **MIT-MAGIC-COOKIE-1**.

extract *FileName DisplayName...*

Authorization entries for each of the specified displays are written to the indicated file. The extracted entries can be read back in using the **merge** and **nmerge** commands. If the file name consists of just a single dash, the entries are written to the binary output.

generate *DisplayName ProtocolName* [*trusted* | *untrusted*] [*timeout seconds*] [*group group-id*] [*data hexdata*]

This command is similar to **add**. The main difference is that instead of requiring the user to supply the key data, it connects to the server specified in *displayname* and uses the SECURITY extension in order to get the key data to store in the authorization file. If the server cannot be contacted or if it does not support the SECURITY extension, the command fails. Otherwise, an authorization entry for the indicated display using the given protocol is added to the authorization file. A protocol name consisting of just a single period is treated as an abbreviation for MIT-MAGIC-COOKIE-1.

If the trusted option is used, clients that connect using this authorization will have full run of the display, as usual. If untrusted is used, clients that connect using this authorization will be considered untrusted and prevented from stealing or tampering with data belonging to trusted clients. See the SECURITY extension specification for full details on the restrictions imposed on untrusted clients. The default is untrusted.

The timeout option specifies how long in seconds this authorization will be valid. If the authorization remains unused (no clients are connected with it) for longer than this time period, the server purges the authorization, and future attempts to connect using it will fail. Note that the purging done by the server does not delete the authorization entry from the authorization file. The default timeout is 60 seconds.

The group option specifies the application group that clients connecting with this authorization should belong to. See the application group extension specification for more details. The default is to not belong to an application group.

The data option specifies data that the server should use to generate the authorization. Note that this is not the same data that gets written to the authorization file. The interpretation of this data depends on the authorization protocol. The *hexdata* is in the same format as the *hexkey* described in the **add** command. The default is to send no data.

list [*DisplayName*...]	Authorization entries for each of the specified displays (or all displays if none are named) are printed on the standard output in a textual format. Key data is always displayed in the hexadecimal format given in the description of the **add** command.
merge [*FileName*...]	Authorization entries are read from the specified files and are merged into the authorization database, superseding any matching existing entries. If a file name consists of just a single dash, the binary input is read if it has not been read before.
[n]extract *Filename DisplayName*...	Authorization entries for each of the specified displays are written to the indicated file. The entries are written in a numeric format suitable for non-binary transmission (such as secure electronic mail). The extracted entries can be read back in using the **merge** and **nmerge** commands. If the file name consists of just a single dash, the entries are written to the standard output.
[n]list [*DisplayName*...]	Authorization entries for each of the specified displays (or all displays if none are named) are printed on the standard output in the numeric format used by the **nextract** command. Key data is always displayed in the hexadecimal format given in the description of the **add** command.
[n]merge [*FileName*...]	Authorization entries are read from the specified files and are merged into the authorization database, superseding any matching existing entries. The numeric format given in the description of the **extract** command is used. If a file name consists of just a single dash, the standard input is read if it has not been read before.
remove *DisplayName*...	Authorization entries matching the specified displays are removed from the authority file.
source *FileName*	The specified file is treated as a script containing **xauth** commands to execute. Blank lines and lines beginning with a # (pound sign) are ignored. A single dash can be used to indicate the standard input, if it has not already been read.
info	Information describing the authorization file, whether or not any changes have been made, and from where **xauth** commands are being read is printed on the standard output.
exit	If any modifications have been made, the authority file is written out (if allowed), and the program exits. An end of file is treated as an implicit exit command.
quit	The program exits, ignoring any modifications. This may also be accomplished by pressing the interrupt character.
help [*String*]	A description of all commands that begin with the given string (or all commands if no string is given) is printed on the standard output.
?	A short list of the valid commands is printed on the standard output.

Display names for the **add**, [**n**]**extract**, [**n**]**list**, [**n**]**merge**, and **remove** commands use the same format as the **DISPLAY** environment variable and the common *display* command-line argument. Display-specific information (such as the screen number) is unnecessary and is ignored. Same-machine connections (such as local-host sockets, shared memory, and the Internet Protocol *HostName LocalHost*) are referred to as `HostName/unix:DisplayNumber` so that local entries for different machines can be stored in one authority file.

Note: Users that have unsecure networks should take care to use encrypted file transfer mechanisms to copy authorization entries between machines. Similarly, the MIT-MAGIC-COOKIE-1 protocol is not very useful in unsecure environments. Sites that are interested in additional security may need to use encrypted authorization mechanisms such as Kerberos. Spaces are currently not allowed in the protocol name. Quoting could be added.

Flags

The following options are used with the **xauth** command. They can be given individually (for example, **-q -i**) or combined (for example, **-qi**).

-f *AuthFile*	Specifies the name of the authority file to use. By default, **xauth** uses the file specified by the **XAUTHORITY** environment variable or .xauthority in the user's home directory.
-v	Indicates that **xauth** should operate verbosely and print status messages indicating the results of various operations (for example, how many records have been read in or written out). This is the default if **xauth** is reading commands from its standard input and its standard output is directed to a terminal.
-q	Indicates that **xauth** should operate quietly and not print unsolicited status messages. This is the default if an **xauth** command is given on the command line or if the standard output is not directed to a terminal.
-i	Indicates that **xauth** should ignore any authority file locks. Normally, **xauth** refuses to read or edit any authority files that have been locked by other programs (usually **xdm** or another **xauth**).
-b	Indicates that **xauth** should attempt to break any authority file locks before proceeding. Use this option only to clean up stale locks.

Example

The most common use for the **xauth** command is to extract the entry for the current display, copy it to another machine, and merge it into the user's authority file on the remote machine:

```
% xauth extract \- $DISPLAY | rsh otherhost xauth merge \-
```

Files

$HOME/.Xauthority	Contains the default authority file if the **XAUTHORITY** environment variable is not defined.

---

xclock Command

Purpose

Continuously displays the current time of day.

Syntax

xclock [*-Xtoolkitoption ...*] [**-analog** | **-digital**] [**-chime**] [**-hd** *Color*] [**-help**] [**-hl** *Color*] [**-padding** *Number*] [**-update** *Seconds*]

Description

The **xclock** command gets the time from the system clock, then displays and updates it in the form of a digital or analog clock. Select the **-analog** or **-digital** flag to display the clock in analog or digital formats. You can also select flags to specify the presentation of the clock, including chime and update frequency, colors, and border width.

This command uses the Athena clock widget, which understands core resource names and classes. To specify these resources, you need to know the hierarchy of the widgets that comprise the **xclock**

command. In the following example, the indented items indicate the hierarchical structure. The widget class name is given first, followed by the widget instance name:

```
XClock xclock
        Clock clock
```

The following examples demonstrate the possible ways to specify resources for this client:

```
xclock.clock.background
XClock*background
xclock*background
```

Note: Specifying resources as xclock.background which worked with the previous version of xclock will not work with this version.

Flags

-Xtoolkitoption	The **xclock** command accepts all of the standard X Toolkit command-line option flags in addition to the specific flags listed. See the List of Enhanced X-Windows Protocols, Toolkit, and Extension Functions for detailed information on the available options.
-analog	Sets the analog display mode, which is the default mode. Draws a conventional 12-hour clock face with ticks for each minute and stroke marks on each hour.
-chime	Specifies the sounding of a chime once on the half hour and twice on the hour.
-digital	Sets the 24-hour digital display mode. Displays the date and time in digital form.
-hd *Color*	Specifies the color of the hands in analog mode on color displays. The default is black.
-help	Prints a brief summary of the allowed options.
-hl *Color*	(lowercase HL) Specifies the highlight color of the edges of the hands of the analog clock. The default is black.
-padding *Number*	Specifies the width in pixels of the padding between the window border and the clock text or picture. The default is 8.
-update *Seconds*	Specifies the frequency in seconds that the **xclock** command updates its display. If the **xclock** window is obscured and then exposed, the **xclock** command redisplays immediately. The specification of an update frequency less than 30 seconds enables the second hand in the analog mode. The default update frequency is 60 seconds.

.Xdefaults Keywords

Use the following keywords to set the defaults for the **xclock** command.

analog (class Boolean)	Specifies an analog clock instead of a digital clock. The default is true.
chime (class Boolean)	Specifies whether a bell sounds on the hour and half hour.
fontSet (class FontSet)	Specifies the fontset for the digital clock. Variable-width fonts do not always display correctly.
foreground (class Foreground)	Specifies the color of tick marks on color displays. If **reverseVideo** is specified, the default is white, otherwise the default is black.
hands (class Foreground)	Specifies the color on the inside of the hands in the analog clock on color displays. If **reverseVideo** is specified, the default is white, otherwise the default is black.
highlight (class Foreground)	Specifies the color used to highlight the clock's hands. If **reverseVideo** is specified, the default is white, otherwise the default is black.
height (class Height)	Specifies the height of the clock. The default for the analog clock is 164 pixels. The default for the digital clock is whatever is required to hold the clock when displayed in the chosen font.
padding (class Margin)	Specifies the amount of internal padding in pixels. The default is 8.

update (class Interval) Specifies the frequency in seconds in which the **xclock** command updates its display.

width (class Width) Specifies the width of the clock. The default for the analog clock is 164 pixels. The default for the digital clock is whatever is needed to hold the clock when displayed in the chosen font.

Environment Variables

DISPLAY Gets the default host and display number.

XENVIRONMENT Gets the name of a resource file that overrides the global resources stored in the RESOURCE_MANAGER property.

Examples

1. To specify a digital clock display, enter:
   ```
   xclock -digital
   ```
2. To specify red hands on an analog clock, enter:
   ```
   xclock -hd red
   ```

File

/usr/lib/X11/app-defaults/XClock Specifies the required resources.

---

xcmsdb Command

Purpose

Loads, queries, or removes Screen Color Characterization Data stored in properties on the root window of the screen.

Note: The **xcmsdb** command is only supported in X11R5 (AIXwindows Version 1.2.3).

Syntax

xcmsdb [**-display** *Display*] [[**-query**] [**-remove**] [**-color**]] | [**-format 32** | **16** | **8**] [*FileName*]

Description

The **xcmsdb** command is used to load, query, or remove Screen Color Characterization Data stored in properties on the root window of the screen. Screen Color Characterization Data is an integral part of **Xlib**, which is necessary for proper conversion between device-independent and device-dependent color specifications. **Xlib** uses the **XDCCC_LINEAR_RGB_MATRICES** and **XDCCC_LINEAR_RGB_CORRECTION** properties to store color characterization data for color monitors. It uses **XDCCC_GRAY_SCREENWWHITEPOINT** and **XDCCC_GRAY_CORRECTION** properties for gray scale monitors. Because **Xlib** allows the addition of Screen Color Characterization Function Sets, added function sets may place their Screen Color Characterization Data on other properties. This utility is unaware of these other properties; therefore, you will need to use a similar utility provided with the function set, or use the example **xprop** utility.

The ASCII readable contents of the *FileName* parameter (or the standard input if no input file is given) are appropriately transformed for storage in properties, provided the **-query** or **-remove** flag options are not specified.

Note: The Xcms API in **libX11.a** is supported; however, the client side color name data base, **/usr/lib/X11/Xcms.txt**, and a device color characterization file, **/usr/lib/X11/XcmsIBM5081.dcc**, are provided as unsupported samples.

Flags

-display*Display*	Specifies the server to which you are converting.
- query	Reads or attempts to read the XDCCC properties off the screen's root window. If successful, it transforms the data into a more readable format, and then sends the data to standard output.
-remove	Removes or attempts to remove the XDCCC properties on the screen's root window.
-color	Sets the **-query** and **-remove** options to only check for the **XDCCC_LINEAR_RGB_MATRICES** and **XDCCC_LINEAR_RGB_CORRECTION** properties. If the **-color** option is not set, the **-query** and **-remove** options check for all the properties.
-format 32 | 16 | 8	Specifies the property format (32, 16, or 8 bits per entry) for the **XDCCC_LINEAR_RGB_CORRECTION** property. Precision of encoded floating-point values increases with the increase in bits per entry. The default is 32 bits per entry.

Parameter

FileName	Specifies the ASCII readable contents of a Screen Color Characterization Data file.

Examples

1. Use the following example to put Screen Color Characterization Data on the root window by telling the **xcmsdb** command to read it from a file:

   ```
   xcmsdb /usr/lib/X11/XcmsIBM5081.dcc
   ```

2. Use the following example after you have already put Screen Color Characterization Data on the root window to tell the **xcmsdb** command to read the data back if it exists:

   ```
   xcmsdb -query
   ```

xdm Command

Purpose

Manages a collection of X Displays with support for XDMCP.

Syntax

xdm [**-config** *ConfigurationFile*] [**-debug** *DebugLevel*] [**-nodaemon**] [**-error** *ErrorLogFile*] [**-resources** *ResourceFile*] [**-server** *ServerEntry*] [**-udpPort** *PortNumber*] [**-session** *SessionProgram*] [**-xrm** *ResourceSpecification*]

Description

The **xdm** (X Display Manager) command manages a collection of X displays, which may be on the local host or remote servers. The design of the **xdm** command was guided by the needs of X terminals as well as the X Consortium standard XDMCP, the *X Display Manager Control Protocol*. The **xdm** command provides services similar to those provided by the **init**, **getty**, and **login** commands on character terminals: prompting for login name and password, authenticating the user, and running a session.

A *session* is defined by the lifetime of a particular process; in the traditional character-based terminal world, it is the user's login shell. In the **xdm** context, it is an arbitrary session manager. This is because in

a windowing environment, a user's login shell process does not necessarily have any terminal-like interface with which to connect. When a real session manager is not available, a window manager or terminal emulator is typically used as the *session manager*, meaning that ending this process ends the user's session.

When the session is ended, **xdm** resets the X server and (optionally) restarts the whole process.

When the **xdm** command receives an **Indirect** query by way of XDMCP, it can run a **chooser** process to perform an XDMCP **BroadcastQuery** (or an XDMCP Query to specified hosts) on behalf of the display and offer a menu of possible hosts that offer XDMCP display management. This feature is useful with X terminals that do not offer a host menu themselves.

Because the **xdm** command provides the first interface that users see, it is designed to be simple to use and easy to customize to the needs of a particular site.

Typical Usage

The **xdm** command is designed to operate in a wide variety of environments.

First, set up the **xdm** configuration file. Make a directory (usually **/usr/lib/X11/xdm**) to contain all of the relevant files. The following is a reasonable configuration file, which could be named **xdm-config**:

```
DisplayManager.servers:       /usr/lib/X11/xdm/Xservers
DisplayManager.errorLogFile:  /usr/lib/X11/xdm/xdm-errors
DisplayManager*resources:     /usr/lib/X11/xdm/Xresources
DisplayManager*startup:       /usr/lib/X11/xdm/Xstartup
DisplayManager*session:       /usr/lib/X11/xdm/Xsession
DisplayManager.pidFile:       /usr/lib/X11/xdm/xdm-pid
DisplayManager._0.authorize:  true
DisplayManager*authorize:     false
```

This file contains references to other files. Some of the resources are specified with an * (asterisk) separating the components. These resources can be made unique for each display by replacing the * (asterisk) with the display name, but typically this is not useful. See the Resources section on the next page for a complete discussion.

The first file, **/usr/lib/X11/xdm/Xservers**, contains the list of displays to manage that are not using **XDMCP**. Most workstations have only one display, numbered 0 (zero), so the file looks something like this:

```
:0 Local local /usr/bin/X11/X -force
```

This keeps **/usr/bin/X11/X** running on this display and manages a continuous cycle of sessions.

The **/usr/lib/X11/xdm/xdm-errors** file contains error messages from **xdm** and anything output to standard error by **Xsetup**, **Xstartup**, **Xsession** or **Xreset** scripts. If you have trouble starting the **xdm** command, check the **/usr/lib/X11/xdm/xdm-errors** file to see if the **xdm** command has any clues to the trouble.

The next configuration entry, **/usr/lib/X11/xdm/Xresources**, is loaded onto the display as a resource database using the **xrdb** command. As the authentication widget reads this database before starting up, it usually contains parameters for that widget.

Flags

All of these options (except **-config**) specify values that can also be specified in the configuration file as resources.

-config *ConfigurationFile* Names the configuration file, which specifies resources to control the behavior of the **xdm** command. The **/usr/lib/X11/xdm/xdm-config** file is the default.

-debug *DebugLevel* Specifies the numeric value for the **DisplayManager.debugLevel** resource. A nonzero value causes **xdm** to print debugging statements to the terminal and disables the **DisplayManager.daemonMode** resource, forcing **xdm** to run synchronously. These error messages may be unclear. To interpret them, check the X11R4 source code for the **xdm** command.

-nodaemon Specifies False as the value for the **DisplayManager.daemonMode** resource. This suppresses the usual daemon behavior, in which the **xdm** command closes all file descriptors, disassociates itself from the controlling terminal, and puts itself in the background when it first starts up.

-error *ErrorLogFile* Specifies the value for the **DisplayManager.errorLogFile** resource. This file contains errors from **xdm** as well as anything written to standard error by the various scripts and programs run during the progress of the session.

-resources *ResourceFile* Specifies the value for the **DisplayManager*resources** resource. This file is loaded using the **xrdb** command to specify configuration parameters for the authentication widget.

-server *ServerEntry* Specifies the value for the **DisplayManager.servers** resource. See the section Server Specification for a description of this resource.

-udpPort *PortNumber* Specifies the value for the **DisplayManager.requestPort** resource. This sets the port number that the **xdm** command monitors for **XDMCP** requests. **XDMCP** uses the registered well-known UDP port 177. Do not change this resource except when debugging.

-session *SessionProgram* Specifies the value for the **DisplayManager*session** resource. This indicates the program to run as the session after the user has logged in.

-xrm *ResourceSpecification* Allows an arbitrary resource to be specified, as in most X Toolkit applications.

Resources

At many stages, the actions of **xdm** can be controlled through the use of its configuration file, which is in the X resource format. Some resources modify the behavior of **xdm** on all displays, while others modify its behavior on a single display. When actions relate to a specific display, the display name is inserted into the resource name between ″DisplayManager″ and the final resource name segment. For example, **DisplayManager.expo_0.startup** is the name of the resource that defines the startup shell file on the ″expo:0″ display. Because the resource manager uses colons to separate the name of the resource from its value and dots to separate resource name parts, **xdm** substitutes underscores for both dots and colons when generating the resource name.

DisplayManager.servers Specifies either a file name full of server entries, one per line (if the value starts with a slash), or a single server entry. See the section Server Specification for details.

DisplayManager.requestPort Indicates the UDP port number that the **xdm** command uses to listen for incoming **XDMCP** requests. Unless you need to debug the system, leave this with its default value of 177.

DisplayManager.errorLogFile Redirects error messages to go to the named file rather than to the console. This file also contains any output directed to standard error by the **Xsetup**, **Xstartup**, **Xsession**, and **Xreset** files, so it will contain descriptions of problems in those scripts as well.

DisplayManager.debugLevel If the integer value of this resource is greater than 0 (zero), the **xdm** command outputs a large amount of debugging information. It also disables daemon mode, which would discard the information and allow nonroot users to run the **xdm** command that would typically not be useful.

DisplayManager.daemonMode The **xdm** command attempts to make itself into a daemon process unassociated with any terminal. This is accomplished by forking and leaving the parent process to exit, and then closing file descriptors and releasing the controlling terminal. In some environments this is not desired (in particular, when debugging). Setting this resource to False disables this feature.

DisplayManager.pidFile
The file name specified is created to contain an ASCII representation of the process ID of the main **xdm** process. The **xdm** command also uses file locking on this file to attempt to eliminate multiple daemons running on the same machine, which would have unpredictable results.

DisplayManager.lockPidFile
Controls whether the **xdm** command uses file locking to keep multiple display managers from running simultaneously.

DisplayManager.authDir
Names a directory in which the **xdm** command stores authorization files while initializing the session. The default value is **/usr/lib/X11/xdm**.

DisplayManager.autoRescan
A Boolean value that controls whether the **xdm** command rescans the configuration, servers, access control, and authentication keys files after a session ends and the files have changed. By default the value is True. You can force the **xdm** daemon to reread these files by sending a **SIGHUP** signal to the main process.

DisplayManager.removeDomainname
When computing the display name for **XDMCP** clients, the name resolver typically creates a fully qualified host name for the terminal. As this is sometimes confusing, the **xdm** command removes the domain name portion of the host name if it is the same as the domain name of the local host when this variable is set. The default value is True.

DisplayManager.keyFile
XDM-AUTHENTICATION-1 style **XDMCP** authentication requires that a private key be shared between the **xdm** daemon and the terminal. This resource specifies the file containing those values. Each entry in the file consists of a display name and the shared key. By default, the **xdm** command does not include support for XDM-AUTHENTICATION-1 because it requires the data encryption method (DES), which is not generally distributable because of United States export restrictions.

DisplayManager.accessFile
To prevent unauthorized **XDMCP** service and to allow forwarding of **XDMCP IndirectQuery** requests, this file contains a database of host names that are allowed direct access to this machine or have a list of hosts to which queries should be forwarded. The format of this file is described in the XDMCP Access Control section.

DisplayManager.exportList
A whitespace-separated list of additional environment variables to pass on to the **Xsetup**, **Xstartup**, **Xsession**, and **Xreset** programs.

DisplayManager.randomFile
A file to checksum to generate the seed of authorization keys. This should be a file that changes frequently. The default is **/dev/mem**.

DisplayManager.choiceTimeout
Number of seconds to wait for the display to respond after a user has selected a host from the chooser. If the display sends an `XDMCP IndirectQuery` within this time, the request is forwarded to the chosen host. Otherwise, it is assumed to be from a new session and the chooser is offered again. The default is 15.

DisplayManager.DISPLAY.resources
Specifies the name of the file to be loaded by the **xrdb** command as the resource database onto the root window of screen 0 of the display. The Login widget, **Xsetup**, and **chooser** programs use the resources set in this file. This resource data base is loaded just before the authentication procedure is started, so it can control the appearance of the login window. See the section Authentication Client , which describes the various resources that are appropriate to place in this file. There is no default value for this resource, but **/usr/lib/X11/xdm/Xresources** is the conventional name.

DisplayManager.DISPLAY.chooser
Specifies the program run to offer a host menu for indirect queries redirected to the special host name **CHOOSER**. **/usr/lib/X11/xdm/chooser** is the default. See the sections **XDMCP Access Control** and **Chooser**.

DisplayManager.DISPLAY.xrdb
Specifies the program used to load the resources. By default, the **xdm** command uses **/usr/bin/X11/xrdb**.

DisplayManager.DISPLAY.cpp
Specifies the name of the C preprocessor that is used by the **xrdb** command.

DisplayManager.DISPLAY.setup
Specifies a program that is run (as root) before offering the login window. This resource may be used to change the appearance of the screen around the login window or to put up other windows (for example, you may want to run **xconsole** here). By default, no program is run. The conventional name for a file used here is **Xsetup**. See the section Setup Program .

DisplayManager.DISPLAY.startup
Specifies a program that is run (as root) after the authentication process succeeds. By default, no program is run. The conventional name for a file used here is **Xstartup**. See the section Startup Program .

DisplayManager.DISPLAY.session
Specifies the session to be run (when not running as root). By default, **/usr/bin/X11/xterm** is run. The conventional name is the **Xsession** script. See the section Session Program.

DisplayManager.DISPLAY.reset
Specifies a program that is run (as root) after the session ends. By default, no program is run. The conventional name is the **Xreset** script. See the section Reset Program .

DisplayManager.DISPLAY.openDelay
Controls the behavior of the **xdm** command when attempting to open intransigent servers by specifying the length of the pause (in seconds) between successive attempts.

DisplayManager.DISPLAY.openRepeat
Controls the behavior of the **xdm** command when attempting to open intransigent servers by specifying the number of attempts to make.

DisplayManager.DISPLAY.openTimeout
Controls the behavior of the **xdm** command when attempting to open intransigent servers by specifying the number of seconds to wait while actually attempting the open (that is, the maximum time spent in the **connect(2)** system call).

DisplayManager.DISPLAY.startAttempts
Controls the behavior of the **xdm** command when attempting to open intransigent servers by specifying the number of times that the entire process is completed before giving up on the server. After the number of attempts specified by the Display Manager **openRepeat** resource have been made, or if the number of seconds specified by the Display Manager **openTimeout** resource elapse in any particular attempt, the **xdm** command ends and restarts the server, attempting to connect again. This process is repeated *startAttempts* times, at which point the display is declared inactive and disabled. Although this behavior may seem arbitrary, it has been empirically developed and works well on most systems. The default is a value of **5** for *openDelay*, a value of **5** for *openRepeat*, a value of **30** for *openTimeout*, and a value of **4** for *startAttempts*.

DisplayManager.DISPLAY.pingInterval
To discover when remote displays disappear, the **xdm** command occasionally pings them, using an X connection and **XSync** calls. This resource specifies the time (in minutes) between ping attempts. By default, it is set to 5 minutes. If you frequently use X terminals, which can become isolated from the managing host, you may want to increase this value.
Note: AIXwindows sessions may continue to exist after the terminal has been accidentally disabled. The **xdm** command does not ping local displays. A workstation session can be ended if the server hangs for NFS service and does not respond to the ping.

DisplayManager.DISPLAY.pingTimeout
To discover when remote displays disappear, the **xdm** command occasionally pings them, using an X connection and **XSync** calls. This resource specifies the maximum amount of time (in minutes) to wait for the terminal to respond to the request. If the terminal does not respond, the session is declared inactive and ended. By default, it is set to 5 minutes. If you frequently use X terminals, which can become isolated from the managing host, you may want to increase this value.
Note: AIXwindows sessions may continue to exist after the terminal has been accidentally disabled. The **xdm** command does not ping local displays. A workstation session could be ended if the server hangs for NFS service and does not respond to the ping.

DisplayManager.DISPLAY.terminateServer
Specifies whether the X server should be canceled when a session ends (instead of resetting it). This option can be used when the server tends to grow without bound over time, to limit the amount of time the server is run. The default value is False.

DisplayManager.DISPLAY.userPath
The **xdm** command sets the **PATH** environment variable for the session to this value. It should be a list of directories separated by colons; see the **sh** command in *AIX 5L Version 5.3 Commands Reference* for a full description.
:/bin:/usr/bin:/usr/bin/X11:/usr/ucb is a common setting. The default value can be specified at build time in the AIXwindows system configuration file with the **DefaultUserPath** resource.

DisplayManager.DISPLAY.systemPath
The **xdm** command sets the **PATH** environment variable for the startup and reset scripts to the value of this resource. The default for this resource is specified at build time by the **DefaultSystemPath** resource entry in the system configuration file; **/etc:/bin:/usr/bin:/usr/bin/X11:/usr/ucb** is a common choice. Note the absence of . (period) (the current directory) from this entry. This is a good practice to follow for root; it avoids many common ″Trojan Horse″ system penetration schemes.

DisplayManager.DISPLAY.systemShell
The **xdm** command sets the **SHELL** environment variable for the startup and reset scripts to the value of this resource. It is **/bin/sh** by default.

DisplayManager.DISPLAY.failsafeClient
If the default session fails to run, the **xdm** command returns to this program. This program is run with no arguments, using the same environment variables as the session would have had (see the section Session Program). By default, **/usr/bin/X11/xterm** is used.

DisplayManager.DISPLAY.grabServer

DisplayManager.DISPLAY.grabTimeout

To improve security, the **xdm** command grabs the server and keyboard while reading the login name and password. The **grabServer** resource specifies if the server should be held for the duration of the name/password reading. When set to False, the server is ungrabbed after the keyboard grab succeeds, otherwise the server is grabbed until just before the session begins. The default value is False. The **grabTimeout** resource specifies the maximum time that the **xdm** command waits for the grab to succeed. The grab may fail if some other client has the server grabbed, or possibly if the network latencies are high. This resource has a default value of 3 seconds; be cautious when raising it, as a user may be confused by a look-alike window on the display. If the grab fails, the **xdm** command becomes inactive and restarts the server (if possible) and the session.

DisplayManager.DISPLAY.authorize

DisplayManager.DISPLAY.authName

The **authorize** is a Boolean resource that controls whether the **xdm** command generates and uses authorization for the local server connections. If authorization is used, the **xdm** command uses the authorization mechanisms indicated as a whitespace-separated list as the value of the **authName** resource. **XDMCP** connections dynamically specify which authorization mechanisms are supported, so the **authName** resource is ignored in this case. When the **authorize** resource is set for a display and authorization is not available, the user is informed by a different message displayed in the Login widget. By default, the **authorize** resource is True; **authName** is **MIT-MAGIC-COOKIE-1**.

DisplayManager.DISPLAY.authFile

Indicates the file is used to communicate the authorization data from the **xdm** command to the server, using the **-auth** server command-line option. It should be kept in a directory with restricted write permissions as it could easily be removed, disabling the authorization mechanism in the server.

DisplayManager.DISPLAY.authComplain

If set to a value of False, this disables the use of the **unsecureGreeting** in the login window. See the section Authentication Client . The default is a value of True.

DisplayManager.DISPLAY.resetSignal

The number of the signal that the **xdm** command sends to reset the server. See the section Controlling the Server . The default is **1(SIGHUP)**.

DisplayManager.DISPLAY.termSignal

The number of the signal that the **xdm** command sends to end the server. See the section Controlling the Server . The default is **15(SIGTERM)**.

DisplayManager.DISPLAY.resetForAuth

Causes the **xdm** command to send SIGHUP to the server after setting up the authorization file, causing an additional server reset to occur, during which time the new authorization information is read. The default is a value of False, which works for all AIXwindows servers.

DisplayManager.DISPLAY.userAuthDir

When the **xdm** command is unable to write to the usual user authorization file (**$HOME/.Xauthority**), it creates a unique file name in this directory and sets the **XAUTHORITY** environment variable to the name of the created file. It uses **/tmp** by default.

XDMCP Access Control

The database file specified by the **DisplayManager.accessFile** resource provides information that the **xdm** command uses to control access from displays requesting **XDMCP** service. This file contains three types of entries:

- Entries that control the response to **Direct** and **Broadcast** queries.
- Entries that control the response to **Indirect** queries.
- Macro definitions.

Direct query entries contain either a host name or a pattern, which is distinguished from a host name by the inclusion of one or more pattern-matching characters. An * (asterisk) matches any sequence of 0 (zero) or more characters, and a ? (question mark) matches any single character. These are compared against the host name of the display device. If the entry is a host name, all comparisons are done using network addresses, so that any name that converts to the correct network address may be used. For patterns, only actual host names are used in the comparison, so ensure that you do not attempt to match aliases. Preceding either a host name or a pattern with an ! (exclamation point) causes hosts that match that entry to be excluded.

An Indirect entry also contains a host name or pattern, but follows it with a list of host names or macros to which **indirect** queries should be sent.

A macro definition contains a macro name and a list of host names and other macros that the macro expands to. To distinguish macros from host names, macro names start with a % (percent) character. Macros may be nested.

Indirect entries may also specify to have the **xdm** command run the **chooser** command to offer a menu of hosts to which to connect. See the section Chooser on the next page.

When checking access for a particular display host, each entry is scanned in turn and the first matching entry determines the response. For example, a **Direct** query entry is ignored when scanning for an **Indirect** entry. A **Broadcast** query entry is ignored when scanning for a **Direct** entry.

Blank lines are ignored. The **#** character is treated as a comment delimiter causing the rest of that line to be ignored, and a \ (backslash) at the end of the line causes the new line to be ignored, allowing indirect host lists to span multiple lines.

The following is an example **Xaccess** file:

```
#
# Xaccess - XDMCP access control file
#

#
# Direct/Broadcast query entries
#

!xtra.lcs.mit.edu       # disallow direct/broadcast service for xtra
bambi.ogi.edu     # allow access from this particular display
*.lcs.mit.edu     # allow access from any display in LCS

#
# Indirect query entries
#

%HOSTS          expo.lcs.mit.edu xenon.lcs.mit.edu \\
               excess.lcs.mit.edu kanga.lcs.mit.edu

extract.lcs.mit.edu    xenon.lcs.mit.edu    #force extract to contact xenon
!xtra.lcs.mit.edu         dummy     #disallow indirect access
*.lcs.mit.edu     %HOSTS           #all others get to choose
```

Chooser

For X terminals that do not offer a host menu for use with **Broadcast** or **Indirect** queries, the **chooser** program can do this for them. In the **Xaccess** file, specify **CHOOSER** as the first entry in the Indirect host list. The **chooser** program sends a **Query** request to each of the remaining host names in the list and offers a menu of all the hosts that respond.

The list may consist of the word **BROADCAST**, in which case **chooser** sends a **Broadcast** query instead, again offering a menu of all hosts that respond.

The following is an example **Xaccess** file using **chooser**:

```
extract.lcs.mit.edu       CHOOSER   %HOSTS         #offer a menu of these hosts
xtra.lcs.mit.edu          CHOOSER   BROADCAST      #offer a menu of all hosts
```

The program to use for **chooser** is specified by the **DisplayManager.DISPLAY.chooser** resource. Resources for this program can be put into the file named by the **DisplayManager.DISPLAY.resources** resource.

The **chooser** has been implemented using a Motif **SelectionBoxWidget**. Refer to the **XmSelectionBoxWidget Class** documentation for a description of resources and widget or gadget names.

Server Specification

The resource **DisplayManager.servers** gives a server specification or, if the values starts with a / (slash), the name of a file containing server specifications, one per line.

Each specification indicates a display that should constantly be managed and that is not using **XDMCP**. Each consists of at least three parts:

- Display name
- Display class
- Display type
- For local servers, a command line to start the server.

A typical entry for local display number 0 would be:

```
:0 IBM-GT local /usr/bin/X11/X :0
```

The display types are:

local	local display: **\fIxdm\fP** must run the server
foreign	remote display: **\fIxdm\fP** opens an X connection to a running server

The display name must be something that can be passed in the **-display** option to an X program. This string is used to generate the display-specific resource names, so be careful to match the names (for example, use `":0 local /usr/bin/X11/X :0"` instead of `"`localhost:0 local /usr/bin/X11/X :0"` if your other resources are specified as `"DisplayManager._0.session"`). The display class portion is also used in the display-specific resources as the class of the resource. This is useful if you have a large collection of similar displays (like a corral of X terminals) and would like to set resources for groups of them. When using XDMCP, the display is required to specify the display class, so the manual for your particular X terminal should document the display class string for your device. If it does not, you can run the **xdm** command in debug mode and look at the resource strings that it generates for that device, which will include the class string.

Setup Program

The **Xsetup** file is run after the server is reset, but before the login window is offered. The file is typically a shell script. It is run as root, so be careful about security. This is the place to change the root background or bring up other windows that should be displayed on the screen along with the Login widget. Because **xdm** grabs the keyboard, other windows will not be able to receive keyboard input. They will be able to interact with the mouse, however; beware of potential security holes here. If **DisplayManager.DISPLAY.grabServer** is set, **Xsetup** will not be able to connect to the display at all. Resources for this program can be put into the file named by **DisplayManager.DISPLAY.resources**.

In addition to any specified by **DisplayManager.exportList**, the following environment variables are passed:

DISPLAY	Specifies the associated display name.
PATH	Specifies the value of **DisplayManager.DISPLAY.systemPath**.
SHELL	Specifies the value of **DisplayManager.DISPLAY.systemShell**.

XAUTHORITY Specifies that it may be set to an authority file.

Authentication Client

The MIT authentication widget has been replaced by an authentication client composed of standard Motif widgets. The following is a list of the widget names (and their widget class):

```
outframe(xmFrameWidget)
  inframe(xmFrameWidget)
    main(XmFormWidget)
      tframe(xmFrameWidget)
        greeting(xmLabelGadget)
      logoline(xmFormWidget)
        dpyname(xmLabelWidget)
      userline(xmRowColumnWidget)
        userlabel(xmLabelWidget)
        username(xmTextWidget)
        passlabel(xmLabelWidget)
        password(xmTextWidget)
      failsafeline(xmFormWidget)
        failsafe(xmToggleButtonWidget)
      cancelline(xmFormWidget)
        cancel(xmPushButtonWidget)
      message(xmLabelWidget)
```

The authentication client reads a name/password pair from the keyboard. Put resources for this client into the file named by **DisplayManager.DISPLAY.resources**. All of these have reasonable default values, so it is unnecessary to specify any of them. See **/usr/lib/X11/xdm/Xresources** for more information on default values for authentication client resources as well as the appropriate widget class documentation. The following resources are also supported by the authentication client:

Xlogin*foreground Specifies the color used for the foreground.

Xlogin*background Specifies the color used for the background.

Xlogin*greeting Specifies a string that identifies this window. The default is AIXwindows environment.

Xlogin*greetFont Specifies the font used to display the greeting.

Xlogin*frameColor Specifies the background color used to display the greeting.

Xlogin*titleMessage Specifies the string displayed in the title. The default is the hostname of the machine on which the authentication client is running.

Xlogin*titleFont Specifies the font used to display the title.

Xlogin*namePrompt Specifies the string displayed to prompt for a user name. The Xrdb program strips trailing white space from resource values. Add spaces escaped with backslashes at the end of the prompt. The default is ″login:″.

Xlogin*passwdPrompt Specifies the string displayed to prompt for a password. The default is ″password:″.

Xlogin*promptFont Specifies the font used to display both prompts.

Xlogin*failPrompt Specifies the label for the failsafe button.

Xlogin*failFont Specifies the font used for the failsafe button.

Xlogin*cancelPrompt Specifies the label for the cancel button.

Xlogin*cancelFont Specifies the font used for the cancel button.

Xlogin*fail Specifies a message displayed to indicate that the authentication fails. The default is ″Login was incorrect.″

Xlogin*messageFontlist Specifies the font used to display the failure message.

Xlogin*failColor Specifies the color used to display the failure message.

Xlogin*failTimeout Specifies the number of seconds that the failure message is displayed. The default is thirty seconds.

Xlogin*sessionArgument Specifies the argument to be passed to the session program.

Xlogin*XmText.translations

This specifies the translations use for the authentification client. Refer to the X Toolkit documentation for a complete discussion on translations. The default translation table is:

```
Ctrl<Key>b: backward-character()\n\
Ctrl<Key>a: beginning-of-line()\n\
Ctrl<Key>e: end-of-line()\n\
Ctrl<Key>f: forward-character()\n\
Ctrl<Key>d: kill-next-character()\n\
Ctrl<Key>k: kill-to-end-of-line()\n\
Ctrl<Key>u: kill-to-start-of-line()\n
```

You may setup XDM to use the standard XDM translations by replacing the XmText translations as defined in Xresources:
Note: Use `<Key>osfHelp` instead of `<Key>F1` due to the Motif default virtual bindings.)

```
Xlogin*XmText.translations: #override\n\

<Key>osfHelp:    set-session-argument(failsafe) finish-field()\n\
Ctrl<Key>Return: set-session-argument(failsafe) finish-field()\n\
Ctrl<Key>H:      delete-previous-character() \n\
Ctrl<Key>D:      delete-character() \n\
Ctrl<Key>B:      move-backward-character() \n\
Ctrl<Key>F:      move-forward-character() \n\
Ctrl<Key>A:      move-to-beginning() \n\
Ctrl<Key>E:      move-to-end() \n\
Ctrl<Key>K:      erase-to-end-of-line() \n\
Ctrl<Key>U:      erase-line() \n\
Ctrl<Key>X:      erase-line() \n\
<Key>Return:     finish-field() \n
<Key>BackSpace:  delete-previous-character() \n\
<Key>Delete:     delete-previous-character() \n
```

In addition to the typical XmText actions, the following actions are also supported by the client to be compatible with the standard XDM translations:

delete-previous-character
Erases the character before the cursor.

delete-character
Erases the character after the cursor.

move-backward-character
Moves the cursor backward.

move-forward-character
Moves the cursor forward.

move-to-beginning
Moves the cursor to the beginning of the editable text.

move-to-end
Moves the cursor to the end of the editable text.

erase-to-end-of-line
Erases all text after the cursor.

erase-line
Erases the entire text.

finish-field
If the cursor is in the name field, proceeds to the password field; if the cursor is in the password field, checks the current name/password pair. If the name/password pair is valid, **xdm** starts the session. Otherwise the failure message is displayed and the user is prompted again.

insert-char
Inserts the character typed.

set-session-argument
Specifies a single word argument that is passed to the session at startup. See the sections Session Program and Typical Usage.

Startup Program

The **Xstartup** file is typically a shell script. Because it is run as the root user, be careful about security when it runs. It usually contains commands that add entries to **/etc/utmp**, mount users' home directories from file servers, display the message of the day, or cancel the session if logins are not allowed.

In addition to the environment variables specified by **DisplayManager.exportList**, the following variables are passed:

DISPLAY	Specifies the associated display name.
HOME	Specifies the initial working directory of the user.
USER	Specifies the user name.
PATH	Specifies the value of **DisplayManager.DISPLAY.systemPath**.
SHELL	Specifies the value of **DisplayManager.DISPLAY.systemShell**.
XAUTHORITY	May be set to an authority file.

No arguments are passed to the script. The **xdm** command waits until this script exits before starting the user session. If the exit value of this script is nonzero, the **xdm** command discontinues the session and starts another authentication cycle.

Session Program

The **Xsession** program establishes the style of the user's session. It is run with the permissions of the authorized user.

In addition to any specified by **DisplayManager.exportList**, the following environment variables are passed:

DISPLAY	Specifies the associated display name.
HOME	Specifies the initial working directory of the user.
USER	Specifies the user name.
PATH	Specifies the value of **DisplayManager.DISPLAY.userPath**.
SHELL	Specifies the user's default shell (from **getpwnam**).
XAUTHORITY	May be set to a nonstandard authority file.

At most installations, the **Xsession** program should look in the user's home directory (**$HOME**) for a file **.xsession**, which contains the commands that the user would like to use as a session. The **Xsession** program should also implement a system default session if no user-specified session exists. See the section Typical Usage.

An argument may be passed to this program from the authentication widget using the `set-session-argument' action. This can be used to select different styles of session. Usually, this feature is used to allow the user to escape from the ordinary session when it fails. This allows users to repair their own **.xsession** if it fails, without requiring administrative intervention. The section Typical Usage demonstrates this feature.

Reset Program

The **Xreset** script is run after the user session has ended. Run as root, it should contain commands that undo the effects of commands in **Xstartup** by removing entries from **/etc/utmp** or unmounting directories from file servers. The environment variables that are passed to **Xstartup** are also passed to **Xreset**. This program is symmetrical with the **Xstartup** program.

Controlling the Server

The **xdm** command controls local servers using POSIX signals. The **SIGHUP** signal is expected to reset the server, closing all client connections and performing other cleanup duties. The **SIGTERM** signal is expected to cancel the server. If these signals do not perform the expected actions, the resources **DisplayManager.DISPLAY.resetSignal** and **DisplayManager.DISPLAY.termSignal** can specify alternate signals.

To control remote terminals that are not using **XDMCP**, the **xdm** command searches the window hierarchy on the display and uses the protocol request **KillClient** in an attempt to clean up the terminal for the next session. This may not actually cause all of the clients to become inactive, because only those that have created windows will be noticed. **XDMCP** provides a more sure mechanism; when the **xdm** command closes its initial connection, the session is over and the terminal is required to close all other connections.

Controlling XDM

The **xdm** command responds to two signals: **SIGHUP** and **SIGTERM**. When sent a **SIGHUP**, **xdm** rereads the configuration file, the access control file, and the servers file. For the servers file, it notices if entries have been added or removed. If a new entry has been added, the **xdm** command starts a session on the associated display. Entries that have been removed are disabled immediately, meaning that any session in progress is ended without notice and no new session is started.

When sent a **SIGTERM**, the **xdm** command stops all sessions in progress and exits. This can be used when shutting down the system.

The **xdm** command attempts to mark its various subprocesses for use by the **ps** command in *AIX 5L Version 5.3 Commands Reference* by editing the command-line argument list in place. Because the **xdm** command cannot allocate additional space for this task, it is useful to start the **xdm** command with a reasonably long command line (using the full path name should be enough). Each process that is servicing a display is marked **-display**.

Other Possibilities

You can use the **xdm** command to run a single session at a time, using the **xinit** command options or other suitable daemons by specifying the server on the command line:

```
xdm -server ":0 local /usr/bin/X11/X :0 -force"
```

It might also run a file server and a collection of X terminals. The configuration for this is identical to the previous sample, except the **Xservers** file would look like the following:

```
extol:0 VISUAL-19 foreign
exalt:0 NCD-19 foreign
explode:0 NCR-TOWERVIEW3000 foreign
```

This directs the **xdm** command to manage sessions on all three of these terminals. See the section Controlling XDM for a description of using signals to enable and disable these terminals.

Note: The **xdm** command does not coexist well with other window systems. To use multiple window systems on the same hardware, use the **xinit** command.

Examples

1. The sample **xstartup** script that follows prevents login while the file **/etc/nologin** exists. As there is no provision for displaying any messages here (there is no core X client that displays files), the setup in this example is not recommended because the login would fail without explanation. Thus, this is not a complete example, but a demonstration of the available functionality.

```
#!/bin/sh
#
# Xstartup
#
# This program is run as root after the user is verified
#
if [ \-f /etc/nologin ]; then
        exit 1
fi
exit 0
```

2. This **Xsession** script recognizes the special **failsafe** mode, specified in the translations in the preceding **Xresources** file, to provide an escape from the ordinary session:

```
#!/bin/sh
exec > $HOME/.xsession-errors 2>&1
case $# in
1)
        case $1 in failsafe)
                exec aixterm -geometry 80x24-0-0
                ;;
        esac
esac
startup=$HOME/.xsession
resources=$HOME/.Xresources
if [ -f /usr/bin/X11/startx ]; then
       exec /usr/bin/X11/startx -t -wait
elif [ -f $startup]; then
       exec $startup
else
       if [ -f $resources ]; then
               xrdb -load $resources
       fi
       mwm &
       exec aixterm -geometry 80x24+10+10 -ls
fi
```

3. To have **xdm** come up from system startup, as root type the following:

```
/usr/lib/X11/xdm/xdmconf
```

4. To disable **xdm** on reboot, as root type the following:

```
/usr/lib/X11/xdm/xdmconf -d
```

5. When using **xdm** to manage your display, an authentication procedure insures that only clients that are allowed can connect to your display. Clients that are built using X11 R4 and X11 R5 libraries understand this protocol. Clients that are built with X11 R3 or earlier libraries do not support this authentication protocol and are not allowed to connect to the Xserver unless **xhost** permission is granted. You can connect local clients by typing the following:

```
xhost =localhost
```

or

```
xhost =machine
```

where *machine* is the hostname of the local client.

Files

/usr/lib/X11/xdm/xdm-config	The default configuration file.
/usr/lib/X11/xdm/Xaccess	The default access file, listing authorized displays.
/usr/lib/X11/xdm/Xservers	The default server file, listing non-XDMCP servers to manage.
$(HOME)/.Xauthority	User authorization file where **xdm** stores keys for clients to read.
/usr/lib/X11/xdm/chooser	The default chooser.
/usr/bin/X11/xrdb	The default resource database loader.
/usr/bin/X11/X	The default server.
/usr/bin/X11/xterm	The default session program and failsafe client.
/usr/lib/X11/xdm/A<host>\-<suffix>	The default place for authorization files.

Related Information

The **X** command, **xinit** command, **startx** command.

---

xfindproxy Command

Purpose

Locates proxy services.

Syntax

xfindproxy -manager *ManagerAddress* **-name** *ServiceName* **-server** *ServerAddress* [**-auth**] [**-host** *HostAddress*] [**-options** *Options*]

Description

xfindproxy is a program used to locate available proxy services. It utilizes the Proxy Management Protocol to communicate with a proxy manager. The proxy manager keeps track of all available proxy services, starts new proxies when necessary, and makes sure that proxies are shared whenever possible.

If **xfindproxy** is successful in obtaining a proxy address, it will print it to stdout. The format of the proxy address is specific to the proxy service being used. For example, for a proxy service of `LBX`, the proxy address would be the X display address of the proxy (e.g, `blah.x.org:63`).

If **xfindproxy** is unsuccessful in obtaining a proxy address, it will print an error to **stderr**.

Flags

-manager	This argument is required, and it specifies the network address of the proxy manager. The format of the address is a standard ICE network id (for example, `tcp/blah.x.org:6500`).
-name	This argument is required, and it specifies the name of the desired proxy service (for example, `LBX`). The name is case insensitive.
-server	This argument is also required, and it specifies the address of the target server. The format of the address is specific to the proxy service specified with the **-name** argument. For example, for a proxy service of `LBX`, the address would be an X display address (e.g, `blah.x.org:0`).

-auth This argument is optional. If specified, **xfindproxy** will read 2 lines from standard input. The first line is an authorization/authentication name. The second line is the authorization/authentication data in hex format (the same format used by xauth). **xfindproxy** will pass this auth data to the proxy, and in most cases, will be used by the proxy to authorize/authenticate itself to the target server.

-host This argument is optional. If **xfindproxy** starts a new proxy service, it will pass the host specified. The proxy may choose to restrict all connections to this host. In the event that **xfindproxy** locates an already existing proxy, the host will be passed, but the semantics of how the proxy uses this host are undefined.

-options This argument is optional. If **xfindproxy** starts a new proxy service, it will pass any options specified. The semantics of the options are specific to each proxy server and are not defined here. In the event that **xfindproxy** locates an already existing proxy, the options will be passed, but the semantics of how the proxy uses these options are undefined.

Related Information

The **proxymngr** command.

---

xfs Command

Purpose

Supplies fonts to X Window System display servers.

Syntax

xfs [**-config** *ConfigurationFile*] [**-ls** *ListenSocket*] [**-port** *Number*]

Description

xfs is the AIXwindows font server. It supplies fonts to AIXwindows display servers.

The **xfs** server responds to the following signals:

SIGTERM Causes the font server to exit cleanly.
SIGUSR1 Causes the server to re-read its configuration file.
SIGUSR2 Causes the server to flush any cached data it may have.
SIGHUP Causes the server to reset, closing all active connections and re-reading the configuration file.

The server is usually run by a system administrator, and started by way of boot files such as **/etc/rc.tcpip**. Users may also wish to start private font servers for specific sets of fonts.

The configuration language is a list of keyword and value pairs. Each keyword is followed by an = (equal sign) and the desired value.

The following list shows recognized keywords and the types and descriptions of valid values:

A comment character when located in the first column.

catalogue (List of string)
Ordered list of font path element names. The current implementation only supports a single catalogue (″all″), containing all of the specified fonts.

alternate-servers (List of string)
List of alternate servers for this font server.

client-limit (Cardinal)
Number of clients that this font server will support before refusing service. This is useful for tuning the load on each individual font server.

clone-self (Boolean)
Whether this font server should attempt to clone itself when it reaches the client-limit.

default-point-size (Cardinal)
The default point size (in decipoints) for fonts that do not specify.

default-resolutions (List of resolutions)
Resolutions the server supports by default. This information may be used as a hint for pre-rendering and substituted for scaled fonts which do not specify a resolution.

A resolution is a comma-separated pair of **x** and **y** resolutions in pixels per inch. Multiple resolutions are separated by commas.

error-file (String)
Filename of the error file. All warnings and errors are logged here.

port (Cardinal)
TCP port on which the server will listen for connections. The default is 7100.

use-syslog (Boolean)
Whether the **syslog** function (on supported systems) is to be used for errors.

deferglyphs (String)
Set the mode for delayed fetching and caching of glyphs. Value is `none`, meaning deferred glyphs is disabled. `all`, meaning deferred glyphs is enabled for all fonts, and `16` , meaning deferred glyphs is enabled only for 16-bit fonts.

One of the following forms can be used to name a font server that accepts TCP connections:

```
tcp/hostname:port
tcp/hostname:port/cataloguelist
```

The hostname specifies the name (or decimal numeric address) of the machine on which the font server is running. The port is the decimal TCP port on which the font server is listening for connections. The cataloguelist specifies a list of catalogue names, with '+' as a separator. The following are some examples:

```
tcp/expo.lcs.mit.edu:7100, tcp/18.30.0.212:7101/all
```

One of the following forms can be used to name a font server that accepts DECnet connections:

```
decnet/nodename::font$objname
decnet/nodename::font$objname/cataloguelist
```

The nodename specifies the name (or decimal numeric address) of the machine on which the font server is running. The objname is a normal, case-insensitive DECnet object name. The cataloguelist specifies a list of catalogue names, with '+' as a separator.

Flags

-config *ConfigurationFile*
Specifies the configuration file the font server will use.

-ls *ListenSocket*
Specifies a file descriptor that is already set up to be used as the listen socket. This option is only intended to be used by the font server itself when automatically spawning another copy of itself to handle additional connections.

-port *Number* Specifies the TCP port number on which the server will listen for connections.

Examples

```
#
# sample font server configuration file
#
# allow a max of 10 clients to connect to this font server
client-limit = 10
# when a font server reaches its limit, start up a new one
clone-self = on
# alternate font servers for clients to use
alternate-servers = hansen:7101,hansen:7102
# where to look for fonts
# the first is a set of Speedo outlines, the second is a set of
# misc bitmaps and the last is a set of 100dpi bitmaps
#
catalogue = /usr/lib/fonts/type1,
   /usr/lib/X11/ncd/fonts/misc,
   /usr/lib/X11/ncd/fonts/100dpi/
# in 12 points, decipoints
default-point-size = 120
# 100 x 100 and 75 x 75
default-resolutions = 100,100,75,75
```

Files

/usr/lib/X11/fs/config The default configuration file.

---

xget Command

Purpose

Receives secret mail in a secure communication channel.

Syntax

xget

Description

The **xget** command is used to receive secret mail in a secure communication channel. The messages can be read only by the intended recipient. The **xget** command asks for your password and enables you to read your secret mail.

The **xget** command is used with the **enroll** command and the **xsend** command to send and receive secret mail. The **enroll** command sets up the password used to receive secret mail. The **xsend** command sends mail that can be read only by the intended recipient.

When you issue the **xget** command, you are prompted for your encryption key. Enter the password you previously set up using the **enroll** command.

The prompt for the **xget** command is a ? (question mark). The following subcommands control message disposition:

q (quit)	Writes any mail not yet deleted to the user's mailbox and exits. Pressing End Of File (Ctrl-D) has the same effect.
n (delete) or **d** (delete) or **Enter**	Deletes the current message and displays the next message.
!*Command*	Runs the specified workstation command.
s[*Filename*]	Saves the message in the named *File* parameter instead of in the default mail file, **mbox**.
w[*Filename*]	Saves the message, without its header, in the specified *File* parameter instead of in the default mail file **mbox**.
? (help)	Displays a subcommand summary.

Examples

1. To receive secret mail, enter:

   ```
   xget
   ```

   You are prompted for the password, established with the **enroll** command. After entering your password, the **xget** command prompt (?) and a listing of any secret mail is displayed.
2. To display your secret mail, at the **xget** prompt (?), press the Enter key.

   After the most recent message is displayed, a ? (question mark) indicates the **xget** command is waiting for one of the **xget** subcommands. Enter `help` or a ? (question mark) to list the subcommands available.
3. To save a message or a file to the default mail file, enter:

   ```
   xget
   ```

   Press the Enter key after the ? (question mark) prompt until the desired file is displayed. When the appropriate file is displayed, enter:

   ```
   s
   ```

   In this example, the file is saved in the default mail file, **mbox**.
4. To save a message or a file to a specific file, enter:

   ```
   xget
   ```

   Press the Enter key after the ? (question mark) prompt until the desired file is displayed. When the appropriate file is displayed, enter:

   ```
   s mycopy
   ```

   In this example, the file is saved in a file named `mycopy`, instead of the default mail file.
5. To delete a message, enter:

   ```
   xget
   ```

   Press the Enter key after the ? (question mark) prompt until the desired file is displayed. When the appropriate file is displayed, enter:

   ```
   d
   ```

   In this example, the current file is deleted.

Files

/var/spool/secretmail/*User***.key**

	Contains the encrypted key for *User*.
/var/spool/secretmail/*User***.[0-9]**	Contains the encrypted mail messages for *User*.
/usr/bin/xget	Contains executable files.

Related Information

The **enroll** command, **mail** command, **xsend** command.

Mail applications in *Networks and communication management*.

Sending and receiving secret mail in *Networks and communication management*.

Mail management in *Networks and communication management*.

---

xhost Command

Purpose

Controls who accesses Enhanced X-Windows on the current host machine.

Syntax

xhost [**+** | **-**] [*Name*]

Description

The **xhost** command adds or deletes host names on the list of machines from which the X Server accepts connections.

This command must be run from the machine with the display connection. You can remove a name from the access list by using the **-***Host* parameter. Do not remove the current name from the access list. If you do, log off the system before making any corrections.

Entering the **xhost** command with no variables shows the current host names with access your X Server and a message indicating whether or not access is enabled.

For security, options that affect access control may only be run from the *controlling host*. For workstations, this is the same machine as the server. For X terminals, it is the login host.

To enable a remote name by default, the name can be defined in the **/etc/X***?***.hosts** file, where *?* is the display number to which you enable access.

For example, the display `jeanne:0` can be accessed by systems defined in the **/etc/X0.hosts** file on a system that uses the default host name of `jeanne`. In both the display name and the file name, 0 indicates the display number that the defined remote systems are allowed to access through Enhanced X-Windows.

Flags

+*Name*	Defines the host name (the plus sign is optional) to be added to the X Server access list.
- *Name*	Defines the host name to be removed from the X Server access list. Existing connections are not broken, but new connection attempts will be denied. Note that you can remove the current machine; however, further connections (including attempts to add it back) are not permitted. The only way to allow local connections again is to reset the server (thereby breaking all connections).
+	Specifies that access is unlimited. Access control is turned off.

- Turns access control on.

The complete *Name* has a the following *family*:*name* syntax:

inet Internet host

local Contains only one name, the empty string

Note: The family is case sensitive. The format of the name varies with the family.

xinit Command

Purpose

Initializes the X Window System.

Syntax

xinit [[*Client*] *Options*] [- - [*Server*] [*Display*] *Options*]

Description

The **xinit** command starts the AIXwindows server and a first client program on systems that cannot start X directly from **/etc/init** or in environments that use multiple window systems. When this first client exits, the **xinit** command stops the X server and then ends.

If no specific client program is given on the command line, the **xinit** command looks for a file to run to start up client programs. The **xinit** command looks for the **$XINITRC** environment variable. If the file is not there, it then looks for the **$HOME/.xinitrc** file. If it still does not find the file, it follows these steps:

1. The **xinit** command looks next to **/usr/lib/X11/$LANG/xinitrc.**
2. Next, it looks to **/usr/lpp/X11/defaults/$LANG/xinitrc**.
3. And finally, it looks to **/usr/lpp/X11/defaults/xinitrc**.

If no such file exists, **xinit** uses the following as a default:

```
aixterm \-geometry +1+1 \-n login \-display :0
```

If no specific server program is given on the command line, the **xinit** command follows these steps:

1. The **xinit** command looks for a file to run as a shell script to start up the server. The **xinit** command looks for files first in the **$XSERVERRC** environment variable.
2. If the file is not there, it looks for the **$HOME/.xserverrc** file.
3. If it still does not find the **$HOME/.xserverrc** file, it looks next to **/usr/lpp/X11/defaults/xserverrc** file.
4. And finally, if it does not find any of the previous files, the **xinit** command runs the **X** command to start the X server and uses the following as a default:

   ```
   X :0
   ```

Note that this assumes that there is a program named X in the current search path. However, servers are usually named X*displaytype* where *displaytype* is the type of graphics display which is driven by this server. The site administrator should, therefore, make a link to the appropriate type of server on the machine, or create a shell script that runs the **xinit** command with the appropriate server.

Note: If you attempt to start AIXwindows without an available pointer device, such as a mouse or a tablet, AIXwindows will not open. Some devices can be plugged in but not defined and thus not available to the system, as well as the reverse.

An important point is that programs which are run by **.xinitrc** should be run in the background if they do not exit right away, so that they do not prevent other programs from starting up. However, the last long-lived program started (usually a window manager or terminal emulator) should be left in the foreground so that the script does not exit (which indicates that the user is done and that xinit should exit).

An alternate client and/or server may be specified on the command line. The desired client program and its arguments should be given as the first command line arguments to **xinit**. To specify a particular server command line, add a — (double dash) to the **xinit** command line (after any client and arguments) followed by the desired server command.

Both the client program name and the server program name must begin with a / (slash) or a . (period). Otherwise, they are treated as an arguments to be added to their respective startup lines. This makes it possible to add arguments (for example, foreground and background colors) without having to retype the whole command line.

If a clear server name is not given and the first argument following the — (double dash) is a : (colon) followed by a number, **xinit** uses that number as the display number instead of zero. All remaining arguments are added to the server command line.

The following environment variables are used with the **xinit** command:

DISPLAY	This variable gets set to the name of the display to which clients should connect.
XINITRC	This variable specifies an init file containing shell commands to start up the initial windows. By default, **.xinitrc** in the home directory is used.
Options	List any option you wish that is available to the client you specified.
Client	Specify the client with which you are working. For example, xterm or aixterm. The client you specify must begin with a . (dot) or a / (slash).
Server	Use any valid xserver. The server you specify must begin with a . (dot) or a / (slash).

Examples

1. To start up a server named X and run the user's **xinitrc** program, if it exists, or else start an **aixterm** command enter:

   ```
   xinit
   ```

2. To start a specific type of server on an alternate display, enter:

   ```
   xinit -- /usr/bin/X11/X qdss:1
   ```

3. To start up a server named X, and add the given arguments to the default **xinitrc** or **aixterm** command, enter:

   ```
   xinit -geometry =80x65+10+10 -fn 8x13 -j -fg white -bg navy
   ```

4. To use the command **/Xsun -l -c** to start the server and add the arguments **-e widgets** to the default **xinitrc** or **aixterm** command, enter:

   ```
   xinit -e widgets -- ./Xsun -l -c
   ```

5. To start a server named X on display 1 with the arguments **-a 2 -t 5**, then start a remote shell on the machine **fasthost** in which it runs the command **cpupig**, telling it to display back on the local workstation, enter:

   ```
   xinit /usr/ucb/rsh fasthost cpupig -display ws:1 -- :1 -a 2 -t 5
   ```

6. The following sample of the **.xinitrc** script starts a clock, several terminals, and leaves the window manager running as the last application. Assuming that the window manager has been configured properly, the user then chooses the **Exit** menu item to end the AIXwindows session.

   ```
   xrdb -load $HOME/.Xresources
   xsetroot -solid gray &
   xclock -g 50x50-0+0 -bw 0 &
   ```

```
xload -g 50x50-50+0 -bw 0 &
xterm -g 80x24+0+0 &
xterm -g 80x24+0-0 &
mwm
```

7. Sites that want to create a common startup environment could simply create a default **.xinitrc** script that references a site-wide startup file:

   ```
   #!/bin/sh . /usr/local/lib/site.xinitrc
   ```

8. Another approach is to write a script that starts the **xinit** command with a specific shell script. Such scripts are usually named `x11`, `xstart`, or `startx` and are a convenient way to provide a simple interface for novice users:

   ```
   #!/bin/sh xinit /usr/local/lib/site.xinitrc -- /usr/bin/X11/X bc
   ```

Files

.xinitrc	Contains the default client script files.
aixterm	Contains the command the client runs if **.xinitrc** does not exist.
.xserverrc	Contains the default server script.
X	Contains the command the server runs if **.xserverrc** does not exist.

Related Information

The **startx** command, **X** command.

---

xkbcomp Command

Purpose

Compiles XKB keyboard description.

Syntax

xkbcomp [**-a**] [**-C**] [**-dflts**] [**-I** *Directory*] [**-l**] [**-m** *Name*] [**-merge**] [**-o** *OutputFile*] [**-opt** *Parts*] [**-R** *Directory*] [**-synch**] [**-w** *Level*] [**-xkb**] [**-xkm**] *Source* [*Destination*]

Description

The **xkbcomp** command is a keymap compiler that converts a description of an XKB keymap into one of several output formats. The most common use for **xkbcomp** is to create a compiled keymap file (**.xkm** extension) which can be read directly by XKB-capable X servers or utilities. The keymap compiler can also produce C header files or XKB source files. The C header files produced by **xkbcomp** can be included by X servers or utilities that need a built-in default keymap. The XKB source files produced by **xkbcomp** are fully resolved and can be used to verify that the files which typically make up an XKB keymap are merged correctly or to create a single file which contains a complete description of the keymap.

The *Source* may specify an X display, or an **.xkb** or **.xkm** file; unless explicitly specified, the format of *destination* depends on the format of the source. Compiling a **.xkb** (keymap source) file generates a **.xkm** (compiled keymap file) by default. If the source is a **.xkm** file or an X display, **xkbcomp** generates a keymap source file by default.

If the *Destination* is an X display, the keymap for the display is updated with the compiled keymap.

The name of the *destination* is usually computed from the name of the source, with the extension replaced as appropriate. When compiling a single map from a file which contains several maps, **xkbcomp** constructs the destination file name by appending an appropriate extension to the name of the map to be used.

Flags

-a	Shows all keyboard information, reporting implicit or derived information as a comment. Only affects **.xkb** format output.
-C	Produces a C header file as output (**.h** extension).
-dflts	Computes the defaults for any missing components, such as key names.
-I *Directory*	Specifies the top-level directories to be searched for files included by the keymap description.
-l	List maps that specify the *map* pattern in any files listed on the command line.
-m *Name*	Specifies a map to be compiled from an file with multiple entries.
-merge	Merges the compiled information with the map from the server.
-o *Name*	Specifies a name for the generated output file. The default is the name of the source file with an appropriate extension for the output format.
-opt *Parts*	Specifies a list of optional parts. Compilation errors in any optional parts are not fatal. *Parts* may consist of any combination of the letters **c**, **g**, **k**, **s**, **t** which specify the **c**ompatibility map, **g**eometry, **k**eycodes, **s**ymbols, and **t**ypes, respectively.
-R *Directory*	Specifies the root directory for relative path names.
-synch	Forces synchronization for X requests.
-w *Level*	Controls the reporting of warnings during compilation. A warning level of 0 disables all warnings; a warning level of 10 enables them all.
-xkb	Generates a source description of the keyboard as output (**.xkb** extension).
-xkm	Generates a compiled keymap file as output (**.xkm** extension).

Related Information

The **X** command.

---

xkbevd Daemon

Purpose

XKB event daemon.

Syntax

xkbevd [**-help**] [**-cfg** *File*] [**-sc** *Command*] [**-sd** *Directory*] [**-display** *Display*] [**-bg**] [**-synch**] [**-v**]

Description

The **xkbevd** event daemon listens for specified XKB events and executes requested commands if they occur. The configuration file consists of a list of event specification/action pairs and/or variable definitions.

An event specification consists of a short XKB event name followed by a string or identifier which serves as a qualifier in parentheses; empty parenthesis indicate no qualification and serve to specify the default command which is applied to events which do not match any of the other specifications. The interpretation of the qualifier depends on the type of the event:

- Bell events match using the name of the bell.

- Message events match on the contents of the message string.
- Slow key events accept any of **press**, **release**, **accept**, or **reject**.

No other events are recognized.

An action consists of an optional keyword followed by an optional string argument. **xkbevd** recognizes the following actions:

- **none**
- **ignore**
- **echo**
- **printEvent**
- **sound**
- **shell**

If the action is not specified, the string is taken as the name of a sound file to be played unless it begins with an exclamation point, in which case it is taken as a shell command.

Variable definitions in the argument string are expanded with fields from the event in question before the argument string is passed to the action processor. The general syntax for a variable is either:

$*c*

or

$**(***str***)**

where *c* is a single character and *str* is a string of arbitrary length. All parameters have both single-character and long names. The list of recognized parameters varies from event to event.

The **ignore**, **echo**, **printEvent**, **sound**, and **shell** actions do what you would expect commands named **ignore**, **echo**, **printEvent**, **sound**, and **shell** to do, except that the **sound** command has only been implemented and tested for SGI machines.

The only currently recognized variables are *soundDirectory* and *soundCommand*.

Flags

-bg	Tells **xkbevd** to fork itself and run in the background.
-cfg *File*	Specifies the configuration file to read. If no configuration file is specified, **xkbevd** looks for **~/.xkb/xkbevd.cf** and **$(LIBDIR)/xkb/xkbevd.cf** in that order.
-display *Display*	Specifies the display to use. If not present, **xkbevd** uses **$DISPLAY**.
-help	Prints a usage message.
-sc *Command*	Specifies the command used to play sounds.
-sd *Directory*	Specifies a top-level directory for sound files.
-synch	Forces synchronization of all X requests. Slow.
-v	Prints more information, including debugging messages. Multiple specifications of **-v** causes more output.

Related Information

The **X** command.

xkbprint Command

Purpose

Prints an XKB keyboard description.

Syntax

xkbprint [**-?** | **-help**] [**-color**] [**-dflts**] [**-diffs**] [**-eps**] [**-fit**] [**-full**] [**-grid** *Resolution*] [**-if** *FontName*] [**-label** *Type*] [**-lc** *Locale*] [**-level1**] [**-level2**] [**-lg** *Group*] [**-ll** *Level*] [**-mono**] [**-n** *Number*] [**-nkg** *Number*] [**-npk** *Number*] [**-o** *File*] [**-R** *Directory* [**-pict** *Which*]] *Source* [*OutputFile*]

Description

The **xkbprint** command generates a printable or encapsulated PostScript description of the XKB keyboard description specifed by *Source*. The *Source* can be any compiled keymap, **.xkm** file, that includes a geometry description or an X display specification. If an *OutputFile* is specified, **xkbprint** writes to it. Otherwise, **xkbprint** creates the output file, replacing the extension of the source file with **.ps** or **.eps** depending on the requested format. If the source is a non-local X display, for example `:0`, **xkbprint** appends the appropriate prefix to the display specification, replacing the colon with a **-** (dash). For a local display, **xkprint** uses `server-`*n* where *n* is the number of the display.

Flags

-? | **-help**
Prints a usage message.

-color
Prints using the colors specified in the geometry file; by default, **xkbprint** prints a black-and-white image of the keyboard.

-dflts
Attempts to compute default names for any missing components, such as keys.

-diffs
Shows symbols only where they are explicitly bound.

-eps
Generates an encapsulated PostScript file.

-fit
Fits the keyboard image on the page, this is the default.

-full
Prints the keyboard at full size.

-grid *Resolution*
Prints a grid with *Resolution***mm** resolution over the keyboard.

-if *FontName*
Specifies an internal PostScript type 1 font to dump to the specified output file or to *fontName***.pfa**, if no output file is specified. No keyboard description is printed if an internal font is dumped.

-label *Type*
Specifies the labels to be printed on keys. Valid types are:

- **none**
- **name**
- **code**
- **symbols**

-lc *Locale*
Specifies a locale in which KeySyms should be resolved.

-level1
Generates a level 1 PostScript.

-level2
Generates a level 2 PostScript.

-lg *Group*
Prints symbols in keyboard groups starting from *Group*.

-ll *Level*
Prints symbols starting from shift level *Level*.

-mono
Generates a black-and-white image of keyboard, this is the default.

-n *Number*
Prints *Number* of copies.

-nkg *Number*
Prints the symbols in *Number* keyboard groups.

-npk *Number*
Specifies the *Number* of keyboard images to print on each page. For EPS files, this specifies the total number of keyboard images to print.

-o *File*	Writes the output to *File*.
-R *Directory*	Use *Directory* as the root directory; all path names are interpreted relative to *Directory*.
-pict *Which*	Controls the use of pictographs instead of keysym names where available. Valid values for *Which* are: • **all** • **none** • **common** (default).
-synch	Forces synchronization for X requests.
-w *Level*	Sets warning level. • 0 for no warning • 10 for all warnings

Related Information

The **X** command and **xkbcomp** command.

---

xlock Command

Purpose

Locks the local X display until a password is entered.

Syntax

xlock [**-batchcount** *Number*] [**-bg** *Color*] [**-delay** *Users*] [**-display** *Display*] [**-fg** *Color*] [**-font** *FontName*] [**-info** *TextString*] [**-invalid** *TextString*] [**-mode** *ModeName*] [**+mono** | **-mono**] [**-username** *TextString*] [**-nice** *Level*] [**+nolock** | **-nolock**] [**-password** *TextString*] [**+remote** | **-remote**] [**+allowaccess** | **-allowaccess**] [**+allowroot** | **-allowroot**] [**+echokeys** | **-echokeys**] [**+enablesaver** | **-enablesaver**] [**-help**] [**-saturation** *Value*] [**-timeout** *Seconds*] [**+usefirst** | **-usefirst**] [**+v** | **-v**] [**-validate** *TextString*]

Description

The **xlock** command locks the X server until the user enters a password at the keyboard. While the **xlock** command is running, all new server connections are refused. The screen saver is disabled, the mouse cursor is turned off, the screen is blanked, and a changing pattern is displayed. If a key or a mouse button is pressed, a prompt asks for the password of the user who started the **xlock** command**.**

If the correct password is typed, the screen is unlocked and the X server is restored. When typing the password, Ctrl-U and Ctrl-H are active as kill and erase, respectively. To return to the locked screen, click in the small icon version of the changing pattern.

To function properly, **xlock** needs to run with root permission since the operating system restricts access to the password and access control files. To give **xlock** root permission, perform the following steps:

1. Log in as root.
2. Go to the directory that contains the **xlock** program file.
3. Run these two commands:
   a. **chown root xlock**
   b. **chmod u+s xlock**

Flags

-batchcount *Number*
Sets the number of things to do per batch. *Number* refers to different things depending on the mode:

qix Refers to the number of lines rendered in the same color.

hop Refers to the number of pixels rendered in the same color.

image Refers to the number of sunlogos on screen at once.

swarm Refers to the number of bees

life and blank
Does not apply.

-bg *Color*
Sets the color of the background on the password screen.

-delay *Number*
Sets the speed at which a mode operates to the number of microseconds to delay between batches of **hopalong** pixels, **qix** lines, **life** generations, **image** bits, and **swarm** motions.

In the **blank** mode, it is important to set this to a small number because the keyboard and mouse are only checked after each delay. A delay of zero would needlessly consume the processing unit while checking for mouse and keyboard input in a tight loop since the **blank** mode has no work to do.

-display *Display*
Sets the X11 display to lock. The **xlock** command locks all available screens on the server and restricts you to locking only a local server, such as **unix:0**, **localhost:0**, or **:0** (unless you set the **-remote** flag).

-fg *Color*
Sets the color of the text on the password screen.

-font *FontName*
Sets the font to be used on the prompt screen.

-help
Prints a brief description of available options.

-info *TextString*
Defines an informational message. The default is `Enter password to unlock; select icon to lock`.

-invalid *TextString*
Specifies an password message. The default is `Invalid login`.

-mode *ModeName*
Specifies one the following six display modes:

blank Displays a black screen.

hop Displays the real plane fractals from the September, 1986 issue of *Scientific American*.

image Displays several randomly appearing sun logos.

life Displays Conway's game of life.

qix Displays spinning lines.

swarm Displays a swarm of bees following a wasp.

-nice *NiceLevel*
Sets system nicelevel of the **xlock** process.

-password *TextString*
Specifies the password prompt string. The default is `Password:`.

-saturation *Value*
Sets saturation of the color ramp. A value of 0 (zero) is grayscale and a value of 1 is very rich color. A value of 0.4 is a medium pastel.

-timeout *Seconds*
Sets the number of seconds before the password screen times out.

-username *TextString*
Specifies the message shown in front of the user name. The default is `Name:`.

-validate *TextString*
Specifies the message that is shown while validating the password. The default is `Validating login...`.

-/+allowaccess
Allows the disabling of the access control list, but still causes the local server to prompt for a password. If **xlock** is killed using the **-KILL** command, the access control list is not lost.

This flag is also needed when running the **xlock** command remotely since access to the control list is restricted.

-/+allowroot
Allows the root password to unlock the server as well as the user who started the **xlock** command.

-/+echokeys	Causes the **xlock** command to echo to screen a '?' (question mark) character for each key typed into the password prompt. The default is no echo.
+/-enablesaver	Enables the default screensaver. It is possible to set delay parameters long enough to cause phosphor burn on some displays. This flag can be used as an added precaution.
+/-mono	Causes the **xlock** command to display monochrome (black and white) pixels rather than the default colored ones on color displays.
+/-nolock	Causes the **xlock** command to only draw the patterns and not to lock the display. A keypress or a mouse click terminates the screen saver.
+/-remote	Allows remote locking of X11 servers. This flag should be used with care. It is intended mainly to lock X11 terminals that cannot run the **xlock** command locally. If you lock a workstation other than your own, that person will need your password to unlock it.The **-remote** option does not disable your ability to toggle to another shell.
+/-usefirst	Allows using the keystroke which obtained the password screen as the first input character in the password. The default ignores the first keystroke.
+/-v	Minus prefix enables the verbose mode to tell which options the **xlock** command is going to use. The plus prefix is the default.

xlsfonts Command

Purpose

Displays the font list for X-Windows.

Syntax

xlsfonts [**-display** *Host*:*Display*] [**-l** [**l** [**l**]]] [**-m**] [**-C**] [**-1**] [**-w** *Width*] [**-n** *Columns*] [**-u**] [**-o**] [**-fn** *Pattern*]

Description

The **xlsfonts** command lists the fonts that match a specified *Pattern* parameter. Use the wildcard character ″*″ (asterisk) to match any sequence of characters (including none), and the ″?″ (question mark) to match any single character. If no pattern is given, ″*″ is assumed.

> **Note:** The ″*″ and ″?″ characters must be placed within quotation marks to prevent them from being expanded by the shell.

You can use flags to specify servers, number and width of columns to print, size of font listings, whether the output should be sorted, and whether to use **OpenFont** instead of **ListFonts**.

Flags

> **Note:** Using the **-l** (lowercase L) flag of the **xlsfonts** command can tie up your server for a long time. This is typical of single-threaded non-preemptable servers, and not a program error.

-1	Indicates that listings should use a single column. This flag is the same as the **-n** *1* flag.
-C	Indicates that listings should use multiple columns. This flag is the same as the **-n** *0* flag.
-display *Host*:*Display*	Identifies the X Server to contact by specifying the host name and display number.
-fn *Pattern*	Specifies the fontname *Pattern* that **xlsfonts** will list.
-l [**l** [**l**]]	(lowercase L) Indicates that medium, long, and very long listings, respectively, should be generated for each font.

-m	Indicates that long listings should also print the minimum and maximum bounds of each font.
-n *Columns*	Specifies the number of columns to use to display the output. By default, the **xlsfonts** command tries to fit as many columns of font names into the number of characters specified by the **-w** *Width* flag.
-o	Instructs the **xlsfonts** command to perform **OpenFont** (and **QueryFont**, if appropriate) instead of **ListFonts**. The **-o** flag is useful if the **ListFonts** or **ListFontsWithInfo** fails to list a known font, as is the case with some scaled font systems.
-u	Indicates that the output should remain unsorted.
-w *Width*	Specifies the width in characters that should be used to determine how many columns to print. The default is 79.

Environment Variable

DISPLAY Gets the default host and display to use.

Examples

1. To specify a medium-sized list of each font, use a lowercase L and enter:
   ```
   xlsfonts -l
   ```
2. To specify a three-column list of each font, enter:
   ```
   xlsfonts -n 3
   ```
3. To display all fonts with the string iso8859 within their names, enter:
   ```
   xlsfonts -ll "*"iso8859"*"
   ```
4. To list all fonts with rom1 plus one following character in their names, enter:
   ```
   xlsfonts rom1"?"
   ```

   This obtains a listing similar to:
   ```
   rom10  rom11  rom14  rom16  rom17
   ```

Related Information

The **X** command, **xset** command.

---

xmbind Command

Purpose

Configures virtual key bindings.

Syntax

xmbind [**-display** *Host:Display:ScreenID*] [*FileName*]

Description

The **xmbind** command is an X Windows System client that configures the virtual key bindings for AIXwindows applications. This action is performed by the **mwm** command at its startup, so the **xmbind** client is only needed when **mwm** is not in use or when you want to change bindings without restarting **mwm**. If a file is specified, its contents are used as the virtual key bindings. If a file is not specified, the **.motifbind** file in the user's home directory is used. If this file is not found, the **xmbind** command loads the default virtual key bindings.

Flags

-display *Host*:*Display*:*ScreenID* Specifies the display to use. The **-display** option has the following parameters:

Host Specifies the host name of a valid system on the network. Depending on the situation, this could be the host name of the user or the host name of a remote system.

Display Specifies the number (usually 0) of the display on the system on which the output is to be displayed.

ScreenID
Specifies the number of the screen where the output is to be displayed. This number is 0 for single-screen systems.

Parameters

FileName Specifies the file containing bindings for virtual mouse and key events.

Exit Status

This command returns the following exit values:

0 Indicates successful completion.
>0 Indicates an error occurred.

Related Information

The **X** command.

---

xmkmf Command

Purpose

Creates a **Makefile** from an **Imakefile**.

Syntax

xmkmf [**-a**] [*TopDir* [*CurDir*]]

Description

The **xmkmf** command creates a **Makefile** from an **Imakefile** shipped with third-party software. When invoked with no arguments or variables in a directory containing an **Imakefile** file, the **imake** command runs with arguments appropriate for your system (configured into **xmkmf** when X was built) and generates a **Makefile**.

Flag

-a First builds the **Makefile** in the current directory, then automatically executes **make Makefiles**, **make includes**, and **make depend**. This is how to configure software that is outside of the MIT X build tree.

Variables

Specify *TopDir* and *CurDir* if you are working inside the MIT X build tree (highly unlikely unless you are an X developer).

TopDir Specify as the relative path name from the current directory to the top of the build tree.
CurDir Specify as a relative path name from the top of the build tree to the current directory.

The *CurDir* variable is required if the current directory has subdirectories; otherwise, the **Makefile** will not be able to build the subdirectories. If a *TopDir* variable is given in its place, **xmkmf** assumes nothing is installed on your system and searches for files in the build tree instead of using the installed versions.

Related Information

The **imake** command, **make** command.

---

xmwlm Command

Purpose

Provides recording of system performance or WLM metrics.

Syntax

xmwlm [**-d** *recording_dir*] [**-n** *recording_name*] [**-t** *trace_level*] [**-L**]

Description

The **xmwlm** agent provides recording capability for a limited set of local system performance metrics. These include common CPU, memory, network, disk, and partition metrics typically displayed by the **topas** command. Daily recordings are stored in the **/etc/perf/daily** directory. The **topasout** command is used to output these recordings in raw ASCII or spreadsheet format. The **xmwlm** agent can also be used to provide recording data from Workload Management (WLM). This is the default format used when **xmwlm** is run without any flags. Daily recordings are stored in the **/etc/perf/wlm** directory. The **wlmmon** command can be used to process WLM-related recordings. The **xmwlm** agent can be started from the command line, from a user script, or can be placed near the end of the **/etc/inittab** file. All recordings cover 24-hour periods and are only retained for seven days by default. You can configure the **retain** value in the **/usr/lpp/perfagent/daily.cf** file to change the default recording duration.

Flags

-d *recording_dir* Specifies the output directory for the recording files. The default location is **/etc/perf/wlm** when **xmwlm** is run without any flags and **/etc/perf/daily** when **xmwlm** is run with the **-L** flag.

-L Specifies the collection of topas-like metrics. The metric set is not user configurable.

-n *recording_name* Specifies the name for the recording file. By default, **xmwlm** creates recording files named in an **xmwlm.***YYMMDD* format. For example, if **-n myrecording** is specified, the recording files will be named **myrecording.***YYMMDD*.

-t *trace_level* Specifies a trace level. **xmwlm** prints various information to a log file in the appropriate **/etc/perf** subdirectory. The trace level can be set from 1 to 9. More trace data is generated at higher trace levels. This trace data is useful to determine **xmwlm** recording status and for debugging purposes. The log file name is **xmwlm.log1** or **xmwlm.log2**. **xmwlm** cycles between these two files after a file reaches the maximum allowable size.

Session Recovery by xmwlm

If the **xmwlm** agent is terminated and restarted, **xmwlm** examines the recording files in the appropriate **/etc/perf** subdirectory or in the directory specified by the **-d** flag. If a recording file exists with the current date, **xmwlm** appends data to this file and continues to write to the recording file. Otherwise, a new recording file is created.

Location

/usr/bin/xmwlm

Files

/usr/bin/xmwlm	Contains the **xmwlm** agent. The agent is part of the perfagent.tools fileset.

Related Information

The **topas** command, **topasout** command, and **wlmmon** command.

---

xmodem Command

Purpose

Transfers files with the **xmodem** protocol, detecting data transmission errors during asynchronous transmission.

Syntax

xmodem { **-s** | **-r** } *FileName*

Description

The **xmodem** shell command is used with the Asynchronous Terminal Emulation (ATE) program to transfer a file, designated by the *FileName* parameter, using the **xmodem** protocol.

The **xmodem** protocol is an 8-bit transfer protocol to detect data transmission errors and retransmit the data. The workstation sending data waits until the remote system sends a signal indicating it is ready to receive data.

After the receiving system get data, it returns an acknowledgment to the sending system. In the ATE program the receiving system times out if data is not received within 90 seconds after the file transfer is initiated.

Sending and receiving with the **xmodem** command are complementary operations. One system must be set to send while the other is set to receive. Use the **xmodem** command on the remote system in combination with the **send** subcommand or the **receive** subcommand from the ATE Connected Main Menu on the local system.

To interrupt an **xmodem** file transfer, press the Ctrl-X key sequence.

Notes:

1. The DOS operating system terminates each line in an ASCII file with a newline character and a carriage return (Ctrl-M) character. UNIX terminates each line in an ASCII file only with a newline character. The carriage return characters are preserved when a DOS file is transferred to AIX. The **vi** text editor can be used to remove spurious Ctrl-M characters using the subcommand

   ```
   :%s/<Ctrl-V><Ctrl-M>//
   ```

where <Ctrl-V> and <Ctrl-M> each represent a single control character that is typed. However, since Ctrl-V is the default ATE MAINMENU_KEY, the ATE defaults must be altered in order to issue the **vi** subcommand while logged in via ATE.

2. The **xmodem** file transfer process adds Ctrl-Z characters to the last packet transferred to make the packet 128 bytes long. Most files transferred will, therefore, have Ctrl-Z characters appended to the end. The DOS operating system terminates an ASCII file with a Ctrl-Z character. Every file transferred from DOS to AIX will, therefore, end with at least one Ctrl-Z character. These extra Ctrl-Z characters can be removed with the **vi** text editor.

Flags

-r Receives data from the local workstation.
-s Sends data to the local workstation.

Examples

Sending a File with the xmodem Protocol

To send the file `myfile` with the **xmodem** protocol, use the **ate** command and the **connect** or **directory** subcommand to establish a connection to the remote system.

1. After logging in to the remote system and before pressing the MAINMENU_KEY (usually the Ctrl-Vkey sequence) to return to ATE on the local system, enter:

   ```
   xmodem -r myfile
   ```

   at the shell command line. The **xmodem** protocol starts receive mode on the remote system.
2. Press the MAINMENU_KEY to return to ATE on the local system.

   The ATE Connected Main Menu displays.
3. Enter the **send** subcommand at the prompt on the ATE Connected Main Menu:

   ```
   s myfile
   ```

   The **send** subcommand instructs the local system to send `myfile` to the remote system. After transferring the file, the ATE Connected Main Menu displays.

Receiving a File with the xmodem Protocol

Receive the file `infile` from a remote system using **xmodem** protocol with the **ate** command and the **connect** or **directory** subcommand establishing a connection to the remote system.

1. After logging in to the remote system and before pressing the MAINMENU_KEY (usually the Ctrl-V key sequence) to return to ATE on the local system, enter:

   ```
   xmodem -s infile
   ```

   at the shell command line. The **xmodem** protocol starts, in send mode, on the remote system.
2. Press the MAINMENU_KEY to return to ATE on the local system.

   The ATE Connected Main Menu displays.
3. Enter the **receive** subcommand at the prompt on the ATE Connected Main Menu:

   ```
   r infile
   ```

   The **receive** subcommand instructs the local system to receive `infile` from the remote system. After transferring the file, the ATE Connected Main Menu displays.

File

ate.def Contains ATE default values.

Related Information

The **ate** command.

The **connect** subcommand, **directory** subcommand, **modify** subcommand, **send** subcommand, **receive** subcommand.

Editing the ATE default file in *Networks and communication management* explains how to permanently change ATE defaults.

Asynchronous Terminal Emulation in *Networks and communication management* introduces the ATE program, its menus, and its control keys.

---

xmodmap Command

Purpose

Modifies keymaps in the X Server.

Syntax

xmodmap [**-display** *Display*] [**-e** *Expression*] [**-grammar** | **-help**] [**-n**] [**-pk**] [**-pke**] [**-pm**] [**-pp**] [**-quiet** | **-verbose**] [*FileName*]

Description

The **xmodmap** command edits and displays the keyboard modifier map and keymap table that client applications use to convert event keycodes into key symbols. It is usually run from the session startup script to configure the keyboard according to the personal tastes of the user.

Every time a keycode expression is evaluated, the server generates a **MappingNotify** event on every client. All of the changes should be batched together and done at one time. Clients that receive keyboard input and ignore **MappingNotify** events will not notice any changes made to keyboard mappings.

The *FileName* parameter specifies a file containing the **xmodmap** command expressions to be run. This file is usually kept in the home directory of the user with a name like **.xmodmaprc**. If no file is specified, input is taken from **stdin**.

The **xmodmap** command program reads a list of expressions and parses them all before attempting to run any of them. This makes it possible to refer to key symbols that are being naturally redefined without having to worry as much about name conflicts.

add	The key symbol names are evaluated as the line is read. This permits you to remove keys from a modifier without worrying about whether they were reassigned.
add *ModifierName* **=** *KeySymbolName...*	Adds the given key symbols to the indicated modifier map. The key symbol names are evaluated after all input expressions are read to make it easy to write expressions to swap keys.
clear *ModifierName*	Removes all entries in the modifier map for the given modifier, where the valid names are **Shift**, **Lock**, **Control**, **Mod1**, **Mod2**, **Mod3**, **Mod4**, and **Mod5** (case does not matter in modifier names, although it does matter for all other names). For example, **clear Lock** removes all keys bound to the **shift lock** modifier.
keycode *Number* **=** *KeySymbolName...*	Assigns the list of key symbols to the indicated keycode (which can be specified in decimal, hex, or octal and be determined by running the **xev** program in the **/usr/lpp/X11/Xamples/demos** directory). Usually only one key symbol is assigned to a given code.

keysym *KeySymbolName* **=** *KeySymbolName...*	The *KeySymbolName* on the left hand side is translated into matching keycodes used to perform the corresponding set of **keycode** expressions. The list of keysym names can be found in the keysym database **/usr/lib/X11/XKeysymDB** or the header file **X11/keysymdef.h** (without the *XK_* prefix). Note that if the same keysym is bound to multiple keys, the expression is run for each matching keycode.
pointer = default	Sets the pointer map back to its default settings (such as, button 1 generates a code of 1, button 2 generates a 2, and so forth).
pointer = *Button1 Button2 Button3...*	Sets the pointer map to contain the indicated button codes. The list always starts with the first physical button.
remove *ModifierName* **=** *KeySymbolName...*	Removes all keys containing the given keysyms from the indicated modifier map. Unlike **add**, the keysym names are evaluated as the line is read in. This allows for the removal of keys from a modifier without having to worry about whether or not they have been reassigned.

Lines that begin with an ! (exclamation point) are taken as comments.

If you want to change the binding of a modifier key, you must also remove it from the appropriate modifier map.

Flags

-display *Display*	Specifies the host and display to use.
-e *Expression*	Specifies an expression to be run. You can specify any number of expressions from the command line.
-grammar	Prints a help message describing the expression grammar used in files and with the **-e** *Expressions* flag prints to standard error.
-help	Prints a brief description of the command line arguments to standard error. This is done whenever an unhandled argument is given to the **xmodmap** command.
-n	Indicates that the **xmodmap** command should not change the mappings, but should display what it would do when given this flag.
-pk	Indicates that the current keymap table should print on the standard output.
-pke	Indicates that the current keymap table should be printed on the standard output in the form of expressions that can be fed back to **xmodmap**. This flag is specific to X11R5.
-pm	Indicates that the current modifier map should print on the standard output.
-pp	Indicates that the current pointer map should print on the standard output.
-quiet	Turns off the verbose logging. This is the default.
-verbose	Indicates that the **xmodmap** command should print logging information as it parses its input.

Examples

1. The following command reverses the button codes that get generated so that the primary button is pressed using the index finger of the left hand on a 3 button pointer:

   ```
   xmodmap -e "pointer = 1 2 3 4 5"
   ```

2. The following command attaches meta to the multi-language key (sometimes labeled Compose Character). It also takes advantage of the fact that applications that need a Meta key simply need to get the keycode and do not require the key symbol to be in the first column of the keymap table. This means that applications that are looking for a Multi_key (including the default modifier map) will not notice any change.

   ```
   keysym Multi_key = Multi_key Meta_L
   ```

3. To automatically generate less than and greater than characters when the comma and period keys are shifted, reset the bindings for the comma and period with the following scripts:

```
!
! make shift-, be < and shift-. be >
!
keysym comma = comma less
keysym period = period greater
```

4. To swap the location of the Control and Shift Lock keys, use the following script:

```
!
! Swap Caps_Lock and Control_L
!
remove Lock = Caps_Lock
remove Control = Control_L
keysym Control_L = Caps_Lock
keysym Caps_Lock = Control_L
add Lock = Caps_Lock
add Control = Control_L
```

Related Information

The **X** command.

---

xmpeek Command

Purpose

The **xmpeek** command allows you to query any host about the status of its **xmtopas** daemon.

Syntax

xmpeek [**-a** | **-I**] [*hostname*]

Description

The **xmpeek** command is used to list down the data consumers that currently have instruments (stat sets) defined with the **xmtopas** daemon, and list down all known data consumers by the **xmtopas** daemon . The **xmpeek** command is also used to print down all the available SPMI statistics for any given host.

Flags

-a	If this flag is specified, one line is listed for each data consumer known by the daemon. If this flag is not used, only data consumers that currently have instruments (stat sets) defined with the daemon are listed. This flag is optional.
-I	This flag is explained in the Using the xmpeek Program to Print Available Statistics document. This flag is optional.
hostname	If the hostname is specified, the daemon on the named host is asked. If no host name is specified, the daemon on the local host is asked.

Examples

The following is an example of the output from the **xmpeek** program:

```
Statistics for xmtopas daemon on *** birte ***
Instruments currently defined:    1
Instruments currently active:     1
Remote monitors currently known:  2
--Instruments--- Values  Packets
                              Internet Protocol
```

```
Defined Active  Active  Sent        Address      Port Hostname
------- ------- ------- ------- ---------------- ---- ----------
1       1       16      3,344   129.49.115.208   3885 xtra
```

Output from **xmpeek** can take two forms.

The first form is a line that informs you that the **xmtopas** daemon is not feeding any data-consumer programs. This form is used if no statsets are defined with the daemon and no command flags are supplied.

The second form includes at least as much as is shown in the preceding example, except that the single detail line for the data consumer on host **xtra** is shown only if either the **-a** flag is used or if the data consumer has at least one instrument (statset) defined with the daemon. Note that **xmpeek** itself appears as a data consumer because it uses the RSi API to contact the daemon. Therefore, the output always shows at least one known monitor.

In the fixed output, first the name of the host where the daemon is running is shown. Then follows three lines giving the totals for current status of the daemon. In the above example, you can see that only one instrument is defined and that it's active. You can also see that two data consumers are known by the daemon, but that only one of them has an instrument defined with the daemon in **birte**. Obviously, this output was produced without the **-a** flag.

An example of more activity is shown in the following sample output from **xmpeek**. The output is produced with the command:

```
xmpeek -a birte
```

Notice that some detail lines show zero instruments defined. Such lines indicate that an **are_you_there** message was received from the data consumer but that no states were ever defined or that any previously defined states were erased.

```
Statistics for smeared daemon on *** birte ***
   Instruments currently defined:    16
   Instruments currently active:     14
   Remote monitors currently known:   6
--Instruments--- Values Packets   Internet Protocol
 Defined Active  Active  Sent         Address      Port  Hostname
   8       8       35    10,232    129.49.115.203  4184   birte
   6       4       28     8,322    129.49.246.14   3211   umbra
   0       0        0         0    129.49.115.208  3861   xtra
   1       1       16     3,332    129.49.246.14   3219   umbra
   0       0        0         0    129.49.115.203  4209   birte
   1       1       16       422    129.49.115.208  3874   xtra
------- ------- ------- ----------
  16      14       95    22,308
```

Notice that the same host name may appear more than once. This is because every running copy of **xmperf** and every other active data-consumer program is counted and treated as a separate data consumer, each identified by the port number used for UDP packets as shown in the **xmpeek** output.

The second detail shows that one particular monitor on host **umbra** has six instruments defined but only four active. This would happen if a remote **xmperf** console has been opened but is now closed. When you close an **xmperf** console, it stays in the Monitor menu of the **xmperf** main window and the definition of the instruments of that console remains in the tables of the data-supplier daemon but the instruments are not active.

xmscheck Command

Purpose

The **xmscheck** command is available to pre-parse a recording configuration file and to determine how the running **xmtopas** daemon is configured for recording.

Syntax

xmscheck [*file_name*]

Description

When the **xmtopas** command is started with the command line argument **-v**, its recording configuration file parser writes the result of the parsing to the log file. The output includes a copy of all lines in the recording configuration file, any error messages, and a map of the time scale with indication of when recording starts and stops.

Although this is useful to document what is read from the recording configuration file, it is not a useful tool for debugging of a new or modified recording configuration file. Therefore, the program **xmscheck** command is available to preparse a recording configuration file before you move it to the **/etc/perf** directory, where the **xmtopas** command looks for the recording configuration file.

When **xmscheck** command is started without any command line argument, it parses the file **/etc/perf/xmservd.cf**. This way, you can determine how the running daemon is configured for recording. If a file name is specified on the command line, that file is parsed.

Output from the **xmscheck** command goes to stdout. The parsing is done by the exact same module that does the parsing in the **xmtopas** command. That module is linked in as part of both programs. The parsing checks that all statistics specified are valid and prints the time scale for starting and stopping recording in the form of a “time table.”

In the time table, each minute has a numeric code. The meaning of codes is as follows:

0 Recording is inactive. Neither a start nor a stop request was given for the minute.
1 Recording is active. Neither a start nor a stop request was given for the minute.
2 Recording is inactive. A stop request was given for the minute.
3 Recording is active. A start request was given for the minute.

Examples

The following example shows how the **xmscheck** command formats the time table. The table only displays part of Tuesday. The example shown in Hot Lines Recording Only section was used to produce this output.

```
Day 2, Hour 00: 000000000000000000000000000000000000000000000000000000000000
Day 2, Hour 01: 000000000000000000000000000000000000000000000000000000000000
Day 2, Hour 02: 000000000000000000000000000000000000000000000000000000000000
Day 2, Hour 03: 000000000000000000000000000000000000000000000000000000000000
Day 2, Hour 04: 000000000000000000000000000000000000000000000000000000000000
Day 2, Hour 05: 000000000000000000000000000000000000000000000000000000000000
Day 2, Hour 06: 000000000000000000000000000000000000000000000000000000000000
Day 2, Hour 07: 000000000000000000000000000000000000000000000000000000000000
Day 2, Hour 08: 000000000000000000000000000000311111111111111111111111111111
Day 2, Hour 09: 111111111111111111111111111111111111111111111111111111111111
Day 2, Hour 10: 111111111111111111111111111111111111111111111111111111111111
Day 2, Hour 11: 111111111111111111111111111111111111111111111111111111111111
Day 2, Hour 12: 200000000000000000000000000000000000000000000000000000000000
Day 2, Hour 13: 311111111111111111111111111111111111111111111111111111111111
Day 2, Hour 14: 111111111111111111111111111111111111111111111111111111111111
```

```
Day 2, Hour 15: 111111111111111111111111111111111111111111111111111111111111
Day 2, Hour 16: 111111111111111111111111111111111111111111111111111111111111
Day 2, Hour 17: 200000000000000000000000000000000000000000000000000000000000
Day 2, Hour 18: 000000000000000000000000000000000000000000000000000000000000
Day 2, Hour 19: 000000000000000000000000000000000000000000000000000000000000
Day 2, Hour 20: 000000000000000000000000000000000000000000000000000000000000
Day 2, Hour 21: 000000000000000000000000000000000000000000000000000000000000
Day 2, Hour 22: 000000000000000000000000000000000000000000000000000000000000
Day 2, Hour 23: 000000000000000000000000000000000000000000000000000000000000
```

xmtopas Command

Purpose

The **xmtopas** daemon acts as a recording facility and is controlled through the xmtopas.cf configuration file. This daemon simultaneously provides near real-time network-based data monitoring and local recording on a given node.

Syntax

xmtopas [**-v**] [**-b** *UDP_buffer_size*] [**-i** *min_remote_interval*] [**-l** *remove_consumer_timeout*] [**-m** *supplier_timeout*] [**-p** *trace_level*] [**-s** *max_logfile_size*] [**-t** *keep_alive_limit*] [**-x** *xmtopas_execution_priority*]

Description

The **xmtopas** command is always started from **inetd** daemon. Therefore, command line options must be specified on the line defining the **xmtopas** command to the **inetd** daemon in the /etc/inetd.conf file.

Flags

-v
Causes parsing information for the **xmtopas** recording configuration file to be written to the **xmtopas** log file.

-b *UDP_buffer_size*
Defines the size of the buffer used by the daemon to send and receive UDP packets. The buffer size must be specified in bytes and can be from 4,096 to 16,384 bytes. The buffer size determines the maximum number of data values that can be sent in one **data_feed** packet. The default buffer size is 4096 bytes, which allows for up to 124 data values in one packet.

-i *min_remote_interval*
Defines the minimum interval in milliseconds with which data feeds can be sent. Default is 500 milliseconds. A value between 100 and 5,000 milliseconds can be specified. Any value specified is rounded to a multiple of 100 milliseconds. Whichever minimum remote interval is specified causes all requests for data feeds to be rounded to a multiple of this value. For more information review the Rounding of Sampling Interval topic.

-l *remove_consumer_timeout*
Sets the **time_to_live** after feeding of statistics data has ceased as described in section Life and Death of xmtopas. Must be followed by a number of minutes. A value of 0 (zero) minutes causes the daemon to stay alive forever. The default time_to_live is 15 minutes.

This value is also used to control when to remove inactive data-consumers as described in Removing Inactive Data Consumers topic.

-m *supplier_timeout*
When a dynamic data-supplier is active, this value sets the number of seconds of inactivity from the DDS before the SPMI assumes the DDS is dead. When the timeout value is exceeded, the **SiShGoAway** flag is set in the shared memory area and the SPMI disconnects from the area. If this flag is not given, the timeout period is set to 90 seconds.

The size of the timeout period is kept in the SPMI common shared memory area. The value stored is the maximum value requested by any data consumer program, including the **xmtopas** command.

-p *trace_level*
Sets the trace level, which determines the types of events written to the **/etc/perf/xmtopas.log1** log file or the **/etc/perf/xmtopas.log2** log file. This flag must be followed by a digit from 0 to 9, with 9 being the most detailed trace level. Default trace level is 0 (zero), which disables tracing and logging of events but logs error messages.

-s *max_logfile_size*
Specifies the approximate maximum size of the log files. At least every **time_to_live** minutes, it is checked if the currently active log file is bigger than the *max_logfile_size* value. If so, the current log file is closed and logging continues to the alternate log file, which is first reset to zero length. The two log files are **/etc/perf/xmtopas.log1** and **/etc/perf/xmtopas.log2**. Default maximum file size is 100,000 bytes. You cannot make the *max_logfile_size* value smaller than 5,000 or larger than 10,000,000 bytes.

-t *keep_alive_limit*
Sets the *keep_alive_limit* value must be followed by a number of seconds from 60 to 900 (1 to 15 minutes). The default is 300 seconds (5 minutes).

-x *xmtopas_execution_priority*
Sets the execution priority of the **xmtopas** command. Use this option if the default execution priority of the **xmtopas** command is unsuitable in your environment. Generally, the daemon should be given as high execution priority as possible (a smaller number gives a higher execution priority).

On systems other than IBMRS/6000systems, the *-x* flag is used to set the nice priority of the **xmtopas** command. The nice priority is a value from -20 to 19. The default is -20.

xntpd Daemon

Purpose

Starts the Network Time Protocol (NTP) daemon.

Syntax

xntpd [**-a**] [**-b**] [**-d**] [**-D** *Level*] [**-m**] [**-x**][**-c** *ConfigFile*] [**-e** *AuthenticationDelay*] [**-f** *DriftFile*] [**-k** *KeyFile*] [**-l** *LogFile*] [**- o** *TraceFile*] [**-p** *pidFile*] [**-r** *BroadcastDelay*] [**-s** *StatsDirectory*] [**-t** *TrustedKey*] [**-v** *SystemVariable*] [**-V** *SystemVariable*]

Description

The **xntpd** daemon sets and maintains a Unix system time-of-day in compliance with Internet standard time servers. The **xntpd** daemon is a complete implementation of the Network Time Protocol (NTP) version 3 standard, as defined by RFC 1305, and also retains compatibility with version 1 and 2 servers as defined by RFC 1059 and RFC 1119, respectively. The **xntpd** daemon does all computations in fixed point arithmetic and does not require floating point code.

The **xntpd** daemon reads from a configuration file (**/etc/ntp.conf** is the default) at startup time. You can override the configuration file name from the command line. You can also specify a working, although limited, configuration entirely on the command line, eliminating the need for a configuration file. Use this method when configuring the **xntpd** daemon as a broadcast or multicast client, that determines all peers by listening to broadcasts at runtime. You can display the **xntpd** daemon internal variables with the **ntpq** command (Network Time Protocol (NTP) query program). You can alter configuration options with the **xntpdc** command.

The **xntpd** daemon operates in several modes, including symmetric active/passive, client/server and broadcast/multicast. A broadcast/multicast client can automatically discover remote servers, compute one-way delay correction factors and configure itself automatically. This mode makes it possible to deploy a group of workstations without specifying a configuration file or configuration details specific to its environment.

> **Note:** When operating in a client mode running AIX 4.2.1 or later, the **xntpd** daemon will exit with an error if no configured servers are within 1000 seconds of local system time. Use the **date** or **ntpdate** command to set the time of a bad skewed system before starting **xntpd**.

Flags

-a	Runs in authenticate mode
-b	Listens for broadcast NTP and synchronizes to them if available.
-c *ConfigFile*	Specifies the name of an alternate configuration file.
-d	Specifies debugging mode. This flag may occur multiple times (maximum of 10), with each occurrence indicating greater detail of display.
-D *Level*	Specifies debugging level directly (value from 1 to 10).
-e *AuthenticationDelay*	Specifies the time, in seconds, it takes to compute the NTP encryption field on this computer.
-f *DriftFile*	Specifies the location of the drift file.
-k *KeyFile*	Specifies the location of the file which contains the NTP authentication keys.
-l *LogFile*	(lowercase L) Specifies the use of a log file instead of logging to syslog.
-m	Listens for multicast messages and synchronizes to them if available. Assumes multicast address 224.0.1.1.
-o *TraceFile*	Specifies trace file name (default is **stderr**).
-p *pidFile*	Specifies the name of the file to record the daemon's process id. There is no default.
-r *BroadcastDelay*	Specifies the default delay (in seconds) if the calibration procedure fails. Normally, the **xntpd** daemon automatically compensates for the network delay between the broadcast/multicast server and the client.
-s *StatsDirectory*	Specifies the directory to use for creating statistics files.
-t *TrustedKey*	Adds the specified key number to the trusted key list.
-v *SystemVariable*	Adds the specified system variable
-V *SystemVariable*	Adds the specified system variable listed by default.
-x	Makes small time adjustments. (SLEWING)

Reference Clock Support

For the purposes of configuration, the **xntpd** daemon treats reference clocks in a manner analogous to normal NTP peers as much as possible. It refers to reference clocks by address, same as a normal peer is, though it uses an invalid IP address to distinguish them from normal peers. AIX 4.2 supports one type of reference clock, based on the system clock (type 1).

Reference clock addresses are of the form 127.127.*Type*.*Unit* where *Type* is an integer denoting the clock type and *Unit* indicates the type-specific unit number. You configure reference clocks by using a server statement in the configuration file where the *HostAddress* is the clock address. The key, version and ttl options are not used for reference clock support.

Reference clock support provides the **fudge** command, which configures reference clocks in special ways. This command has the following format:

fudge 127.127.*Type***.***Unit* [**time1** *Seconds*] [**time2** *Seconds*] [**stratum** *Integer*] [**refid** *Integer*] [**flag1 0** | **1**] [**flag2 0** | **1**] [**flag3 0** | **1**] [**flag4 0** | **1**]

The **time1** and **time2** options are in fixed point seconds and used in some clock drivers as calibration constants.

The **stratum** option is a number in the range zero to 15 and used to assign a nonstandard operating stratum to the clock. Since the **xntpd** daemon adds one to the stratum of each peer, a primary server ordinarily displays stratum one. In order to provide engineered backups, use the **stratum** option to specify the reference clock stratum as greater than zero. Except where noted, this option applies to all clock drivers.

The **refid** option is an ASCII string in the range one to four characters and used to assign a nonstandard reference identifier to the clock.

The binary flags: **flag1**, **flag2**, **flag3** and **flag4** are for customizing the clock driver. The interpretation of these values, and whether they are used at all, is a function of the needs of the particular clock driver.

Exit Status

This command returns the following exit values:

0 Successful completion.
>0 An error occurred.

Security

Access Control: You must have root authority to run this command.

Auditing Events: N/A

Examples

1. To start the **xntpd** daemon, enter:
   ```
   startsrc -s xntpd
   ```
2. To stop the **xntpd** daemon, enter:
   ```
   stopsrc -s xntpd
   ```
3. To use the authentication key file `/etc/ntp.new.keys` when running the **xntpd** daemon, enter:
   ```
   /usr/sbin/xntpd -k /etc/ntp.new.keys
   ```

Files

/usr/sbin/xntpd	Contains the **xntpd** daemon.
/etc/ntp.conf	Contains the default configuration file.
/etc/ntp.drift	Contains the default drift file.
/etc/ntp.keys	Contains the default key file.

Related Information

The **ntpq**, **ntpdate**, **ntptrace**, and **xntpdc** commands.

---

xntpdc Command

Purpose

Starts the query/control program for the Network Time Protocol daemon, **xntpd**.

Syntax

xntpdc [**-i**] [**-l**] [**-n**] [**-p**] [**-s**] [**-c** *SubCommand*] [*Host* ...]

Description

The **xntpdc** command queries the **xntpd** daemon about its current state and requests changes to that state. It runs either in interactive mode or by using command-line arguments. The **xntpdc** command interface displays extensive state and statistics information. Nearly all the configuration options that can be specified at start-up using the **xntpd** daemon's configuration file, can also be specified at run-time using the **xntpdc** command.

If you enter the **xntpdc** command with one or more request flags, the NTP servers running on each of the hosts specified (or defaults to local host) receive each request. If you do not enter any request flags, the **xntpdc** command tries to read commands from standard input and run them on the NTP server running on the first host specified or on the local host by default. It prompts for subcommands if standard input is the terminal.

The **xntpdc** command uses NTP mode 7 packets to communicate with the NTP server and can query any compatible server on the network that permits it.

The **xntpdc** command makes no attempt to retransmit requests, and will time-out requests if the remote host does not respond within a suitable time.

Specifying a flag other than **-i** or **-n** sends the queries to the specified hosts immediately. Otherwise, the **xntpdc** command attempts to read interactive format commands from standard input.

Flags

-c *SubCommand*	Specifies an interactive format command. This flag adds *SubCommand* to the list of commands to run on the specified hosts. You can enter multiple **-c** flags.
-i	Specifies interactive mode. Standard output displays prompt and standard input reads commands.
-l	(lowercase L) Displays a list of the peers known to the servers. This is the same as the **listpeers** subcommand.
-n	Displays all host addresses in dotted decimal format (0.0.0.0) rather than the canonical host names.
-p	Displays a list of the peers known to the server and a summary of their state. This is the same as the **peers** subcommand.
-s	Displays a list of the peers known to the server and a summary of their state but in a format different from the **-p** flag. This is the same as the **dmpeers** subcommand.

Parameters

Host ...	Specifies the hosts.

xntpdc Internal Subcommands

You can run a number of interactive format subcommands entirely within the **xntpdc** command that do not send NTP mode 7 requests to a server. The following subcommands can only be used while running the **xntpdc** query program.

Interactive Format Subcommands

Interactive format subcommands consist of a keyword followed by zero to four arguments. You only need to type enough characters of the full keyword to uniquely identify the subcommand. The output of a subcommand goes to standard output, but you can redirect the output of individual subcommands to a file by appending a greater-than sign (**>**), followed by a file name, to the command line.

? [*SubCommand*]	Displays command usage information. When used without *SubCommand*, displays a list of all the **xntpdc** command keywords. When used with *SubCommand*, displays function and usage information about the command.
help [*SubCommand*]	Same as the **?** [*Subcommand*] subcommand.
delay *Milliseconds*	Specifies the time interval to add to timestamps included in requests that require authentication. This subcommand enables unreliable server reconfiguration over long delay network paths or between machines whose clocks are unsynchronized. If you enter this subcommand without an argument, it prints the current setting for this subcommand.
host *HostName*	Specifies the host to send queries to. *HostName* may be either a host name or a numeric address. If you enter this subcommand without an argument, it prints the current setting for this subcommand.
hostnames yes | **no**	Specifies whether to display the host name (**yes**) or the numeric address (**no**). The default is **yes** unless the **-n** flag is used. If you enter this subcommand without an argument, it prints the current setting for this subcommand.
keyid *Number*	Specifies the server key number to use to authenticate configuration requests. If you enter this subcommand without an argument, it prints the current setting for this subcommand.
passwd	Prompts you to type in the NTP server authentication password to use to authenticate configuration requests.
quit	Exits the **xntpdc** query program.
timeout *Milliseconds*	Specifies the time-out period for responses to server queries. The default is 8000 milliseconds. If you enter this subcommand without an argument, it prints the current setting for this subcommand.

Query Subcommands

The **xntpdc** query subcommands result in sending NTP mode 7 packets containing requests to the server. These subcommands are read-only (they do not modify the server configuration state).

clkbug *ClockPeerAddress* [*Addr2*] [*Addr3*] [*Addr4*]	Displays debugging information for a reference clock driver. Some clock drivers provide this information that is mostly undecodable without a copy of the driver source in hand.
clockbug *ClockPeerAddress* [*Addr2*] [*Addr3*] [*Addr4*]	Displays information concerning a peer clock. The values obtained provide information on the setting of fudge factors and other clock performance information.

dmpeers
Displays a list of peers for which the server is maintaining state, along with a summary of that state. Identical to the output of the **peers** subcommand except for the character in the leftmost column. Characters only are displayed beside peers that were included in the final stage of the clock selection algorithm.

The possible character in the leftmost column are:

. Indicates that this peer was cast off in the falseticker detection.

+ Indicates that the peer made it through.

***** Denotes the peer the server is currently synchronizing with.

iostats
Displays statistics counters maintained in the input-output module.

kerninfo
Displays kernel phase-lock loop operating parameters. This information is available only if the kernel of the hosts being generated has been specially modified for a precision timekeeping function.

listpeers
Displays a brief list of the peers for which the server is maintaining state. These include all configured peer associations as well as those peers whose stratum is such that the server considers them to be possible future synchronization candidates.

loopinfo [**oneline** | **multiline**]
Displays the values of selected loop filter variables. The loop filter is the part of NTP that adjusts the local system clock. The `offset` is the last offset given to the loop filter by the packet processing code. The `frequency` is the frequency error of the local clock in parts-per-million (ppm). The `poll adjust` controls the stiffness (resistance to change) of the phase-lock loop and the speed at which it can adapt to oscillator drift. The `watchdog timer` is the number of elapsed seconds since the last sample offset given to the loop filter. The **oneline** and **multiline** options specify the format to display this information. The **multiline** option is the default.

memstats
Displays statistics counters related to memory allocation code.

monlist
Displays traffic counts collected and maintained by the monitor facility.

peers

Displays a list of peers for which the server is maintaining state, along with a summary of that state. Summary information includes:

- address of the remote peer,
- reference ID (0.0.0.0 for an unknown reference ID),
- the stratum of the remote peer (a stratum of 16 indicates the remote peer is unsynchronized),
- the polling interval (seconds),
- the reachability register (octal), and
- the current estimated delay, offset and dispersion of the peer (seconds).

The character in the left margin indicates the mode this peer entry is in:

+ symmetric active.

- symmetric passive.

= remote server polled in client mode.

^ server is broadcasting to this address.

~ remote peer is sending broadcasts.

***** marks the peer the server is currently synchronizing to.

The contents of the host field may be a host name, an IP address, a reference clock implementation name with its parameter or REFCLK (*ImplementationNumber*, *Parameter*). Only IP addresses display when using **hostnames no**.

pstats *PeerAddress* [*Addr2*] [*Addr3*] [*Addr4*]

Displays per-peer statistic counters associated with the specified peers.

reslist

Displays the server's restriction list which may help to understand how the restrictions are applied.

sysinfo

Displays a variety of system state variables related to the local server. All except the last four lines are described in the NTP Version 3 specification, RFC 1305. The system flags show various system flags, some of which can be set and cleared by the **enable** and **disable** configuration statements. The `stability` is the residual frequency error remaining after applying the system frequency correction. You use it for maintenance and debugging. In most architectures, this value will initially decrease from as high as 500 ppm to a nominal value in the range .01 to 0.1 ppm. If it remains high for some time after starting the daemon, something may be wrong with the local clock, or the value of the kernel variable *Tick* may be incorrect. The `broadcastdelay` shows the default broadcast delay, as set by the **broadcastdelay** configuration statement, while the `authdelay` shows the default authentication delay, as set by the **authdelay** configuration statement.

sysstats

Displays statistics counters maintained in the protocol module.

timerstats

Displays statistics counters maintained in the timer/event queue support code.

Runtime Configuration Requests Subcommands

The server authenticates all requests that cause state changes in the server by using a configured NTP key. The server can also disable this facility by not configuring a key. You must make the key number and the corresponding key known to the **xtnpdc** command. You can do this by using the **keyid** and **passwd** subcommands, which prompts at the terminal for a password to use as the encryption key. The **xtnpdc** command will also prompt you automatically for both the key number and password the first time you give a subcommand that would result in an authenticated request to the server. Authentication not only verifies that the requester has permission to make such changes, but also protects against transmission errors.

Authenticated requests always include a timestamp in the packet data, as does the computation of the authentication code. The server compares this timestamp to the time at which it receives the packet.

The server rejects the request if they differ by more than 10 seconds. This makes simple replay attacks on the server, by someone able to overhear traffic on your LAN, much more difficult. It also makes it more difficult to request configuration changes to your server from topologically remote hosts. While the reconfiguration facility works well with a server on the local host, and may work adequately between time-synchronized hosts on the same LAN, it works very poorly for more distant hosts. So, if you choose reasonable passwords, take care in the distribution and protection of keys and apply appropriate source address restrictions, the run-time reconfiguration facility should provide an adequate level of security.

The following subcommands all make authenticated requests.

addpeer *PeerAddress* [*Keyid*] [*Version*] [**prefer**]	Adds a configured peer association operating in symmetric active mode at the specified address. You may delete an existing association with the same peer or simply convert an existing association to conform to the new configuration when using this subcommand. If the *Keyid* is a nonzero integer, all outgoing packets to the remote server will have an authentication field attached encrypted with this key. To specify no authentication , enter *Keyid* as 0 or leave blank. The values for *Version* can be 1, 2 or 3, with 3 as the default. The **prefer** option indicates a preferred peer used primarily for clock synchronization if possible. The preferred peer also determines the validity of the PPS signal. If the preferred peer is suitable for synchronization, so is the PPS signal.
addserver *PeerAddress* [*Keyid*] [*Version*] [**prefer**]	Same as the **addpeer** subcommand, except that the operating mode is client.
addtrap *Address* [*Port*] [*Interface*]	Sets a trap for asynchronous messages at the specified address and port number for sending messages with the specified local interface address. If you do not specify the port number, the value defaults to 18447. If you do not specify the interface address, the value defaults to the source address of the local interface.
authinfo	Displays information concerning the authentication module, including known keys and counts of encryptions and decryptions performed.
broadcast *PeerAddress* [*Keyid*] [*Version*]	Same as the **addpeer** subcommand, except that the operating mode is broadcast. The *PeerAddress* can be the broadcast address of the local network or a multicast group address assigned to NTP (224.0.1.1).
clrtrap *Address* [*Port*] [*Interface*]	Clears a trap for asynchronous messages at the specified address and port number for sending messages with the specified local interface address. If you do not specify the port number, the value defaults to 18447. If you do not specify the interface address, the value defaults to the source address of the local interface.
delrestrict *Address Mask* [**ntpport**]	Deletes the matching entry from the restrict list.
disable *Option* ...	Disables various server options. Does not affect options not mentioned. The **enable** subcommand describes the options.

enable *Option* ...

Enables various server options. Does not affect options not mentioned. You can specify one or more of the following values for *Option*:

auth Causes the server to synchronize with unconfigured peers only if the peer has been correctly authenticated using a trusted key and key identifier. The default for this option is disable (off).

bclient Causes the server to listen for a message from a broadcast or multicast server, following which an association is automatically instantiated for that server. The default for this argument is disable (off).

monitor
Enables the monitoring facility, with default enable (on).

pll Enables the server to adjust its local clock, with default enable (on). If not set, the local clock free-runs at its intrinsic time and frequency offset. This option is useful when the local clock is controlled by some other device or protocol and NTP is used only to provide synchronization to other clients.

stats Enables statistics facility filegen, with default enable (on).

fudge *PeerAddress* [*Time1*] [*Time2*] [*Stratum*] [*Refid*]

Provides a way to set certain data for a reference clock.

Time1 and *Time2* are in fixed point seconds and used in some clock drivers as calibration constants.

Stratum is a number in the range zero to 15 and used to assign a nonstandard operating stratum to the clock.

Refid is an ASCII string in the range one to four characters and used to assign a nonstandard reference identifier to the clock.

monitor yes | **no**

Enables or disables the monitoring facility. A **monitor no** subcommand followed by a **monitor yes** subcommand is a good way of resetting the packet counts.

readkeys

Purges the current set of authentication keys and obtains a new set by rereading the keys file specified in the **xntpd** configuration file. This allows you to change encryption keys without restarting the server.

reset *Module*

Clears the statistics counters in various modules of the server. You can specify one or more of the following values for *Module*: **io**, **sys**, **mem**, **timer**, **auth**, **allpeers**.

restrict *Address Mask Option* ...

Adds the values of *Option* to an existing restrict list entry, or adds a new entry to the list with the specified *Option*. The **mask** option defaults to 255.255.255.255, meaning that *Address* is treated as the address of an individual host. You can specify one or more of the following values for *Option*:

ignore Ignore all packets from hosts that match this entry. Does not respond to queries nor time server polls.

limited Specifies that these hosts are subject to client limitation from the same net. Net in this context refers to the IP notion of net (class A, class B, class C, etc.). Only accepts the first **client_limit** hosts that have shown up at the server and that have been active during the last **client_limit_period** seconds. Rejects requests from other clients from the same net. Only takes into account time request packets. Private, control, and broadcast packets are not subject to client limitation and therefore do not contribute to client count. The monitoring capability of the **xntpd** daemon keeps a history of clients. When you use this option, monitoring remains active. The default value for **client_limit** is 3. The default value for **client_limit_period** is 3600 seconds.

lowpriotrap
Declare traps set by matching hosts to low priority status. The server can maintain a limited number of traps (the current limit is 3), assigned on a first come, first served basis, and denies service to later trap requestors. This parameter modifies the assignment algorithm by allowing later requests for normal priority traps to override low priority traps.

nomodify
Ignore all NTP mode 6 and 7 packets that attempt to modify the state of the server (run time reconfiguration). Permits queries that return information.

nopeer Provide stateless time service to polling hosts, but not to allocate peer memory resources to these hosts.

noquery
SIgnore all NTP mode 6 and 7 packets (information queries and configuration requests) from the source. Does not affect time service.

noserve
Ignore NTP packets whose mode is not 6 or 7. This denies time service, but permits queries.

notrap Decline to provide mode 6 control message trap service to matching hosts. The trap service is a subsystem of the mode 6 control message protocol intended for use by remote event logging programs.

notrust
STreat these hosts normally in other respects, but never use them as synchronization sources.

ntpport
Match the restriction entry only if the source port in the packet is the standard NTP UDP port (123).

setprecision *Precision*
Sets the precision that the server advertises. *Precision* should be a negative integer in the range -4 through -20.

traps
Displays the traps set in the server.

trustkey *Keyid* ...
Adds one or more keys to the trusted key list. When you enable authentication, authenticates peers with trusted time using a trusted key.

unconfig *PeerAddress* [*Addr2*] [*Addr3*] [*Addr4*]
Removes the configured bit from the specified peers. In many cases deletes the peer association. When appropriate, however, the association may persist in an unconfigured mode if the remote peer is willing to continue on in this fashion.

unrestrict *Address Mask Option* ...
Removes the specified options from the restrict list entry indicated by *Address* and *Mask*. The **restrict** subcommand describes the values for *Option*.

untrustkey *Keyid* ...
Removes one or more keys from the trusted key list.

Exit Status

This command returns the following exit values:

0 Successful completion.
>0 An error occurred.

Security

Access Control: You must be part of the system group to run this command.

RBAC Disclaimer

Auditing Events: N/A

Displays per-peer statistic counters associated with the specified peers.

Examples

1. To start the query/control program for the Network Time Protocol daemon, enter:
   ```
   xntpdc
   ```
2. To display the statistic counters of the peer at address 127.127.1.0 on host 9.3.149.107, enter:
   ```
   xntpdc -c "pstats 127.127.1.0" 9.3.149.107
   ```

   Output similar to the following is displayed:
   ```
   remote host:   LOCAL(0)
   local interface:   127.0.0.1
   time last received:   49s
   time until next send:   15s
   reachability change:   818s
   packets sent:   13
   packets received:   13
   bad authentication:   0
   bogus origin:   0
   duplicate:   0
   bad dispersion:   4
   bad reference time:   0
   candidate order:   1
   ```

Files

/usr/sbin/xntpdc Contains the **xntpdc** command.

Related Information

The **ntpq**, **ntpdate**, and **ntptrace** commands.

The **xntpd** daemon.

RBAC in *Security*.

Trusted AIX in *Security*.

xpr Command

Purpose

Formats a window dump file for output to a printer.

Syntax

xpr [**-append** *FileName* [**-noff**] | **-output** *FileName*] [**-landscape** | **-portrait**] [**-compact**] [**-cutoff** *Level*] [**-density** *Dpi*] [**-gray** { **2** | **3** | **4** }] [**-header** *String*] [**-height** *Inches*] [**-left** *Inches*] [**-noposition**] [**-plane** *PlaneNumber*] [**-psfig**] [**-report**] [**-rv**] [**-scale** *Scale*] [**-split** *Number*] [**-top** *Inches*] [**-trailer** *String*] [**-width** *Inches*] [**-device** *Device*] [*ImageFile*]

Description

The **xpr** command uses a window dump file produced by the **xwd** utility as input and formats the dump file for output on all printers supported by the hardware. If you do not specify a file argument, the **xpr** command uses standard input. By default, the **xpr** command prints the largest possible representation of the window on the output page.

The **xpr** command options allow you to add headers and trailers, specify margins, adjust the scale and orientation, and append multiple window dumps to a single output file. Output is to standard output unless the **-output** flag is specified.

Flags

-append *FileName*
: Specifies a file name previously produced by the **xpr** command to which the window is to append. (This flag is not supported on PostScript printers.)

-compact
: Uses simple run-length encoding for compact representation of windows with many white pixels. This flag compresses white space but not black space, so it is not useful for reverse-video windows.

  (This flag supports PostScript, LIPS II+, and LIPSIII output only.)

-cutoff *Level*
: Changes the intensity level where colors are mapped to black or white for monochrome output on a LaserJet printer. The *Level* variable is expressed as a percentage of full brightness. Fractions are acceptable.

-device *Device*
: Specifies the device on which the file prints. The **xpr** command supports the following printers:

  3812 or pp
  : IBM PP3812

  4207
  : Proprinter

  5201
  : IBM Quietwriter 1 model 2

  5202
  : IBM Quietwriter 2

  jprinter
  : IBM Japanese Printer (Japanese data stream)

  ljet
  : HP LaserJet and IBM Laser Printer

  ps
  : PostScript printers (this is the default)

  lips2
  : Canon LaserShot LIPS II+ mode

  lips3
  : Canon LaserShot LIPS III mode

-density *Dpi*
: Indicates the dots-per-inch (dpi) density that the HP printer uses. 300 dpi is the default. Allowable densities are 300, 150, 100, and 75 dpi.

-gray *Number*
Specifies gray-scale conversion to a color image, rather than mapping to a black-and-white image. The *Number* variable must be one of the following:

2 2 x 2 conversion

3 3 x 3 conversion

4 4 x 4 conversion

This conversion doubles, triples, or quadruples, respectively, the effective width and height of the image.

Note: This option is valid only for PostScript printers.

-header *String*
Specifies a header string to print above the window.

-height *Inches*
Specifies the maximum height of the page.

ImageFile
Contains the captured bitmap of the image. If you do not specify the *ImageFile* parameter, the **xpr** command uses standard input.

-landscape
Forces the window to print in landscape mode. (The display is laid out with the windows being wider than they are high.) By default, a window prints so that its longest side follows the long side of the paper.

-left *Inches*
Specifies the left margin in inches. Fractions are acceptable. By default, this flag prints the window on the center of the page.

-noff
When specified in conjunction with the **-append** flag, the window is displayed on the same page as the previous window. (This flag is not supported on PostScript printers.)

-noposition
Causes the header, trailer, and image positioning command generation to be bypassed for the LaserJet printer.

-output *FileName*
Specifies an output file name. If you do not specify this option, the **xpr** command uses standard output.

-plane*PlaneNumber*
Specifies which bit plane to use in an image. The default uses the entire image and maps values into black and white based on color intensities. This option is not supported for the LaserJet printer.

-portrait
Forces the window to print in portrait mode. (The display is laid out with the windows being higher than they are wide.) By default, a window prints so that its longest side follows the long side of the paper.

-psfig
Suppresses translation of the PostScript picture to the center of the page.

-report
Prints out statistics to standard error about the window *ImageFile* parameter.

-rv
Forces the window to print in reverse video.

-scale *Scale*
Affects the size of the window on the page. PostScript printers are able to translate each bit in a window pixel map into a grid of a specified size. For example, each bit might translate into a 3 x 3 grid. To specify a 3 x 3 grid, enter **-scale 3**. By default, a window prints with the largest scale that fits on the page for the specified orientation. If you do not specify a device, the aspect ratio can vary.

-split *Number*
Splits a window into several pages. This might be necessary for very large windows that would otherwise cause the printer to overload and print the page in an obscure manner. (This flag is not supported on PostScript or HP Laserjet printers.)

-top *Inches*
Specifies the top margin for the window in inches. Fractions are acceptable. By default, this flag prints the window on the center of the page.

-trailer *String*
Specifies a trailer string to print below the window.

-width *Inches*
Specifies the maximum width of the page.

Note: The 4207, 5201, and 5202 printers' images must be recorded by the **xwd** utility in XYPixmap or XYBitmap format. XYPixmap images are converted into bitmaps using a thresholding algorithm. For the HP Laserjet printer, multiplane images must be recorded in ZPixmap format. Single plane images may be either XYPixmap, XYBitmap, or ZPixmap formats.

Related Information

The **X** command, **xwd** command, **xwud** command.

xpreview Command

Purpose

Displays troff files on an X display.

Syntax

xpreview [**-BackingStore** *BackingStoreType*] [**-page** *Number*] [*ToolKitFlag ...*] { *File* | **-** }

Description

The **xpreview** command is an AIXwindows 2.1- and Motif2.1-based application that displays output from the **troff** command on an AIXwindows display. The **troff** command output file must be prepared for the devX100 device.

The user interface contains the standard AIXwindows interface controls for calling the root menu, iconifying the window, and setting the window to full screen size. The interface also includes a main window with a scrollable display area for text. Use the pushbuttons for Next, Previous, Goto Page, Print Page, Print File, and Newfile to manipulate the viewing document.

Mouse button three actuates a popup menu for configuring print capabilities. The menu includes an option to set the command line and another to select a printer queue. The command line dialog box expects command line input through the **troff** command. For example,

```
pic -Tibm3816 troff-input-file |tbl|troff -mm -Tibm3816
```

is an acceptable command line. The printer queue option displays a list of configured printer queues. If this option is not selected, the **xpreview** command uses the system-defined default queue.

When you are previewing an input file, the Print Page and Print File buttons require command line input. Note that once a printer queue is selected, it remains selected for the duration of the viewing session, or until an alternate printer queue is selected.

Fonts supported for the devX100 device in European locales are:

- Times New Roman in normal, italic, and bold
- Courier in normal and bold
- Helvetica in normal and bold
- Symbol

The **xpreview** command supports the following font sizes: 8, 10, 14, 18, 24, 30, and 36.

The **xpreview** command does not display files resulting from the **troff** command constructed for a device other than those described in this document.

To preview a file on a certain device, the **xpreview** command requires the fonts found in the following directories:

- **/usr/lib/X11/fonts** directory for files formatted for font files other than Japanese
- **/usr/lib/X11/fonts/JP** for Japanese font files

Multibyte Support

The **xpreview** command supports multibyte locales. Also, to display Japanese characters, Japanese 16-dot fonts (part of the Japanese BSL package) and 24- and 32-dot fonts (part of the AIXwindows font package) must be installed. To display Korean characters, Korean fonts (part of the Korean BSL package) must be installed.

Japanese support currently includes the following font sets:

- In 16-dot: RomanKn12, Kanji12, and IBM_JPN12
- In 24-dot: RomanKn17, Kanji17, and IBM_JPN17
- In 32-dot: RomanKn23, Kanji23, and IBM_JPN23, or RomanKn23G, Kanji23G, and IBM_JPN23G

Korean support currently includes the following font sets:

- In 16-dot, EnglHg16 and Hangul16
- In 24-dot, EnglHg24 and Hangul24

Flags

The **xpreview** command accepts the standard **X** Toolkit command line flags, as well as the following flags:

-
Requires input to be read from standard input.

-help
Indicates that a brief summary of the allowed command line flags should be printed.

-BackingStore *BackingStoreType*
The **-BackingStore** flag causes the server to save the window contents so that when it is scrolled around the viewport, the window is painted from contents saved in server backing store. Redisplays of the drawing window can take up to a second or so. The *BackingStoreType* parameter can have one of the following values: **Always**, **WhenMapped** or **NotUseful**.
Tip: Enter a space between the **-BackingStore** flag and its *BackingStoreType* parameter.
Requirement: Use of this flag requires that the server be started with backing store enabled.

-page *Number*
Specifies the page number of the document to be first displayed.

ToolKitFlag
The following standard **X** Toolkit flags are commonly used with the **xpreview** command:

-bg *Color*
Specifies the color to use for the background of the window. The default is white.

-bg *Color*
Specifies the color to use for the background of the window. The default is white.

-fg *Color*
Specifies the color to use for displaying text. The default is black.

-geometry *Geometry*
Specifies the preferred size and position of the window.

-display *Host*:*Display*
Specifies the **X** server to contact.

-xrm *ResourceString*
Specifies a resource string to be used.

File
Specifies the file to be printed.

Examples

1. To build files output by the **troff** command into files that are suitable for use with the **xpreview** command, enter the following commands:

```
troff-TX100 troff-input | xpreview
pic -TX100 pic-troff-input | tbl | troff -man -TX100 | xpreview
```

2. To build files output by the **troff** command into files that are suitable for use with the Japanese language version of the **xpreview** command, enter the following commands:

```
LANG=ja_JP
troff -TX100 troff-input | xpreview -
pic -TX100 pic-troff-input | tbl | troff -man -TX100 \
        | xpreview -
```

Files

/usr/lib/X11/app-defaults/XPreview	Contains user-configurable applications defaults file.
/usr/lib/X11/Ja_JP/app-defaults/XPreview	Contains user-configurable applications default file for the Japanese (IBM-932) locale.
/usr/lib/X11/ja_JP/app-defaults/XPreview	Contains user-configurable applications default file for the Japanese (IBM-eucJP) locale.
/usr/lib/X11/ko_KR/app-defaults/XPreview	Contains user-configurable applications default file for the Korean locale.
/usr/lib/X11/zh_TW/app-defaults/XPreview	Contains user-configurable applications default file for the Traditional Chinese locale.
/usr/lib/font/devX100	Contains troff fonts for devX100 devices.
/usr/lib/X11/fonts	Contains X fonts for 100 dpi devices.
/usr/lib/X11/fonts/JP	Contains X fonts for multi-byte characters.
/usr/lib/X11/fonts/JP	Contains X fonts for Japanese characters.

Related Information

The **cat** command, **csplit** command, **diff** command, **lint** command, **lp** command, **lpr** command, **pg** command, **pr** command, **qprt** command, **sed** command, **sort** command, **tabs** command, **X** command, **xrdb** command.

The **eqn** command, **grap** command, **pic** command, **tbl** command, **troff** command, **X** command, **xrdb** command.

The **nl_langinfo** subroutine in *AIX 5L Version 5.3 Technical Reference: Base Operating System and Extensions*.

---

xprofiler Command

Purpose

Starts Xprofiler, a GUI-based AIX performance profiling tool.

Syntax

xprofiler [*program*] [**-b**] [**-s**] [**-z**] [**-a** *path*] [**-c** *file*] [**-L** *pathname*] [[**-e** *function*]...] [[**-E** *function*]...] [[**-f** *function*]...] [[**-F** *function*]...] [**-disp_max** *number_of_functions*] [[**gmon.out**]...]

xprofiler -h | **-help**

Description

The **xprofiler** command invokes Xprofiler, a GUI-based AIX performance profiling tool. Xprofiler is used to analyze the performance of both serial and parallel applications. Xprofiler uses data collected by the **-pg** compiling option and presents a graphical representation of the functions in the application in addition to providing textual data in several report windows. These presentation formats are intended to identify the functions which are most processor-intensive.

Flags

-a To specify an alternate search path or paths for library files and source code files. If more than one path is specified, the paths must be embraced by ″,″ and each path should be separated by either ″**:**″ or space.

-b Suppresses the printing of the field descriptions for the Flat Profile, Call Graph Profile, and Function Index reports when they are written to a file with the Save As option of the File menu.

-c Loads a configuration file that contains information to be used to determine which functions will be displayed when Xprofiler is brought up.

-disp_max Sets the number of function boxes that Xprofiler initially displays in the function call tree. The value supplied with this flag can be any integer between 0 and 5,000. Xprofiler displays the function boxes for the most processor-intensive functions through the number you specify. For instance, if you specify 50, Xprofiler displays the function boxes for the 50 functions in your program that consume the most processor. After this, you can change the number of function boxes that are displayed via the Filter menu options. This flag has no effect on the content of any of the Xprofiler reports.

-e De-emphasizes the general appearance of the function box or boxes for the specified functions in the function call tree, and limits the number of entries for these function in the Call Graph Profile report. This also applies to the specified function's descendants, as long as they have not been called by non-specified functions. In the function call tree, the function boxes for the specified functions appear greyed-out. Its size and the content of the label remain the same. This also applies to descendant functions, as long as they have not been called by non-specified functions. In the Call Graph Profile report, an entry for the specified function only appears where it is a child of another function, or as a parent of a function that also has at least one non-specified function as its parent. The information for this entry remains unchanged. Entries for descendants of the specified function do not appear unless they have been called by at least one non-specified function in the program.

-E Changes the general appearance and label information of the function box or boxes for the specified functions in the function call tree. Also limits the number of entries for these functions in the Call Graph Profile report, and changes the processor data associated with them. These results also apply to the specified function's descendants, as long as they have not been called by non-specified functions in the program. In the function call tree, the function box for the specified function appears greyed-out, and its size and shape also changes so that it appears as a square of the smallest allowable size. In addition, the processor time shown in the function box label, appears as 0 (zero). The same applies to function boxes for descendant functions, as long as they have not been called by non-specified functions. This option also causes the processor time spent by the specified function to be deducted from the left side processor total in the label of the function box for each of the specified function's ancestors. In the Call Graph Profile report, an entry for the specified function only appears where it is a child of another function, or as a parent of a function that also has at least one non-specified function as its parent. When this is the case, the time in the self and descendants columns for this entry is set to 0 (zero). In addition, the amount of time that was in the descendants column for the specified function is subtracted from the time listed under the descendants column for the profiled function. As a result, be aware that the value listed in the % time column for most profiled functions in this report will change.

-f De-emphasizes the general appearance of all function boxes in the function call tree, except for that of the specified function(s) and its descendant(s). In addition, the number of entries in the Call Graph Profile report for the non-specified functions and non-descendant functions is limited. The **-f** flag overrides the **-e** flag. In the function call tree, all function boxes except for that of the specified function(s) and it descendant(s) appear greyed-out. The size of these boxes and the content of their labels remain the same. For the specified function(s), and it descendants, the appearance of the function boxes and labels remain the same. In the Call Graph Profile report, an entry for a non-specified or non-descendant function only appears where it is a parent or child of a specified function or one of its descendants. All information for this entry remains the same.

-F Changes the general appearance and label information of all function boxes in the function call tree except for that of the specified function(s) and its descendants. In addition, the number of entries in the Call Graph Profile report for the non-specified and non-descendant functions is limited, and the processor data associated with them is changed. The **-F** flag overrides the **-E** flag. In the function call tree, the function box for the specified function appears greyed-out, and its size and shape also changes so that it appears as a square of the smallest allowable size. In addition, the processor time shown in the function box label, appears as 0 (zero). In the Call Graph Profile report, an entry for a non-specified or non-descendant function only appears where it is a parent or child of a specified function or one of its descendants. The time in the self and descendants columns for this entry is set to 0 (zero). When this is the case, the time in the self and descendants columns for this entry is set to 0 (zero). As a result, be aware that the value listed in the % time column for most profiled functions in this report will change.

-h -help Writes the Xprofiler usage to STDERR and then exits. The information includes **xprofiler** command line syntax and a description of Xprofiler runtime options.

-L Uses an alternate path name for locating shared libraries. If you plan to specify multiple paths, use the Set File Search Paths option of the File menu on the Xprofiler GUI.

-s If multiple **gmon.out** files are specified when Xprofiler is started, produces the **gmon.sum** profile data file. The **gmon.sum** file represents the sum of the profile information in all the specified profile files. Note that if you specify a single **gmon.out** file, the **gmon.sum** file contains the same data as the **gmon.out** file.

-z Includes functions that have both zero processor usage and no call counts in the Flat Profile, Call Graph Profile, and Function Index reports. A function will not have a call count if the file that contains its definition was not compiled with the **-pg** option, which is common with system library files.

Example

To use **xprofiler**, you must first compile your program (for example, **foo.c**) with **-pg**:

```
xlc -pg -o foo foo.c
```

1. When the program **foo** is executed, one **gmon.out** file will be generated for each processor involved in the execution. To invoke **xprofiler**, enter:

   ```
   xprofiler foo [[gmon.out]...]
   ```

Files

/usr/lib/X11/app-defaults/Xprofiler Location of the **xprofiler** command.

Related Information

Commands: **gprof(1)**, **xlc(1)**, **xlf(1)**.

---

xrdb Command

Purpose

X Server resource database utilities.

Syntax

xrdb [**-display** *Display*] [**-help**] [**-quiet**] [**-retain**] [**-cpp** *FileName* | **-nocpp**] [**-D** *Name*=*Value*] [**-I** *Directory*] [**-U** *Name*] [**-all** | **-global** | **-screen** | **-screens**] [**-n**] [**-edit** *FileName* | [**-backup** *String*] | **-merge** [*FileName*] | **-load** [*FileName*] | **-query** | **-remove** | **symbols**] **-override**]

Description

The **xrdb** command gets or sets the contents of the RESOURCE_MANAGER property on the root window of screen 0 or the SCREEN_RESOURCES property on the root window of any or all screens, or everything combined. You normally run this program from your X startup file.

Most X clients use the RESOURCE_MANAGER and SCREEN_RESOURCES properties to get user preferences about color, fonts, and so on for applications. Having this information in the server (where it is available to all clients) instead of on disk solves the problem in previous versions of X that required you to maintain *defaults* files on every machine that you might use. It also allows for dynamic changing of defaults without editing files.

The RESOURCE_MANAGER property specifies resources that apply to all screens of the display. The SCREEN_RESOURCES property on each screen specifies additional (or overriding) resources to be used for that screen. (When there is only one screen, SCREEN_RESOURCES is normally not used; all resources are placed in the RESOURCE_MANAGER property.)

For compatibility, if there is no RESOURCE_MANAGER property defined (either because the **xrdb** command was not run or if the property was removed), the resource manager looks for a file called **.Xdefaults** in your home directory.

The file name (or the standard input if - or no file name is given) is optionally passed through the C preprocessor with the following symbols defined, based on the capabilities of the server being used:

SERVERHOST=*Hostname*	Specifies the hostname portion of the display to which you are connected.
SRVR_*name*	Turns the **SERVERHOST** hostname string into a legal identifier. For example `my-dpy.lcs.mit.edu` becomes `SRVR_my_dpy_lcs_mit_edu`.
HOST=*Hostname*	Specifies the hostname portion of the display to which you are connected.
DISPLAY_NUM=*num*	Specifies the number of the display on the server host.
CLIENTHOST=*Hostname*	Specifies the name of the host on which xrdb is running.
CLNT_*name*	Turns the **CLIENTHOST** hostname string into a legal identifier. For example `expo.lcs.mit.edu` becomes `CLNT_expo_lcs_mit_edu`.
WIDTH=*Number*	Specifies the width of the default screen in pixels.
HEIGHT=*Number*	Specifies the height of the default screen in pixels.
X_RESOLUTION=*Number*	Specifies the x resolution of the default screen in pixels per meter.
Y_RESOLUTION=*Number*	Specifies the y resolution of the default screen in pixels per meter.
PLANES=*Number*	Specifies the number of bit planes (the depth) of the root window of the default screen.
RELEASE=*Number*	Specifies the vendor release number for the server. The interpretation of this number varies depending on **VENDOR**.
REVISION=*Number*	Specifies the X protocol minor version supported by this server (currently 0).
VERSION=*Number*	Specifies the X protocol major version supported by this server (should always be 11).
VENDOR=*Vendor*	A string specifying the vendor of the server.
VNDR_*name*	Turns the **VENDOR** name string into a legal identifier. For example `MIT X Consortium` becomes `VNDR_MIT_X_Consortium`.
EXT_*name*	Turns each extension string into a legal identifier. A symbol is defined for each protocol extension supported by the server. For example `X3D-PEX` becomes `EXT_X3D_PEX`.
NUM_SCREENS=*num*	Specifies the total number of screens.
SCREEN_NUM=*num*	Specifies the number of current screen. from 0 (zero).

BITS_PER_RGB=*Number*	Specifies the number of significant bits in an RGB color specification. This is the log base 2 of the number of distinct shades of each primary that the hardware can generate. Note that it is not related to **PLANES**.
CLASS=*VisualClass*	Specifies the visual class of the root window of the default screen which is one of the following:
CLASS_*visualclass*=*visualid*	Specifies the visual class of the root window in a form can *#ifdef* on. The value is the numeric id of the visual.
	DirectColor, **GrayScale**, **PseudoColor**, **StaticColor**, **StaticGray**, **TrueColor**
CLASS_*visualclass_depth*=*num*	A symbol is defined for each visual supported for the screen. The symbol includes the class of the visual and its depth; the value is the numeric id of the visual. (If more than one visual has the same class and depth, the numeric id of the first one reported by the server is used.)S
COLOR	Defined only if **CLASS** is one of **StaticColor**, **PseudoColor**, **TrueColor**, or **DirectColor**.

Comment lines begin with an ! (exclamation mark) and are ignored.

Since **xrdb** can be read from standard input, use it to change the contents of properties directly from a terminal or from a shell script.

Flags

-all	Indicates that operation is performed on the screen-independent resource property (RESOURCE_MANAGER), as well as the screen-specific property (SCREEN_RESOURCES) on every screen of the display. For example, when used in conjunction with **-query**, the contents of all properties are output. For **-load** and **-merge**, the input file is processed once for each screen. The resources that occur in common in the output for every screen are collected and applied as the screen-independent resources. The remaining resources are applied for each individual per-screen property. This is the default mode of operation. This option is specific to X11R5.
-backup *String*	Specifies a suffix to append to the file name. Use it with **-edit** to generate a backup file. **-edit** is a prerequisite for **-backup** *String.*
-cpp *FileName*	Specifies the pathname of the C preprocessor program to use. Although the **xrdb** command was designed to use CPP, any program that acts as a filter and accepts the **-D**, **-I**, and **-U** flags can be used.
-D*Name***=***Value*	Passes through to the preprocessor and defines symbols to use with conditionals such as `#ifdef`**.**
-display *Display*	Specifies the X Server to use. It also specifies the screen to use for the **-screen** option, and it specifies the screen from which preprocessor symbols are derived for the **-global** option.
-edit *FileName*	Indicates that the contents of the specified properties should be edited into the given file, replacing any values listed there. This allows you to put changes you made to your defaults back into your resource file, preserving any comments or preprocessor lines.
-global	Indicates that the operation should only be performed on the screen-independent RESOURCE_MANAGER property. This option is specific to X11R5.
-help	Prints a brief description of the allowed flags.
-I*Directory*	(uppercase i) Passes through to the preprocessor and specifies a directory to search for files referenced with `#include`**.**
-load	Indicates that the input is loaded as the new value of the specified properties, replacing the old contents. This is the default action.

-merge	Indicates that the input merges with, instead of replaces, the current contents of the specified properties. This option performs a lexicographic sorted merge of the two inputs, which is probably not what you want, but remains for backward compatibility.
-n	Indicates that changes to the specified properties (when used with **-load** or **-merge**) or to the resource file (when used with **-edit**) should be shown on the standard output, but should not be performed. This option is specific to X11R5.
-nocpp	Indicates that the **xrdb** command should not run the input file through a preprocessor before loading it into properties.
-override	Indicates that the input should be added to, instead of replacing, the current contents of the specified properties. New entries override previous entries.
-query	Indicates that the current contents of the specified properties should print onto the standard output. Note that since preprocessor commands in the input resource file are part of the input file, not part of the property, they do not appear in the output from this flag.
-quiet	Indicates that a warning about duplicate entries should not display. This option is specific to X11R5.
-remove	Indicates that the specified properties should be removed from the server.
-retain	Indicates that the server should be instructed not to reset if the **xrdb** command is the first client. This should never be necessary under normal conditions, since the **xdm** and **xinit** commands always act as the first client. This option is specific to X11R5.
-screen	Indicates that the operation should only be performed on the SCREEN_RESOURCES property of the default screen of the display. This option is specific to X11R5.
-screens	Indicates that the operation should be performed on the SCREEN_RESOURCES property of each screen of the display. For **-load** and **-merge**, the input file is processed once for each screen. This option is specific to X11R5.
-symbols	Indicates that the symbols defined for the preprocessor should be printed onto the standard output.
-U*Name*	Passes through to the preprocessor and removes any definitions of this symbol.

Examples

1. To load a file into the database:

   ```
   xrdb -load myfile
   ```

2. To take the contents of the database just loaded and edit or put it into newfile:

   ```
   xrdb -edit newfile
   ```

Files

The **xrdb** command generalizes the **~/.Xdefaults** files.

---

xsend Command

Purpose

Sends secret mail in a secure communication channel.

Syntax

xsend *User*

Description

The **xsend** command sends messages that can be read only by the intended recipient. This command is similar to the **mail** command, but the mail sent with this command is intended to be secret.

The **xsend** command is used with the **enroll** command and the **xget** command to send secret mail. The **enroll** command sets up the password used to receive secret mail. The **xget** command uses that password to receive the mail.

The **xsend** command reads standard input until an EOF (Ctrl-D) or a . (period) is entered. It then encrypts this text along with some header information and sends it. After sending the encrypted message,the **xsend** command mails a standard mail message to the recipient informing them they have received secret mail.

Note: Secret mail can only be sent to local users.

Examples

1. To send secret mail, enter:

   ```
   xsend ron
   ```

   When you have issued the **xsend** command with the recipient's name, the mail system is used to enter the text of the message. When you finish entering the message to user `ron`, press the Enter key, then Ctrl-D or a . (period) to exit the mail editor and send the message. The **xsend** command encrypts the message before it is sent.
2. To send a file to another user, enter:

   ```
   xsend lance <proposal
   ```

   In this example, the file `proposal` is sent to user `lance`.

Files

/var/spool/secretmail/*.keys	Contains the encrypted key for *User.*
/var/spool/secretmail/*.[0-9]	Contains the encrypted mail messages for *User*.
/usr/bin/xsend	Contains the command executable files.

Related Information

The **bellmail** command, **enroll** command, **mail** command, **xget** command.

Mail applications, Sending and receiving secret mail in *Networks and communication management*.

---

xset Command

Purpose

Sets options for your X-Windows environment.

Syntax

xset [**-display** *Display*] [**b** [*Volume* [*Pitch* [*Duration*]]] | **-b** | **b on** | **b off**] [**bc** | **-bc**] **c** [*Volume*] | **-c** | **c on** | **c off**] [[**-** | **+**] **fp** [**-** | **+** | **=**] **Path** [**,***Path***,** [*...*]]] [**fp default**] [**fp rehash**] [[**-**] **led** [*Integer*]] [**led on** | **led off**] [**m** [*Accelerator*] [*Threshold*]]] [**m** [**ouse**] **default**] [**p** *Pixel Color*] [[**-**] **r**] [**r on** | **r off**] [**s** [*Length* [*Period*]]] [**s blank** | **s noblank**] [**s expose** | **s noexpose**] [**s on** | **s off**] [**s activate**] [**s reset**] [**s default**] [**q**]

Description

The **xset** command customizes your X-Windows environment.

Flags

-display *Host*:*Display*
Specifies the X server to use. For more information about servers, see the **X** command.

b or **b on**
Turns the bell on. This is the default setting.

> **Note:** Not all hardware is able to vary the bell characteristics, but for that which can, all of the **b** flag permutations and its variables are available.

b [*Volume* [*Pitch* [*Duration*]]]
Specifies the bell volume, pitch, and duration. This flag accepts up to three numeric values.

Volume If only one numeric is given then it is assumed to be *Volume*. The bell volume is set to that numeric as a percentage of the bell's maximum possible volume dependent on current hardware capabilities.

Pitch The second numeric in hertz values, is the tonal sound of the bell.

Duration
The third numeric in milliseconds, is the length of time that the bell rings.

-b or **b off**
Turns the bell off.

bc or **-bc**
Controls bug compatibility mode in the server, if possible. A preceding **-** (dash) disables this mode; otherwise, bug compatibility mode is enabled. The server must support the MIT-SUNDRY-NONSTANDARD protocol extension for the **bc** flag to work.

New application development should be performed with bug compatibility mode disabled.

The **bc** flag is provided for pre-X11 Release 4 (X11R4) clients. Some pre-X11R4 clients pass illegal values in various protocol requests. Such clients, when run with an X11R4 server, end abnormally or otherwise fail to operate correctly.

This flag explicitly reintroduces certain bugs into the X server so that such clients still can be run.

c or **c on**
Turns on the click. System default.

c *Volume*
A numeric from 0 to 100 that specifies a percentage of the click's maximum possible volume dependent on current hardware capabilities.

-c or **c off**
Turns off the click.

fp=*Path*,...
Sets the font path to the directories given in the *Path* parameter. The directories are interpreted by the server, not by the client, and are server-dependent. The server ignores directories that do not contain font databases created by the **mkfontdir** command. All of the options and variables supported by the **fp** flag are available.

fp- or **-fp**
Deletes the font path specified by the *Path* parameter from the end of the current font path if the **-** (dash) precedes **fp** and from the front of the font path if the **-** (dash) follows **fp.**

fp+ or **+fp**
Adds the font path specified by the *Path* parameter to the bottom of font list if the **-** (dash) precedes **fp** and from the end of the font path if the **-** (dash) follows **fp**.

fp default
Resets the font path to the server's default.

fp rehash
Causes the server to reread the font databases in the current font path. Usually used only when adding new fonts to a font directory after running **mkfontdir** to recreate the font database.

led or**led on**
Turns all LEDs on.

-led *Integer*
Turns the LED specified by *Integer* off. Valid values are between 1 and 32.

led *Integer*
Turns the LED specified by *Integer* on. Valid values are between 1 and 32.

-led or **led off**
Turns all LEDs off.

Note: Not all hardware assigns the same *Integer* variables to the same LED functions.

m
Allows you to control the precision of the mouse or other pointing device. If no variable or the **default** argument is specified, the system defaults are used. This flag accepts the following optional arguments and parameters:

Acceleration
Sets the multiplier for the mouse movement. The value can be specified as an integer or a fraction.

Threshold
Sets the minimum number of pixels needed to invoke a movement of the mouse. The value is specified in pixels.

If only one parameter is given, it will be interpreted as the *Acceleration* parameter.

default Uses the system defaults.

p
Controls pixel color values. The root background colors may be changed on some servers by altering the entries for BlackPixel and WhitePixel. Although these values are often **0** and **1**, they need not be.

Also, a server may choose to allocate those colors privately, in which case the **xset** command generates an error. The **xset** command also generates an error if the map entry is a read-only color.

Valid parameters are:

Pixel Specifies the color map entry number in decimal.

Color Specifies a color.

r or **r on**
Enables autorepeat.

-r or **r off**
Disables autorepeat.

s or **s default**
Sets screen saver parameters to the default screen-saver characteristics.

s [*Length*[*Period*]]
Specifies the length of time the server must be inactive for the screen saver to activate. *Period* specifies the period in which the background pattern must be changed to avoid burn in. The values of *Length* and *Period* are specified in seconds. If only one numerical parameter is given, it is read as a *Length* parameter.

s on or **s off**
Turns the screen saver functions on and off, respectively.

s activate	Causes the screen saver to activate, even if it has been turned off.
s reset	Causes the screen saver to deactivate if it was activated.
s blank	Sets the preference to blank the video (if the hardware can do so) rather than display a background pattern.
s noblank	Sets the preference to display a pattern rather than blank the video.
s expose	Sets the preference to allow window exposures (the server can freely discard window contents).
s noexpose	Sets the preference to disable screen saver unless the server can regenerate the screens without causing exposure events.
q	Reports information on the current settings. These settings will be reset to default values when you log out.

Note: Not all X implementations are guaranteed to honor all of these options.

Examples

1. To set the bell volume to medium, the tone to 50 hertz, and length of time the bell rings to 50 milliseconds:

   ```
   xset b 50,50,50
   ```

2. To set the font path to the **/usr/lib/X11/fonts** directory:

   ```
   xset fp= /usr/lib/x11/fonts
   ```

3. To cause the server to reread the font databases in the current font path:

   ```
   xset fp rehash
   ```

4. To see information on the current settings:

   ```
   xset q
   ```

   which produces output similar to the following:

   ```
   Keyboard Control:
     auto repeat:  on    key click percent:  0    LED mask:  00000000
     auto repeating keys:  0000000000000000
                           0000000000000000
                           0000000000000000
                           0000000000000000
     bell percent:  50    bell pitch:  400    bell duration:  100

   Pointer Control:
     acceleration:  2 = 2 / 1    threshold:  4

   Screen Saver:
     prefer blanking:  no    allow exposures:  no
     timeout:  0    cycle:  0

   Colors:
     default colormap:  0x8006e    BlackPixel:  0    WhitePixel:  1

   Font Path:
      /usr/lib/X11/fonts/,/usr/lib/X11/fonts/75dpi/,/usr/lib/X11/fonts/100dpi/,/usr/
   lib/X11/fonts/oldx10/,/usr/lib/X11/fonts/oldx11/,/usr/lib/X11/fonts/bmug/,/usr/l
   ib/X11/fonts/info-mac/,/usr/lib/X11/fonts/JP/,/usr/lib/X11/fonts/misc/
   ```

Related Information

The **X** command, **xmodmap** command, **xrdb** command, **xsetroot** command.

---

xsetroot Command

Purpose

Sets the root window parameters for the **X** command.

Syntax

xsetroot [**-bg** *Color*] [**-cursor** *CursorFile MaskFile*] [**-cursor_name** *CursorName*] [**-def**] [**-display** *Display*] [**-fg** *Color*] [**-help**] [**-name** *String*] [**-rv**] [**-bitmap** *FileName* | **-gray** | **-grey** | **-mod** *X Y* | **-solid** *Color*]

Description

The **xsetroot** command allows you to tailor the appearance of the background (root) window on a workstation display running X. Normally, you experiment with the **xsetroot** command until you find a personalized look that you like, then put the **xsetroot** command that produces it into your X startup file. If no options are specified or if the **-def** flag is specified, the window is reset to its default state. The **-def** flag can be specified with other flags and only the unspecified characteristics are reset to the default state.

Only one of the background color (tiling) changing flags (**-bitmap**, **-solid**, **-gray**, **-grey**, or **-mod**) can be specified at a time.

Flags

-bg *Color*	Uses the *Color* parameter as the background color.
-bitmap *FileName*	Uses the bitmap specified in the file to set the window pattern. You can make your own bitmap files (little pictures) using the **bitmap** program. The entire background is made of repeated tiles of the bitmap.
-cursor *CursorFile MaskFile*	Changes the pointer cursor to what you want when it is outside of any window. Cursor and mask files are bitmaps (little pictures) that can be made with the **bitmap** program. You probably want the mask file to be all black until you get used to the way masks work.
-cursor_name *CursorName*	Changes the pointer cursor to one of the standard cursors from the cursor font.
-def	Resets unspecified attributes to the default values. (Restores the background to the familiar gray mesh and the cursor to the hollow x shape.)
-display *Display*	Specifies the server connection. See the **X** command.
-fg *Color*	Uses the *Color* parameter as the foreground color. Foreground and background colors are meaningful only with the **-cursor**, **-bitmap**, or **-mod** flags.
-gray	Makes the entire background gray.
-grey	Makes the entire background grey.
-help	Prints a usage message and exits.
-mod *X Y*	Makes a plaid-like grid pattern on your screen. The *X* and *Y* parameters are integers ranging from 1 to 16. Zero and negative numbers are taken as 1.

-name *String*	Sets the name of the root window to the *String* parameter. There is no default value. Usually a name is assigned to a window so that the window manager can use a text representation when the window is iconified. This flag is not used because you cannot iconify the background.
-rv	Exchanges the foreground and background colors. Normally the foreground color is black and the background color is white.
-solid *Color*	Sets the background of the root window to the specified color. This flag is only used on color servers.

Related Information

The **X** command, **xset** command, **xrdb** command.

---

xss Command

Purpose

Improves the security of unattended workstations.

Syntax

xss [**-e** *CommandString*] [**-timeout** *Seconds*] [**-display** *DisplayPtr*] [**-v**] [**-fg** *Color*] [**-bg** *Color*] [**-geometry** *wxh+x+y*]

Description

The **xss** command works with the newly added Massachusetts Institute of Technology (MIT) Screen Saver Extensions in order to implement a user controllable screen saver/lock. This command is designed to improve the security of unattended workstations.

The **xss** command executes a user-specified command string when it receives a screen saver timeout message, or when the user activates the pushbutton. When no user-specified command is given, the **xss** command defaults to the **xlock** command.

> **Note:** The **xss** command only uses the newly added MIT Screen Saver Extensions. The **xss** command does not work on an older X server, or when using an older X extension library.

Flags

-e *CommandString*	Sets the **xss** command to execute when either the screen saver times out, or the user activates the pushbutton. Note that if the *CommandString* parameter value is longer than one word, it must be surrounded by ″ ″ (double quotations).
-timeout *Seconds*	Sets the number of seconds of user inactivity before the screen saver times out, and causes the **xss** command to run the *CommandString* parameter.
-display *DisplayPtr*	Sets the connection to the X11 display.
-v	Turns on verbose mode.
-fg *Color*	Sets the foreground color of the pushbutton.
-bg *Color*	Sets the background color of the pushbutton.
-geometry *wxh+x+y*	Specifies the size and location of the client window.

Examples

When running remotely and using the **-display** flag for the **xss** command, remember that you may also have to use the **-display** flag option for the command that will be executed by the **xss** command. See the following running remote example:

1. Running remote:
   ```
   xss -display myhost:0 -e "xlock -remote -display myhost:0"
   ```
2. Screen saver only:
   ```
   xss -e "xlock -nolock"
   ```
3. Simple example:
   ```
   xss -e xlock
   ```

xstr Command

Purpose

Extracts strings from C programs to implement shared strings.

Syntax

xstr [**-v**] [**-c**] [**-**] [*File*]

Description

The **xstr** command maintains a file **strings** into which strings in component parts of a large program are hashed. These strings are replaced with references to this array. This serves to implement shared constant strings, most useful if they are also read-only.

The command:

```
xstr -c File
```

extracts the strings from the C source in the *File* parameter, replacing string references by expressions of the form (**&xstr**[*number*]) for some number. An appropriate declaration of the xstr array is prepended to the file. The resulting C text is placed in the file **x.c**, to then be compiled. The strings from this file are appended into the **strings** file if they are not there already. Repeated strings and strings which are suffixes of existing strings do not cause changes to the file **strings**.

If a string is a suffix of another string in the file but the shorter string is seen first by the **xstr** command, both strings are placed in the file **strings**.

After all components of a large program have been compiled, a file **xs.c** declaring the common xstr array space can be created by a command of the form:

```
xstr
```

This **xs.c** file should then be compiled and loaded with the rest of the program. If possible, the array can be made read-only (shared), saving space and swap overhead.

The **xstr** command can also be used on a single file. The command:

```
xstr File
```

creates files **x.c** and **xs.c** as before, without using or affecting any **strings** file in the same directory.

It may be useful to run the **xstr** command after the C preprocessor if any macro definitions yield strings or if there is conditional code which contains strings which may not, in fact, be needed.

The **xstr** command reads from its standard input when the **-** (minus sign) flag is given and does not alter the **strings** file unless the **-c** flag is specified also.

An appropriate command sequence for running the **xstr** command after the C preprocessor is:

```
cc -E name.c | xstr -c -
cc -c x.c
mv x.o name.o
```

The **xstr** command does not touch the file **strings** unless new items are added, thus the **make** command can avoid remaking the **xs.o** file unless truly necessary.

Flags

-c Extracts strings from the specified file, and places them in the **strings** file.
-v Verbose mode. Tells when strings are found, or new in the **strings** file.
- Reads from standard input.

Examples

1. To extract the strings from the C source in the *File.c* parameter, replacing string references by expressions of the form (**&xstr**[*number*]):

   ```
   xstr -c File.c
   ```

   An appropriate declaration of the xstr array is prepended to the file. The resulting C text is placed in the file **x.c**, to then be compiled.
2. To declare the common xstr array space in the **xs.c** file:

   ```
   xstr
   ```

Files

strings	File which contains the extracted strings.
x.c	Massaged C source.
xs.c	C source for definition of array xstr.
/tmp/xs*	Temporary file when **xstr** command does not touch the **strings** file.
/usr/ccs/bin/mkstr	Contains an executable file.
/**usr/ccs/bin/mkstr**	Contains an executable file for Berkeley environment.

Related Information

The **mkstr** command.

---

xterm Command

Purpose

Provides a terminal emulator for the X Window System.

Note: The **xterm** command is ported from the Massachusetts Institute of Technology (MIT) X Window System, Version 11, Release 6 with no functional enhancements. The **xterm** command does not have support for localization or internationalization. For the localized and internationalized terminal emulator, the user can use the **aixterm** or **dtterm** commands.

Syntax

xterm [-*ToolkitOption* ...] [-*Option* ...]

Description

The **xterm** program is a terminal emulator for the X Window System. It provides DEC VT102 and Tektronix 4014 compatible terminals for programs that cannot use the window system directly. If the underlying

operating system supports terminal resizing capabilities, the **xterm** program uses the facilities to notify programs running in the window whenever it is resized.

The VT102 and Tektronix 4014 terminals each have their own window so that you can edit text in one and look at graphics in the other at the same time. To maintain the correct aspect ratio (height/width), Tektronix graphics are restricted to the largest box with a 4014 aspect ratio that will fit in the window. This box is located in the upper left area of the window.

Although both windows might be displayed at the same time, one of them is considered the *active window* for receiving keyboard input and terminal output. This is the window that contains the text cursor. The active window can be chosen through escape sequences, the VT Options menu in the VT102 window, and the Tek Options menu in the 4014 window.

Emulations

The VT102 emulation is fairly complete, but does not support smooth scrolling, VT52 mode, the blinking character attribute, or the double-wide and double-size character sets. The **termcap** file entries that work with the **xterm** command include **xterm**, **vt102**, **vt100** and ``ansi,'' and the **xterm** command automatically searches the **termcap** file in this order for these entries and then sets the **TERM** and the **TERMCAP** environment variables.

Many of the special **xterm** features might be modified under program control through a set of escape sequences different from the standard VT102 escape sequences.

The Tektronix 4014 emulation is also fairly good. It supports 12-bit graphics addressing, scaled to the window size. Four different font sizes and five different lines types are supported. There is no write-thru or defocused mode support.

The Tektronix text and graphics commands are recorded internally by the **xterm** command and may be written to a file by sending the COPY escape sequence (or through the Tektronix menu, as described in the following sections). The name of the file will be **COPY***yy*-*MM*-*dd*.*hh*:*mm*:*ss*, where *yy*, *MM*, *dd*, *hh*, *mm*, and *ss* are the year, month, day, hour, minute, and second when the copy is performed (the file is created in the directory that the **xterm** command is started in, or the home directory for a login **xterm**).

Other Features

The **xterm** command automatically highlights the text cursor when the pointer enters the window (selected) and unhighlights it when the pointer leaves the window (unselected). If the window is the focus window, the text cursor is highlighted no matter where the pointer is located.

In VT102 mode, there are escape sequences to activate and deactivate an alternate screen buffer, which is the same size as the display area of the window. When activated, the current screen is saved and replaced with the alternate screen. Saving of lines scrolled off the top of the window is disabled until the usual screen is restored.

The **termcap** file entry for the **xterm** command allows the **vi** command editor to switch to the alternate screen for editing and to restore the screen on exit.

In either VT102 or Tektronix mode, there are escape sequences to change the name of the windows.

Options

The **xterm** terminal emulator accepts all of the standard X Toolkit command-line options as well as the following (if the option begins with a + instead of a -, the option is restored to its default value):

-help Causes the **xterm** command to print out a message describing its options.

-132 Usually, the VT102 DECCOLM escape sequence that switches between 80- and 132-column mode is ignored. This option causes the DECCOLM escape sequence to be recognized, and the xterm window will resize appropriately.

-ah Indicates that the **xterm** command should always highlight the text cursor. By default, the **xterm** command will display a hollow text cursor whenever the focus is lost or the pointer leaves the window.

+ah Indicates that the **xterm** command should do text cursor highlighting based on focus.

-b *Number* Specifies the size of the inner border (the distance between the outer edge of the characters and the window border) in pixels. The default is 2.

-cc *CharacterClassRange*:*Value*[**,**...] Sets classes indicated by the given ranges for use in selecting by words.

-cn Indicates that newlines should not be cut in line-mode selections.

+cn Indicates that newlines should be cut in line-mode selections.

-cr *Color* Specifies the color to use for the text cursor. The default is to use the same foreground color that is used for text.

-cu Indicates that the **xterm** command should work around a bug in the **more** program that causes it to incorrectly display lines that are exactly the width of the window and are followed by a line beginning with a tab (the leading tabs are not displayed). This option is so named because it was originally thought to be a bug in the **curses** function cursor motion package.

+cu Indicates that **xterm** should not work around the **more** function bug previously mentioned.

-e *Program* [*Arguments*] Specifies the program (and its command-line arguments) to be run in the xterm window. It also sets the window title and icon name to be the base name of the program being run if neither the **-T** nor the **-n** option is given on the command line.
Note: This must be the last option on the command line.

-fb *Font* Specifies a font to be used when displaying bold text. This font must be the same height and width as the normal font. If only one of the normal or bold fonts is specified, it will be used as the normal font and the bold font will be produced by overstriking this font. The default is to do overstriking of the normal font.

-i Turns on the **useInsertMode** resource.

+i Turns off the **useInsertMode** resource.

-j Indicates that the **xterm** command should do jump scrolling. Usually, text is scrolled one line at a time; this option allows the **xterm** command to move multiple lines at a time so that it does not fall as far behind. Its use is strongly recommended because it makes the **xterm** command much faster when scanning through large amounts of text. The VT100 escape sequences for enabling and disabling smooth scrolling as well as the VT Options menu can be used to turn this feature on or off.

+j Indicates that the **xterm** command should not do jump scrolling.

-ls Indicates that the shell that is started in the xterm window is a login shell (in other words, the first character of the *ArgumentVector* parameter is a dash, indicating to the shell that it should read the user's **.login** or **.profile** file).

+ls Indicates that the shell that is started should not be a login shell (in other words, it will be a usual subshell).

-mb Indicates that the **xterm** command should ring a margin bell when the user types near the right end of a line. This option can be turned on and off from the VT Options menu.

+mb Indicates that the margin bell should not be rung.

-mc *Milliseconds* Specifies the maximum time between multiclick selections.

-ms *Color* Specifies the color to be used for the pointer cursor. The default is to use the foreground color.

-nb *Number* Specifies the number of characters from the right end of a line at which the margin bell, if enabled, will ring. The default is 10.

-rw Indicates that reverse wraparound should be allowed. This allows the cursor to back up from the leftmost column of one line to the rightmost column of the previous line. This is very useful for editing long shell command lines and is encouraged. This option can be turned on and off from the VT Options menu.

+rw Indicates that reverse wraparound should not be allowed.

-aw Indicates that auto wraparound should be allowed. This allows the cursor to automatically wrap to the beginning of the next line when it is at the rightmost position of a line and text is output.

+aw Indicates that auto wraparound should not be allowed.

-s Indicates that the **xterm** command may scroll asynchronously, meaning that the screen does not have to be kept completely up to date while scrolling. This allows the **xterm** command to run faster when network latencies are high and is typically useful when running across a large Internet or many gateways.

+s Indicates that the **xterm** command should scroll synchronously.

-sb Indicates that some number of lines that are scrolled off the top of the window should be saved and that a scrollbar should be displayed so that those lines can be viewed. This option can be turned on and off from the VT Options menu.

+sb Indicates that a scrollbar should not be displayed.

-sf Indicates that Sun Function Key escape codes should be generated for function keys.

+sf Indicates that the standard escape codes should be generated for function keys.

-si Indicates that output to a window should not automatically reposition the screen to the bottom of the scrolling region. This option can be turned on and off from the VT Options menu.

+si Indicates that output to a window should cause it to scroll to the bottom.

-sk Indicates that pressing a key while using the scrollbar to review previous lines of text should cause the window to be repositioned automatically in the usual position at the bottom of the scroll region.

+sk Indicates that pressing a key while using the scrollbar should not cause the window to be repositioned.

-sl *Number* Specifies the number of lines to save that have been scrolled off the top of the screen. The default is 64.

-t Indicates that the **xterm** command should start in Tektronix mode, rather than in VT102 mode. Switching between the two windows is done using the Options menus.

+t Indicates that the **xterm** command should start in VT102 mode.

-tm *String* Specifies a series of terminal-setting keywords followed by the characters that should be bound to those functions, similar to the **stty** program. Allowable keywords include: **intr**, **quit**, **erase**, **kill**, **eof**, **eol**, **swtch**, **start**, **stop**, **brk**, **susp**, **dsusp**, **rprnt**, **flush**, **weras**, and **lnext**. Control characters might be specified as ^*Character* (for example, ^c or ^u), and ^? may be used to indicate Delete.

-tn *Name* Specifies the name of the terminal type to be set in the **TERM** environment variable. This terminal type must exist in the **termcap** database and should have **li#** and **co#** entries.

-ut Indicates that the **xterm** command should not write a record into the **/etc/utmp** system log file.

+ut Indicates that the **xterm** command should write a record into the **/etc/utmp** system log file.

-vb Indicates that a visual bell is preferred over an audible one. Instead of ringing the terminal bell whenever the Ctrl+G key sequence signal is received, the window will flash.

+vb Indicates that a visual bell should not be used.

-wf Indicates that the **xterm** command should wait for the window to be mapped the first time before starting the subprocess so that the initial terminal size settings and environment variables are correct. It is the application's responsibility to catch subsequent terminal size changes.

+wf Indicates that the **xterm** command should not wait before starting the subprocess.

-C Indicates that this window should receive console output. This is not supported on all systems. To obtain console output, you must be the owner of the console device, and you must have read and write permission for it. If you are running X windows under **xdm** on the console screen, you may need to have the session startup and reset programs explicitly change the ownership of the console device in order to get this option to work.

-S*ccn* Specifies the last two letters of the name of a pseudoterminal to use in slave mode, plus the number of the inherited file descriptor. The option is parsed ``%c%c%d''. This allows the **xterm** command to be used as an input and output channel for an existing program and is sometimes used in specialized applications.

The following command-line arguments are provided for compatibility with older versions. They may not be supported in the next release as the X Toolkit provides standard options that accomplish the same task.

%*geom* Specifies the preferred size and position of the Tektronix window. It is shorthand for specifying the ***tekGeometry** resource.

#*geom* Specifies the preferred position of the icon window. It is shorthand for specifying the ***iconGeometry** resource.

-T *String* Specifies the title for the **xterm** program's windows. It is equivalent to **-title**.

-n *String* Specifies the icon name for the **xterm** program's windows. It is shorthand for specifying the ***iconName** resource. Note that this is not the same as the Toolkit option **-name** (see the following). The default icon name is the application name.

-r Indicates that reverse video should be simulated by swapping the foreground and background colors. It is equivalent to **-rv**.

-w *Number* Specifies the width in pixels of the border surrounding the window. It is equivalent to **-borderwidth** or **-bw**.

The following standard X Toolkit command-line arguments are commonly used with the **xterm** command:

-bg *Color* Specifies the color to use for the background of the window. The default is white.

-bd *Color* Specifies the color to use for the border of the window. The default is black.

-bw *Number* Specifies the width in pixels of the border surrounding the window.

-fg *Color* Specifies the color to use for displaying text. The default is black.

-fn *Font* Specifies the font to be used for displaying usual text. The default is fixed.

-name *Name* Specifies the application name under which resources are to be obtained, rather than the default executable file name. The *Name* parameter should not contain . (dot) or * (asterisk) characters.

-title *String* Specifies the window title string, which may be displayed by window managers if the user so chooses. The default title is the command line specified after the **-e** option, if any; otherwise, the application name.

-rv Indicates that reverse video should be simulated by swapping the foreground and background colors.

-geometry *Geometry* Specifies the preferred size and position of the VT102 window; see the **X** command.

-display *Display* Specifies the X server to contact; see the **X** command.

-xrm *ResourceString* Specifies a resource string to be used. This is especially useful for setting resources that do not have separate command-line options.

-iconic Indicates that the **xterm** command should ask the window manager to start it as an icon rather than as the usual window.

Resources

The program understands all of the core X Toolkit resource names and classes as well as:

iconGeometry (class **IconGeometry**) Specifies the preferred size and position of the application when iconified. It is not necessarily obeyed by all window managers.

termName (class **TermName**) Specifies the terminal type name to be set in the **TERM** environment variable.

title (class **Title**) Specifies a string that may be used by the window manager when displaying this application.

ttyModes (class **TtyModes**) Specifies a string containing terminal-setting keywords and the characters to which they may be bound. Allowable keywords include: **intr**, **quit**, **erase**, **kill**, **eof**, **eol**, **swtch**, **start**, **stop**, **brk**, **susp**, **dsusp**, **rprnt**, **flush**, **weras**, and **lnext**. Control characters may be specified as ^*Character* (for example, ^c or ^u) and ^? may be used to indicate Delete. This is very useful for overriding the default terminal settings without having run an **stty** program every time an **xterm** window is started.

useInsertMode (class **useInsertMode**)
Forces the use of insert mode by adding appropriate entries to the **TERMCAP** environment variable. This is useful if the system termcap is broken. The default is **false**.

utmpInhibit (class **UtmpInhibit**)
Specifies whether **xterm** should try to record the user's terminal in **/etc/utmp**.

sunFunctionKeys (class **SunFunctionKeys**)
Specifies whether Sun Function Key escape codes should be generated for function keys instead of standard escape sequences.

waitForMap (class **WaitForMap**)
Specifies whether the **xterm** command should wait for the initial window map before starting the subprocess. The default is False.

The following resources are specified as part of the **vt100** widget (class **VT100**):

allowSendEvents (class **AllowSendEvents**)
Specifies whether synthetic key and button events (generated using the X protocol **SendEvent** request) should be interpreted or discarded. The default is False, meaning they are discarded. Note that allowing such events creates a large security hole.

alwaysHighlight (class **AlwaysHighlight**)
Specifies whether **xterm** should always display a highlighted text cursor. By default, a hollow text cursor is displayed whenever the pointer moves out of the window or the window loses the input focus.

appcursorDefault (class **AppcursorDefault**)
If True, the cursor keys are initially in application mode. The default is False.

appkeypadDefault (class **AppkeypadDefault**)
If True, the keypad keys are initially in application mode. The default is False.

autoWrap (class **AutoWrap**)
Specifies whether auto wraparound should be enabled. The default is True.

bellSuppressTime (class **BellSuppressTime**)
Specifies the number of milliseconds after a bell command is sent during which additional bells will be suppressed. The default is 200. If set to nonzero, additional bells will also be suppressed until the server reports that processing of the first bell has been completed; this feature is most useful with the visible bell.

boldFont (class **BoldFont**)
Specifies the name of the bold font to use instead of overstriking.

c132 (class **C132**)
Specifies whether the VT102 DECCOLM escape sequence should be honored. The default is False.

charClass (class **CharClass**)
Specifies comma-separated lists of character class bindings of the form [*low*-]*high*:*value*. These are used in determining which sets of characters should be treated the same when doing cut and paste. See "Character Classes" on page 230.

curses (class **Curses**)
Specifies whether the last column bug in the **curses** function should be worked around. The default is False.

cutNewline (class **cutNewline**)
If **false**, triple clicking to select a line does not include the Newline at the end of the line. If **true**, the Newline is selected. The default is **true**.

cutToBeginningofLines (class **CutToBeginningOfLine**)
If **false**, triple clicking to select a line selects only from the current word forward. If **true**, the entire line is selected. The default is **true**.

background (class **Background**)
Specifies the color to use for the background of the window. The default is white.

foreground (class **Foreground**) Specifies the color to use for displaying text in the window. Setting the class name instead of the instance name is an easy way to have everything that would usually be displayed in the text color to change color. The default is black.

cursorColor (class **Foreground**) Specifies the color to use for the text cursor. The default is black.

eightBitInput (class **EightBitInput**) If True, meta characters input from the keyboard are presented as a single character with the eighth bit turned on. If False, meta characters are converted into a 2-character sequence with the character itself preceded by **ESC**. The default is True.

eightBitOutput (class **EightBitOutput**) Specifies whether 8-bit characters sent from the host should be accepted as is or stripped when printed. The default is True.

font (class **Font**) Specifies the name of the normal font. The default is fixed.

font1 (class **Font1**) Specifies the name of the first alternative font.

font2 (class **Font2**) Specifies the name of the second alternative font.

font3 (class **Font3**) Specifies the name of the third alternative font.

font4 (class **Font4**) Specifies the name of the fourth alternative font.

font5 (class **Font5**) Specifies the name of the fifth alternative font.

font6 (class **Font6**) Specifies the name of the sixth alternative font.

geometry (class **Geometry**) Specifies the preferred size and position of the VT102 window.

hpLowerleftBugCompat (class **hpLowerleftBugCompat**) Specifies whether to work around a bug in **xdb**, which ignores termcap and always sends ESC F to move to the lower left corner. **true** causes **xterm** in interpret ESC F as a request to move to the lower left corner of the screen. The default is **false**.

internalBorder (class **BorderWidth**) Specifies the number of pixels between the characters and the window border. The default is 2.

jumpScroll (class **JumpScroll**) Specifies whether jump scrolling should be used. The default is True.

loginShell (class **LoginShell**) Specifies whether the shell to be run in the window should be started as a login shell. The default is False.

marginBell (class **MarginBell**) Specifies whether the bell should be run when the user types near the right margin. The default is False.

multiClickTime (class **MultiClickTime**) Specifies the maximum time in milliseconds between multiclick select events. The default is 250 milliseconds.

multiScroll (class **MultiScroll**) Specifies whether scrolling should be done asynchronously. The default is False.

nMarginBell (class **Column**) Specifies the number of characters from the right margin at which the margin bell should be rung, when enabled.

pointerColor (class **Foreground**) Specifies the foreground color of the pointer. The default is **XtDefaultForeground**.

pointerColorBackground (class **Background**) Specifies the background color of the pointer. The default is **XtDefaultBackground**.

pointerShape (class **Cursor**) Specifies the name of the shape of the pointer. The default is **xterm**.

resizeGravity (class **ResizeGravity**)
Affects the behavior when the window is resized to be taller or shorter. **NorthWest** specifies that the top line of text on the screen stays fixed. If the window is made shorter, lines are dropped from the bottom; if the window is made taller, blank lines are added at the bottom.

This is compatible with the behavior in MIT version X11R4. **SouthWest** (the default) specifies that the bottom line of text on the screen stays fixed. If the window is made taller, additional saved lines will be scrolled down onto the screen; if the window is made shorter, lines will be scrolled off the top of the screen, and the top saved lines will be dropped.

reverseVideo (class **ReverseVideo**)
Specifies whether reverse video should be simulated. The default is False.

reverseWrap (class **ReverseWrap**)
Specifies whether reverse wraparound should be enabled. The default is False.

saveLines (class **SaveLines**)
Specifies the number of lines to save beyond the top of the screen when a scrollbar is turned on. The default is 64.

scrollBar (class **ScrollBar**)
Specifies whether the scrollbar should be displayed. The default is False.

scrollTtyOutput (class **ScrollCond**)
Specifies whether output to the terminal should automatically cause the scrollbar to go to the bottom of the scrolling region. The default is True.

scrollKey (class **ScrollCond**)
Specifies whether pressing a key should automatically cause the scrollbar to go to the bottom of the scrolling region. The default is False.

scrollLines (class **ScrollLines**)
Specifies the number of lines that the **scroll-back** and **scroll-forw** actions should use as a default. The default value is 1.

signalInhibit (class **SignalInhibit**)
Specifies whether the entries in the Main Options menu for sending signals to **xterm** should be disallowed. The default is False.

tekGeometry (class **Geometry**)
Specifies the preferred size and position of the Tektronix window.

tekInhibit (class **TekInhibit**)
Specifies whether the escape sequence to enter Tektronix mode should be ignored. The default is False.

tekSmall (class **TekSmall**)
Specifies whether the Tektronix mode window should start in its smallest size if no explicit geometry is given. This is useful when running the **xterm** command on displays with small screens. The default is False.

tekStartup (class **TekStartup**)
Specifies whether **xterm** should start up in Tektronix mode. The default is False.

titeInhibit (class **TiteInhibit**)
Specifies whether **xterm** should remove **ti** and **te termcap** file entries (used to switch between alternate screens during startup of many screen-oriented programs) from the **TERMCAP** string. If set, the **xterm** command also ignores the escape sequence to switch to the alternate screen.

translations (class **Translations**)
Specifies the key and button bindings for menus, selections, programmed strings, and so forth. See ″Actions″ .

visualBell (class **VisualBell**)
Specifies whether a visible bell (flashing) should be used instead of an audible bell when the Ctrl+G key sequence signal is received. The default is False.

The following resources are specified as part of the **tek4014** widget (class **Tek4014**):

width (class **Width**)	Specifies the width of the Tektronix window in pixels.
height (class **Height**)	Specifies the height of the Tektronix window in pixels.
fontLarge (class **Font**)	Specifies the large font to use in the Tektronix window.
font2 (class **Font**)	Specifies font number 2 to use in the Tektronix window.
font3 (class **Font**)	Specifies font number 3 to use in the Tektronix window.
fontSmall (class **Font**)	Specifies the small font to use in the Tektronix window.
initialFont (class **InitialFont**)	Specifies which of the four Tektronix fonts to use initially. Values are the same as for the **set-tek-text** action. The default is large.
ginTerminator (class **GinTerminator**)	Specifies what characters should follow a GIN report or status report. The possibilities are `none,' which sends no terminating characters; CRonly, which sends CR; and CR&EOT, which sends both CR and EOT. The default is none.

The resources that may be specified for the various menus are described in the documentation for the **Athena SimpleMenu** widget. Following is a list of the names and classes of the entries in each of the menus.

The mainMenu has the following entries:

securekbd (class **SmeBSB**)	Invokes the **secure()** action.
allowsends (class **SmeBSB**)	Invokes the **allow-send-events(toggle)** action.
redraw (class **SmeBSB**)	Invokes the **redraw()** action.
line1 (class **SmeLine**)	This is a separator.
suspend (class **SmeBSB**)	Invokes the **send-signal(tstp)** action on systems that support job control.
continue (class **SmeBSB**)	Invokes the **send-signal(cont)** action on systems that support job control.
interrupt (class **SmeBSB**)	Invokes the **send-signal(int)** action.
hangup (class **SmeBSB**)	Invokes the **send-signal(hup)** action.
terminate (class **SmeBSB**)	Invokes the **send-signal(term)** action.
kill (class **SmeBSB**)	Invokes the **send-signal(kill)** action.
line2 (class **SmeLine**)	This is a separator.
quit (class **SmeBSB**)	Invokes the **quit()** action.

The vtMenu has the following entries:

scrollbar (class **SmeBSB**)	Invokes the **set-scrollbar(toggle)** action.
jumpscroll (class **SmeBSB**)	Invokes the **set-jumpscroll(toggle)** action.
reversevideo (class **SmeBSB**)	Invokes the **set-reverse-video(toggle)** action.
autowrap (class **SmeBSB**)	Invokes the **set-autowrap(toggle)** action.
reversewrap (class **SmeBSB**)	Invokes the **set-reversewrap(toggle)** action.
autolinefeed (class **SmeBSB**)	Invokes the **set-autolinefeed(toggle)** action.
appcursor (class **SmeBSB**)	Invokes the **set-appcursor(toggle)** action.
appkeypad (class **SmeBSB**)	Invokes the **set-appkeypad(toggle)** action.
scrollkey (class **SmeBSB**)	Invokes the **set-scroll-on-key(toggle)** action.
scrollttyoutput (class **SmeBSB**)	Invokes the **set-scroll-on-tty-output(toggle)** action.

allow132 (class **SmeBSB**) Invokes the **set-allow132(toggle)** action.
cursesemul (class **SmeBSB**) Invokes the **set-cursesemul(toggle)** action.
visualbell (class **SmeBSB**) Invokes the **set-visualbell(toggle)** action.
marginbell (class **SmeBSB**) Invokes the **set-marginbell(toggle)** action.
altscreen (class **SmeBSB**) This entry is currently disabled.
line1 (class **SmeLine**) This is a separator.
softreset (class **SmeBSB**) Invokes the **soft-reset()** action.
hardreset (class **SmeBSB**) Invokes the **hard-reset()** action.
clearsavedlines (class **SmeBSB**) Invokes the **clear-saved-lines()** action.
line2 (class **SmeLine**) This is a separator.
tekshow (class **SmeBSB**) Invokes the **set-visibility(tek,toggle)** action.
tekmode (class **SmeBSB**) Invokes the **set-terminal-type(tek)** action.
vthide (class **SmeBSB**) Invokes the **set-visibility(vt,off)** action.

The fontMenu has the following entries:

fontdefault (class **SmeBSB**) Invokes the **set-vt-font(d)** action.
font1 (class **SmeBSB**) Invokes the **set-vt-font(1)** action.
font2 (class **SmeBSB**) Invokes the **set-vt-font(2)** action.
font3 (class **SmeBSB**) Invokes the **set-vt-font(3)** action.
font4 (class **SmeBSB**) Invokes the **set-vt-font(4)** action.
font5 (class **SmeBSB**) Invokes the **set-vt-font(5)** action.
font6 (class **SmeBSB**) Invokes the **set-vt-font(6)** action.
fontescape (class **SmeBSB**) Invokes the **set-vt-font(e)** action.
fontsel (class **SmeBSB**) Invokes the **set-vt-font(s)** action.

The tekMenu has the following entries:

tektextlarge (class **SmeBSB**) Invokes the **set-tek-text(l)** action.
tektext2 (class **SmeBSB**) Invokes the **set-tek-text(2)** action.
tektext3 (class **SmeBSB**) Invokes the **set-tek-text(3)** action.
tektextsmall (class **SmeBSB**) Invokes the **set-tek-text(s)** action.
line1 (class **SmeLine**) This is a separator.
tekpage (class **SmeBSB**) Invokes the **tek-page()** action.
tekreset (class **SmeBSB**) Invokes the **tek-reset()** action.
tekcopy (class **SmeBSB**) Invokes the **tek-copy()** action.
line2 (class **SmeLine**) This is a separator.
vtshow (class **SmeBSB**) Invokes the **set-visibility(vt,toggle)** action.
vtmode (class **SmeBSB**) Invokes the **set-terminal-type(vt)** action.
tekhide (class **SmeBSB**) Invokes the **set-visibility(tek,toggle)** action.

The following resources are useful when specified for the **Athena Scrollbar** widget:

thickness (class **Thickness**) Specifies the width in pixels of the scrollbar.
background (class **Background**) Specifies the color to use for the background of the scrollbar.
foreground (class **Foreground**) Specifies the color to use for the foreground of the scrollbar. The *thumb* of the scrollbar is a simple checkerboard pattern with alternating pixels for foreground and background colors.

Pointer Usage

After the VT102 window is created, the **xterm** command allows you to select text and copy it within the same or other windows.

The selection functions are invoked when the pointer buttons are used with no modifiers, and when they are used with the Shift key. The assignment of the functions to keys and buttons may be changed through the resource database.

Pointer button 1 (usually left) is used to save text into the cut buffer. Move the cursor to beginning of the text, and then hold the button down while moving the cursor to the end of the region and releasing the button. The selected text is highlighted and is saved in the global cut buffer and made the PRIMARY selection when the button is released.

Double-clicking selects by words, triple-clicking selects by lines, and quadruple-clicking goes back to characters. Multiple-click is determined by the amount of time from button up to button down, so you can change the selection unit in the middle of a selection. If the key or button bindings specify that an X selection is to be made, the **xterm** command will leave the selected text highlighted for as long as it is the selection owner.

Pointer button 2 (usually middle) ″types″ (pastes) the text from the PRIMARY selection, if any, otherwise from the cut buffer, inserting it as keyboard input.

Pointer button 3 (usually right) extends the current selection. If pressed while closer to the right edge of the selection than the left, it extends or contracts the right edge of the selection. If you contract the selection past the left edge of the selection, the **xterm** command assumes you really meant the left edge, restores the original selection, and then extends or contracts the left edge of the selection.

And the opposite also applies: if pressed while closer to the left edge of the selection than the right, it extends/contracts the left edge of the selection. If you contract the selection past the right edge of the selection, the **xterm** command assumes you really meant the right edge, restores the original selection, and then extends/contracts the right edge of the selection. Extension starts in the selection unit mode that the last selection or extension was performed in; you can multiple-click to cycle through them.

By cutting and pasting pieces of text without trailing new lines, you can take text from several places in different windows and form a command to the shell, for example, or take output from a program and insert it into your favorite editor. Because the cut buffer is globally shared among different applications, regard it as a “file” whose contents you know. The terminal emulator and other text programs should be treating it as if it were a text file; in other words, the text is delimited by new lines.

The scroll region displays the position and amount of text currently showing in the window (highlighted) relative to the amount of text actually saved. As more text is saved (up to the maximum), the size of the highlighted area decreases.

Clicking button 1 with the pointer in the scroll region moves the adjacent line to the top of the display window.

Clicking button 3 moves the top line of the display window down to the pointer position.

Clicking button 2 moves the display to a position in the saved text that corresponds to the pointer’s position in the scrollbar.

Unlike the VT102 window, the Tektronix window does not allow the copying of text. It does allow Tektronix GIN mode, and in this mode the cursor will change from an arrow to a cross. Pressing any key will send that key and the current coordinates of the cross cursor. Pressing button one, two, or three will return the letters l, m, and r, respectively.

If the Shift key is pressed when a pointer button is pressed, the corresponding uppercase letter is sent. To distinguish a pointer button from a key, the high bit of the character is set (but this bit is usually stripped unless the terminal mode is RAW; see the **tty** command for details).

Menus

The **xterm** command has four menus, named mainMenu, vtMenu, fontMenu, and tekMenu. Each menu opens under the correct combinations of key and button presses. Most menus are divided into two section, separated by a horizontal line. The top portion contains various modes that can be altered. A check mark is displayed next to a mode that is currently active. Selecting one of these modes toggles its state. The bottom portion of the menu lists command entries; selecting one of these performs the indicated function.

The xterm menu opens when the control key and pointer button one are pressed in a window. The mainMenu contains items that apply to both the VT102 and Tektronix windows. The **Secure Keyboard** mode is used when typing in passwords or other sensitive data in an unsecure environment.

Notable entries in the command section of the menu are **Continue**, **Suspend**, **Interrupt**, **Hangup**, **Terminate**, and **Kill**, which send the **SIGCONT**, **SIGTSTP**, **SIGINT**, **SIGHUP**, **SIGTERM**, and **SIGKILL** signals, respectively, to the process group of the process running under **xterm** (usually the shell). The **Continue** function is especially useful if the user has accidentally pressed Ctrl+Z, suspending the process.

The vtMenu sets various modes in the VT102 emulation, and is opened when the control key and pointer button two are pressed in the VT102 window. In the command section of this menu, the soft reset entry will reset scroll regions. This can be convenient when some program has left the scroll regions set incorrectly (often a problem when using VMS or TOPS-20).

The full reset entry will clear the screen, reset tabs to every eight columns, and reset the terminal modes (such as wrap and smooth scroll) to their initial states just after the **xterm** command has finished processing the command-line options.

The fontMenu sets the font used in the VT102 window. In addition to the default font and a number of alternatives that are set with resources, the menu offers the font last specified by the Set Font escape sequence (See ″ Control Sequences″) and the current selection as a font name (if the PRIMARY selection is owned).

The tekMenu sets various modes in the Tektronix emulation, and is opened when the control key and pointer button two are pressed in the Tektronix window. The current font size is checked in the Modes section of the menu. The **PAGE** entry in the command section clears the Tektronix window.

Security

X windows environments differ in their security consciousness. MIT servers, run under **xdm**, are capable of using a *magic cookie* authorization scheme that can provide a reasonable level of security for many people. If your server is only using a host-based mechanism to control access to the server (see the **xhost** command), and if you enable access for a host and other users are also permitted to run clients on that same host, there is every possibility that someone can run an application that will use the basic services of the X protocol to snoop on your activities, potentially capturing a transcript of everything you type at the keyboard.

This is of particular concern when you want to type in a password or other sensitive data. The best solution to this problem is to use a better authorization mechanism than host-based control, but a simple mechanism exists for protecting keyboard input in the **xterm** command.

The xterm menu contains a **Secure Keyboard** entry that, when enabled, ensures that all keyboard input is directed *only* to the **xterm** command (using the **GrabKeyboard** protocol request). When an application prompts you for a password (or other sensitive data), you can enable **Secure Keyboard** using the menu, type in the data, and then disable **Secure Keyboard** using the menu again.

Only one X client at a time can secure the keyboard, so when you attempt to enable **Secure Keyboard** it may fail. In this case, the bell will sound. If the **Secure Keyboard** succeeds, the foreground and

background colors will be exchanged (as if you selected the **Reverse Video** entry in the Modes menu); they will be exchanged again when you exit secure mode. If the colors do *not* switch, be *very* suspicious that you are being spoofed.

If the application you are running displays a prompt before asking for the password, it is safest to enter secure mode *before* the prompt gets displayed, and to make sure that the prompt gets displayed correctly (in the new colors), to minimize the probability of spoofing. You can also bring up the menu again and make sure that a check mark is displayed next to the entry.

Secure Keyboard mode will be disabled automatically if your xterm window becomes iconified (or otherwise unmapped), or if you start up a reparenting window manager (that places a title bar or other decoration around the window) while in **Secure Keyboard** mode. (This is a feature of the X protocol not easily overcome.) When this happens, the foreground and background colors will be switched back and the bell will sound in warning.

Character Classes

Clicking the middle mouse button twice in rapid succession will cause all characters of the same class (such as letters, white space, punctuation) to be selected. Because different people have different preferences for what should be selected (for example, should file names be selected as a whole or only the separate subnames), the default mapping can be overridden through the use of the **charClass** (class **CharClass**) resource.

This resource is a series of comma-separated *range*:*value* pairs. The *range* is either a single number or *low-high* in the range of 0 to 127, corresponding to the ASCII code for the character or characters to be set. The *value* is arbitrary, although the default table uses the character number of the first character occurring in the set.

The default table is:

```
static int charClass[128] = {
/* NUL  SOH  STX  ETX  EOT  ENQ  ACK  BEL */
    32,   1,   1,   1,   1,   1,   1,   1,
/*  BS   HT   NL   VT   NP   CR   SO   SI */
     1,  32,   1,   1,   1,   1,   1,   1,
/* DLE  DC1  DC2  DC3  DC4  NAK  SYN  ETB */
     1,   1,   1,   1,   1,   1,   1,   1,
/* CAN   EM  SUB  ESC   FS   GS   RS   US */
     1,   1,   1,   1,   1,   1,   1,   1,
/*  SP    !    "    #    $    %    &    ' */
    32,  33,  34,  35,  36,  37,  38,  39,
/*   (    )    *    +    ,    -    .    / */
    40,  41,  42,  43,  44,  45,  46,  47,
/*   0    1    2    3    4    5    6    7 */
    48,  48,  48,  48,  48,  48,  48,  48,
/*   8    9    :    ;    <    =    >    ? */
    48,  48,  58,  59,  60,  61,  62,  63,
/*   @    A    B    C    D    E    F    G */
    64,  48,  48,  48,  48,  48,  48,  48,
/*   H    I    J    K    L    M    N    O */
    48,  48,  48,  48,  48,  48,  48,  48,
/*   P    Q    R    S    T    U    V    W */
    48,  48,  48,  48,  48,  48,  48,  48,
/*   X    Y    Z    [    \    ]    ^    _ */
    48,  48,  48,  91,  92,  93,  94,  48,
/*   `    a    b    c    d    e    f    g */
    96,  48,  48,  48,  48,  48,  48,  48,
/*   h    i    j    k    l    m    n    o */
    48,  48,  48,  48,  48,  48,  48,  48,
/*   p    q    r    s    t    u    v    w */
    48,  48,  48,  48,  48,  48,  48,  48,
/*   x    y    z    {    |    }    ~  DEL */
    48,  48,  48, 123, 124, 125, 126,   1};
```

For example, the string `33:48,37:48,45-47:48,64:48` indicates that the exclamation mark, percent sign, dash, period, slash, and & characters should be treated the same way as characters and numbers. This is useful for cutting and pasting electronic mailing addresses and file names.

Actions

It is possible to rebind keys (or sequences of keys) to arbitrary strings for input by changing the translations for the **vt100** or **tek4014** widgets. Changing the translations for events other than key and button events is not expected, and will cause unpredictable behavior. The following actions are provided for using within the vt100 or tek4014 translations resources:

bell([*Percent*]**)**	Rings the keyboard bell at the specified percentage above or below the base volume.
ignore()	Ignores the event but checks for special pointer position escape sequences.
insert()	Inserts the character or string associated with the key that was pressed.
insert-seven-bit()	Is a synonym for **insert()**.
insert-eight-bit()	Inserts an 8-bit (meta) version of the character or string associated with the key that was pressed. The exact action depends on the value of the **eightBitInput** resource.
insert-selection(*SourceName* [**,** ...]**)**	Inserts the string found in the selection or cutbuffer indicated by the *SourceName* parameter. Sources are checked in the order given (case is significant) until one is found. Commonly used selections include PRIMARY, SECONDARY, and CLIPBOARD. Cut buffers are typically named CUT_BUFFER0 through CUT_BUFFER7.
keymap(*Name***)**	Dynamically defines a new translation table whose resource name is *Name* with the suffix *Keymap* (case is significant). The name None restores the original translation table.
popup-menu(*MenuName***)**	Displays the specified popup menu. Valid names (case is significant) include mainMenu, vtMenu, fontMenu, and tekMenu.
secure()	Toggles the **Secure Keyboard** mode described in the section named ″ Security″ , and is invoked from the **securekbd** entry in mainMenu.
select-start()	Begins text selection at the current pointer location. See the section entitled ″ Pointer Usage″ for information on making selections.
select-extend()	Tracks the pointer and extends the selection. Only bind this to Motion events.
select-end(*DestName* [**,** ...]**)**	Puts the currently selected text into all of the selections or cutbuffers specified by *DestName*.
select-cursor-start()	Is similar to **select-start** except that it begins the selection at the current text cursor position.
select-cursor-end(*DestName* [**,** ...]**)**	Is similar to **select-end** except that it should be used with **select-cursor-start**.

set-vt-font(*d/1/2/3/4/5/6/e/s* [**,***NormalFont* [**,** *BoldFont*]]**)**
Sets the font or fonts currently being used in the VT102 window. The first argument is a single character that specifies the font to be used:

d or *D* indicates the default font (the font initially used when the **xterm** command was started),

1 through *6* indicate the fonts specified by the *font1* through *font6* resources,

e or *E* indicates the normal and bold fonts that have been set through escape codes (or specified as the second and third action arguments, respectively), and

s or *S* indicates the font selection (as made by programs such as the **xfontsel** program) specified by the second action argument.

start-extend()
Is similar to **select-start** except that the selection is extended to the current pointer location.

start-cursor-extend()
Is similar to **select-extend** except that the selection is extended to the current text cursor position.

string(*String***)**
Inserts the specified text string as if it had been typed. Quotation is necessary if the string contains white space or nonalphanumeric characters. If the string argument begins with the characters ``0x,'' it is interpreted as a hex character constant.

scroll-back(*Count* [**,***Units*]**)**
Scrolls the text window backward so that text that had previously scrolled off the top of the screen is now visible. The *Count* argument indicates the number of *Units* (which may be *page*, *halfpage*, *pixel*, or *line*) by which to scroll.

scroll-forw(*Count* [**,***Units*]**)**
Scrolls is similar to **scroll-back** except that it scrolls the other direction.

allow-send-events(*On/Off/Toggle***)**
Sets or toggles the **allowSendEvents** resource and is also invoked by the **allowsends** entry in mainMenu.

redraw()
Redraws the window and is also invoked by the **redraw** entry in mainMenu.

send-signal(*SigName***)**
Sends the signal named by *SigName* to the **xterm** subprocess (the shell or program specified with the **-e** command-line option) and is also invoked by the **suspend**, **continue**, **interrupt**, **hangup**, **terminate**, and **kill** entries in mainMenu. Allowable signal names are (case is not significant):

tstp (if supported by the operating system),

suspend (same as tstp),

cont (if supported by the operating system),

int,

hup,

term,

quit,

alrm,

alarm (same as alrm), and

kill.

quit()
Sends a **SIGHUP** to the subprogram and exits. It is also invoked by the **quit** entry in mainMenu.

set-scrollbar(*On/Off/Toggle***)**	Toggles the **scrollbar** resource and is also invoked by the **scrollbar** entry in vtMenu.
set-jumpscroll(*On/Off/Toggle***)**	Toggles the **jumpscroll** resource and is also invoked by the **jumpscroll** entry in vtMenu.
set-reverse-video(*On/Off/Toggle***)**	Toggles the **reverseVideo** resource and is also invoked by the **reversevideo** entry in vtMenu.
set-autowrap(*On/Off/Toggle***)**	Toggles automatic wrapping of long lines and is also invoked by the **autowrap** entry in vtMenu.
set-reversewrap(*On/Off/Toggle***)**	Toggles the **reverseWrap** resource and is also invoked by the **reversewrap** entry in vtMenu.
set-autolinefeed(*On/Off/Toggle***)**	Toggles automatic insertion of linefeeds and is also invoked by the **autolinefeed** entry in vtMenu.
set-appcursor(*On/Off/Toggle***)**	Toggles the handling Application Cursor Key mode and is also invoked by the **appcursor** entry in vtMenu.
set-appkeypad(*On/Off/Toggle***)**	Toggles the handling of Application Keypad mode and is also invoked by the **appkeypad** entry in vtMenu.
set-scroll-on-key(*On/Off/Toggle***)**	Toggles the **scrollKey** resource and is also invoked from the **scrollkey** entry in vtMenu.
set-scroll-on-tty-output(*On/Off/Toggle***)**	Toggles the **scrollTtyOutput** resource and is also invoked from the **scrollttyoutput** entry in vtMenu.
set-allow132(*On/Off/Toggle***)**	Toggles the **c132** resource and is also invoked from the **allow132** entry in vtMenu.
set-cursesemul(*On/Off/Toggle***)**	Toggles the **curses** resource and is also invoked from the **cursesemul** entry in vtMenu.
set-visual-bell(*On/Off/Toggle***)**	Toggles the **visualBell** resource and is also invoked by the **visualbell** entry in vtMenu.
set-marginbell(*On/Off/Toggle***)**	Toggles the **marginBell** resource and is also invoked from the **marginbel**l entry in vtMenu.
set-altscreen(*On/Off/Toggle***)**	Toggles between the alternate and current screens.
soft-reset()	Resets the scrolling region and is also invoked from the **softreset** entry in vtMenu.
hard-reset()	Resets the scrolling region, tabs, window size, and cursor keys and clears the screen. It is also invoked from the **hardreset** entry in vtMenu.
clear-saved-lines()	Performs **hard-reset** (see previous entry) and also clears the history of lines saved off the top of the screen. It is also invoked from the **clearsavedlines** entry in vtMenu.
set-terminal-type(*Type***)**	Directs output to either the vt or tek windows, according to the *Type* string. It is also invoked by the **tekmode** entry in vtMenu and the **vtmode** entry in tekMenu.
set-visibility(*vt/tek*, *On/Off/Toggle***)**	Controls whether or not the vt or tek windows are visible. It is also invoked from the **tekshow** and **vthide** entries in vtMenu and the **vtshow** and **tekhide** entries in tekMenu.
set-tek-text(*large/2/3/small***)**	Sets font used in the Tektronix window to the value of the resources **tektextlarge**, **tektext2**, **tektext3**, and **tektextsmall** according to the argument. It is also by the entries of the same names as the resources in tekMenu.
tek-page()	Clears the Tektronix window and is also invoked by the **tekpage** entry in tekMenu.
tek-reset()	Resets the Tektronix window and is also invoked by the **tekreset** entry in tekMenu.
tek-copy()	Copies the escape codes used to generate the current window contents to a file in the current directory beginning with the name **COPY**. It is also invoked from the **tekcopy** entry in tekMenu.
visual-bell()	Flashes the window quickly.

The Tektronix window also has the following action:

gin-press(*l/L/m/M/r/R***)** Sends the indicated graphics input code.

The default bindings in the VT102 window are:

```
Shift <KeyPress> Prior:     scroll-back(1,halfpage) \n\
Shift <KeyPress> Next:      scroll-forw(1,halfpage) \n\
Shift <KeyPress> Select:    select-cursor-start \
                            select-cursor-end(PRIMARY,
                            CUT_BUFFER0) \n\
Shift <KeyPress> Insert:    insert-selection(PRIMARY,
                            CUT_BUFFER0) \n\
~Meta<KeyPress>:            insert-seven-bit \n\
Meta<KeyPress>:             insert-eight-bit \n\
!Ctrl <Btn1Down>:           popup-menu(mainMenu) \n\
!Lock Ctrl <Btn1Down>:      popup-menu(mainMenu) \n\
~Meta <Btn1Down>:           select-start \n\
~Meta <Btn1Motion>:         select-extend \n\
!Ctrl <Btn2Down>:           popup-menu(vtMenu) \n\
!Lock Ctrl <Btn2Down>:      popup-menu(vtMenu) \n\
~Ctrl ~Meta <Btn2Down>:     ignore \n\
~Ctrl ~Meta <Btn2Up>:       insert-selection(PRIMARY,
                            CUT_BUFFER0) \n\
!Ctrl <Btn3Down>:           popup-menu(fontMenu) \n\
!Lock Ctrl <Btn3Down>:      popup-menu(fontMenu) \n\
~Ctrl ~Meta <Btn3Down>:     start-extend \n\
~Meta <Btn3Motion>:         select-extend \n\
<BtnUp>:                    select-end(PRIMARY, CUT_BUFFER0) \n\
<BtnDown>:                  bell(0)
```

The default bindings in the Tektronix window are:

```
~Meta<KeyPress>:            insert-seven-bit \n\
Meta<KeyPress>:             insert-eight-bit \n\
!Ctrl <Btn1Down>:           popup-menu(mainMenu) \n\
!Lock Ctrl <Btn1Down>:      popup-menu(mainMenu) \n\
!Ctrl <Btn2Down>:           popup-menu(tekMenu) \n\
!Lock Ctrl <Btn2Down>:      popup-menu(tekMenu) \n\
Shift ~Meta<Btn1Down>:      gin-press(L) \n\
~Meta<Btn1Down>:            gin-press(l) \n\
Shift ~Meta<Btn2Down>:      gin-press(M) \n\
~Meta<Btn2Down>:            gin-press(m) \n\
Shift ~Meta<Btn3Down>:      gin-press(R) \n\
~Meta<Btn3Down>:            gin-press(r)
```

The following is an example of how the **keymap** action is used to add special keys for entering commonly typed works:

```
*VT100.Translations:          #override <Key>F13: keymap(dbx)
*VT100.dbxKeymap.translations:
 \

    <Key>F14:    keymap(None) \n\
    <Key>F17:    string("next") string(0x0d) \n\
    <Key>F18:    string("step") string(0x0d) \n\
    <Key>F19:    string("continue") string(0x0d) \n\
    <Key>F20:    string("print ")
                 insert-selection(PRIMARY,CUT_BUFFER0)
```

Environment

The **xterm** command sets the environment variables **TERM** and **TERMCAP** properly for the size window you have created. It also uses and sets the **DISPLAY** environment variable to specify which bitmap display terminal to use. The **WINDOWID** environment variable is set to the X window ID number of the xterm window.

Bugs

Large pastes do not work on some systems. This is not a bug in the **xterm** command; it is a bug in the pseudo terminal driver of those systems. The **xterm** command feeds large pastes to the pty only as fast as the pty will accept data, but some pty drivers do not return enough information to know if the write operation has succeeded.

Many of the options are not resettable after the **xterm** command starts.

Only fixed-width, character-cell fonts are supported.

Control Sequences

This section lists control sequences available for the **xterm** command.

Definitions

The following information shows how to interpret key sequences in this section.

c	The literal characters *c*.
C	A single (required) character.
P_s	A single (usually optional) numeric parameter, composed of one or more digits.
P_m	A multiple numeric parameter composed of any number of single numeric parameters, seperated by a **;** (semi-colon) character or characters.
P_t	A text parameter composed of printable characters.

VT100 Mode

Most of these control sequences are standard VT102 control sequences, but there are some sequences here from later DEC VT terminals, too. Major VT102 features not supported are smooth scrolling, double-size characters, blinking characters, and VT52 mode.

There are additional control sequences to provide xterm-dependent functions, like the scrollbar or window size. Where the function is specified by DEC or ISO 6429, the code assigned to it is given in parentheses. The escape codes to designate character sets are specified by ISO 2022; see that document for a discussion of character sets.

Control Sequence	Description
BEL	Bell (Ctrl+G)
BS	Backspace (Ctrl+H)
TAB	Horizontal Tab (HT) (Ctrl+I)
LF	Line Feed or New Line (NL) (Ctrl+J)
VT	Vertical Tab (Ctrl+K) same as LF
FF	Form Feed or New Page (NP) (Ctrl+L) same as LF
CR	Carriage return (Ctrl+M)
SO	Shift Out (Ctrl+N) –> Switch to ALternate Character Set: Invokes the G1 character set.
SI	Shift In (Ctrl+O) –> Switch to Standard Character Set: Invokes the G0 character set (the default).
ESC # 8	DEC Screen Test (DCECALN)

Control Sequence	Description
ESC (*C*	Designate G0 Character Set (ISO 2022) ***C* = 0** DEC Special Character and Line Drawing Set ***C* = A** United Kingdom (UK) ***C* = B** United States (USASCII)
ESC) *C*	Designate G1 Character Set (ISO 2022) ***C* = 0** DEC Special Character and Line Drawing Set ***C* = A** United Kingdom (UK) ***C* = B** United States (USASCII)
ESC * *C*	Designate G2 Character Set (ISO 2022) ***C* = 0** DEC Special Character and Line Drawing Set ***C* = A** United Kingdom (UK) ***C* = B** United States (USASCII)
ESC + *C*	Designate G3 Character Set (ISO 2022) ***C* = 0** DEC Special Character and Line Drawing Set ***C* = A** United Kingdom (UK) ***C* = B** United States (USASCII)
ESC 7	Save Cursor (DECSC)
ESC 8	Restore Cursor (DECRC)
ESC =	Application Keypad (DECPAM)
ESC >	Normal Keypad (DECNM)
ESC D	Index (IND)
ESC E	Next Line (NEL)
ESC H	Tab Set (HTS)
ESC M	Reverse Index (RI)
ESC N	Single Shift Select of G2 Character Set (SS2): Affects next character only.
ESC P	Single Shift Select of G3 Character Set (SS2): Affects next character only.
**ESC O *P_t* ESC **	Device Control String (DCS). xterm implements no DCS functions; ***P_t*** is ignored. ***P_t*** need not be printable characters.
ESC Z	Return Terminal ID (DECID). Obsolete form of **ESC [*c*** (DA)
ESC [*P_s* @	Insert ***P_s*** (Blank) Character of Characters (default=1) (ICH)
ESC [*P_s* A	Cursor Up ***P_s*** Times (default=1) (CUU)
ESC [*P_s* B	Cursor Down ***P_s*** Times (default=1) (CUD)
ESC [*P_s* C	Cursor Forward ***P_s*** Times (default=1) (CUF)
ESC [*P_s* D	Cursor Backward ***P_s*** Times (default=1) (CUB)
ESC [*P_s* ; *P_s* H	Cursor Position [row;column] (default=1) (CUP)

ESC [P_s J	Erase in Display (ED) **P_s = 0** Clear Below (Default) **P_s = 1** Clear Above **P_s = 2** Clear All
ESC [P_s K	Erase in Line (EL) **P_s = 0** Clear to Right (Default) **P_s = 1** Clear to Left **P_s = 2** Clear All
ESC [P_s L	Insert **P_s** Lines (default=1) (IL)
ESC [P_s M	Delete **P_s** Lines (default=1) (DL)
ESC [P_s P	Delete **P_s** Characters (default=1) (DCH)
ESC [P_s ; P_s ; P_s ; P_s ; P_s T	Initiate hilite mouse tracking. Parameters are [*Func*;*Startx*;Starty;*FirstRow*;*LastRow*]. See “Mouse Tracking” on page 240.
ESC [P_s c	SendDevice Attributes (DA)Delete **P_s** Characters (default=1) (DCH) **P_s =0 or omitted** Request attribute from terminal **ESC [? 1 ; 2 c** (″I am a VT100 with Advanced Video Option.″)
ESC [P_s ; P_s f	Horizontal and Vertical Position [row;column] (default = [1,1]) (HVP)
ESC [P_s g	Tab Clear (TBC) **P_s = 0** Clear Current Column (default) **P_s = 3** Clear All
ESC [P_m h	Set Modes (SM) **P_s = 4** Insert Mode (IRM) **P_s = 2 0** Automatic Newline (LNM)
ESC [P_m l	Reset Modes (RM) **P_s = 4** Replace Mode (IRM) **P_s = 2 0** Normal Linefeed (LNM)
ESC [P_m m	Character Attributes (SGR) **P_s = 0** Nomal (default) **P_s = 1** Bold **P_s = 4** Underscore **P_s = 5** Blink (displayed as Bold) **P_s = 7** Inverse

ESC [***Ps*** **n**	Device Status Report (DSR) ***Ps*** **= 5** Status Report **ESC [0 n** (″OK″) ***Ps*** **= 6** Report Cursor Position (CPR) [row;column] as **ESC [r ;** *c* **R** ***Ps*** **= 2 0** Automatic Newline (LNM)
ESC [***Ps*** **;** ***Ps*** **r**	Set Scroll Region [top;bottom] (default = fullsize of window) (DECSTBM)
ESC [***Ps*** **x**	Request Terminal Parameters (DECREQTPARM)
ESC [? ***Pm*** **h**	DEC Private Mode (DECSET) ***Ps*** **= 1** Application Cursor Keys (DECCKM) ***Ps*** **= 2** Designate USASCII for character sets G0–G3. (In VT102, this selects VT52 mode (DECANM), which **xterm** does not support.) ***Ps*** **= 3** 132 Column Mode (DECCOLM) ***Ps*** **= 4** Smooth (Slow) Scroll (DECSCLM) ***Ps*** **= 5** Reverse Video (DECSCNM) ***Ps*** **= 6** Origin Mode (DECOM) ***Ps*** **= 7** Wraparound Mode (DECAWM) ***Ps*** **= 8** Auto-repeat Keys (DECARM) ***Ps*** **= 9** Set Mouse X and Y on button press. See “Mouse Tracking” on page 240. ***Ps*** **= 3 8** Enter Tektronix Mode (DECTEK) ***Ps*** **= 4 0** Allow 80 <—> 132 Mode ***Ps*** **= 4 1** **curses** function fix ***Ps*** **= 4 4** Turn On Margin Bell ***Ps*** **= 4 5** Reverse Wraparound Mode ***Ps*** **= 4 7** Use Alternate Screen Buffer (unless disabled by **titeInhibit** resource) ***Ps*** **= 1 0 0 0** Set Mouse X and Y on button press and release. See “Mouse Tracking” on page 240. ***Ps*** **= 1 0 0 1** Use Hilite Mouse Tracking.

ESC [? P_m l	DEC Private Mode Reset (DECRST) **P_s = 1** Normal Cursor Keys (DECCKM) **P_s = 3** 80 Column Mode (DECCOLM) **P_s = 4** Jump Fast Scroll (DECSCLM) **P_s = 5** Normal Video (DECSCNM) **P_s = 6** Normal Cursor Mode (DECOM) **P_s = 7** No Wraparound Mode (DECAWM) **P_s = 8** No Auto-repeat Keys (DECARM) **P_s = 9** Do not Send Mouse X and Y on button press. **P_s = 4 0** Disallow 80 <—> 132 Mode **P_s = 4 1** No **curses** function fix **P_s = 4 4** Turn Off Margin Bell **P_s = 4 5** No Reverse Wraparound Mode **P_s = 4 7** Use Normal Screen Buffer **P_s = 1 0 0 0** Do not Send Mouse X and Y on button press and release. **P_s = 1 0 0 1** Do not Use Hilite Mouse Tracking. xxx
ESC [? P_m r	Restore DEC Private Mode Values. The value of P_s previously saved is restored. P_s values are the same as DECSET.
ESC [? P_m s	Save DEC Private Mode Values. P_s values are the same as DECSET.
ESC]? P_s ; P_t BEL	Set Text Parameters **P_s = 0** Change Icon Name and Window Title to P_t **P_s = 1** Change Icon Name to P_t **P_s = 2** Change Window Title to P_t **P_s = 5 0** Set Font to P_t
**ESC P_t ESC **	Private Mesage (PM). **xterm** implements no PM functions; P_t need not be printable characters.
**ESC _ P_t ESC **	Application Program Command (APC). Private Mesage (PM). **xterm** implements no APC functions; P_t is ignored. $P_t$need not be printable characters.
ESC c	Full Reset (RIS)
ESC n	Select the G2 Character Set (LS2)
ESC o	Select the G3 Character Set (LS3)
ESC l	Invoke the G3 Character Set as GR (LS3R). Has no visible effect in **xterm**.

ESC }	Invoke the G2 Character Set as GR (LS2R). Has no visible effect in **xterm**.
ESC	Invoke the G1 Character Set as GR (LS1R). Has no visible effect in **xterm**.

XTERM Description Limitation

The xterm terminal description in the DEC.TI file on AIX Version 4 provides underline mode by using the SGR attribute. The SMUL and RMUL attributes are not currently defined in the XTERM terminal description on AIX Version 4. Use the more generic capability named SGR.

```
tput sgr x y
```

Where *x* is either a 1 or a 0 to turn standout mode on or off respectively, and *y* is either a 1 or a 0 to turn underline mode on or off respectively. See the article ″**terminfo** file format″ for more details on the SGR capability.

```
tput sgr 0 1    turn off standout; turn on underline
tput sgr 0 0    turn off standout; turn off underline
tput sgr 1 1    turn on standout; turn on underline
tput sgr 1 0    turn on standout; turn off underline
```

Mouse Tracking

The **VT** widget can be set to send the mouse position and other information on button presses. These modes are typically used by editors and other full-screen applications that want to make use of the mouse.

There are three mutually exclusive modes, each enabled (or disabled) by a different parameter in the DECSET (or DECRST) escape sequence. Parameters for all mouse tracking escape sequences generated by the **xterm** command encode numeric parameters in a single character as *value*+040. The screen coordinate system is 1-based.

For example **!** is 1. The screen screen coordinate system is 1-based.

X10 compatibility mode sends an escape sequence on button press encoding the location and the mouse button pressed. It is enabled by specifying parameter 9 to DECSET. On button press, the **xterm** command sends the following ″6 characters″ . *C*b is button-1. *C*x and *C*y are the *x* and *y* coordinates of the mouse when the button was pressed.

ESC [M $C_bC_xC_y$

Normal tracking mode sends an escape sequence on both button press and release. Modifier information is also sent. It is enabled by specifying parameter 1000 to DECSET. On button press or release, the **xterm** command sends the following ″key sequence″ :

ESC [M $C_bC_xC_y$

The low two bits of C_b encode button information: 0=MB1 pressed, 1=MB2 pressed, 2=MB3 pressed, 3=release. The upper bits encode what modifiers were down when the button was pressed and are added together. 4=Shift, 8=Meta, 16=Control. C_x and C_y are the *x* and *y* coordinates of the mouse event. The upper left corner is (1,1).

Mouse hilite tracking notifies a program of a button press, receives a range of lines from the program, highlights the region covered by the mouse within that range until button release, and then sends the program the release coordinates. It is enabled by specifying parameter 1001 to DECSET.

Attention: Use of this mode requires a cooperating program or it will hang the **xterm** command. On button press, the same information as for normal tracking is generated; the **xterm** command then waits for the program to send mouse tracking information. *All X events are ignored until the following proper escape sequence is received from the pty*:

ESC [P_s **;** P_s **;** P_s **;** P_s **; T**

The parameters are *Func*, *Startx*, *Starty*, *FirstRow*, and *LastRow*. The *Func* parameter is nonzero to initiate hilite tracking and 0 (zero) to abort. The *Startx* and *Starty* parameters give the starting x and y location for the highlighted region. The ending location tracks the mouse, but is never above row *FirstRow* and is always above row *LastRow*. (The top of the screen is row 1.) When the button is released, the **xterm** command reports the ending position one of two ways: if the start and end coordinates are valid text locations, the **xterm** command reports the ″ending position″ as follows:

ESC [t C_xC_y

If either coordinate is past the end of the line, the **xterm** command reports the “ending position” as follows:

ESC [T $C_xC_yC_xC_yC_xC_y$

The parameters are *Startx*, *Starty*, *Endx*, *Endy*, *Mousex*, and *Mousey*. The *Startx*, *Starty*, *Endx*, and *Endy* parameters give the starting and ending character positions of the region. The *Mousex* and *Mousey* parameters give the location of the mouse at button up, which might not be over a character.

Tektronix 4014 Mode

Most of these sequences are standard Tektronix 4014 control sequences. The major features missing are the write-thru and defocused modes. This document does not describe the commands used in the various Tektronix plotting modes but does describe the commands to switch modes.

Related Information

The **aixterm** command, **resize** command, **tset** command, **vi** or **vedit** command.

---

xwd Command

Purpose

Dumps the image of an Enhanced X-Windows window.

Syntax

xwd [**-add** *Value*] [**-frame**] [**-display** *Display*] [**-help**] [**-nobdrs**] [**-xy**] [**-out** *File*] [**-root** | **-id** *id* | **-name** *Name*] [**-icmap**] [**-screen**]

Description

The **xwd** command is an Enhanced X-Windows window dumping utility. The **xwd** command allows you to store window images in a specially formatted dump file. This file can then be read by various other X utilities that perform functions such as redisplaying, printing, editing, formatting, archiving, and image processing. Select the target window by clicking the mouse in the desired window. The keyboard bell rings once at the beginning of the dump and twice when the dump is completed.

Flags

-add *Value*	Specifies a signed value to add to every pixel. This option is specific to X11R5.
-frame	This option indicates that the window manager frame should be included when manually selecting a window.

-display *Display*	Specifies the server connection.
-help	Prints the usage command syntax summary.
-nobdrs	Specifies that the window dump does not include the pixels that compose the X window border. This is useful if you want to include the window contents in a document as an illustration. The result of the **-nobdrs** flag depends on which window manager is running. Many window managers remove all borders from the client. For example, the **XGetWindowAttributes** function returns the value of 0 for the `border_width` field regardless of the border width when the client was started. Therefore, any border that is visible on the screen belongs to the window manager; the client has no knowledge of it. In this case, the **-nobdrs** flag has no effect.
-out *File*	Specifies the output file on the command line. The default is to output to standard out.
-root	Indicates that the root window should be selected for the window dump, without requiring the user to select a window with the pointer. This option is specific to X11R5.
-id *id*	Indicates that the window with the specified resource id should be selected for the window dump, without requiring the user to select a window with the pointer. This option is specific to X11R5.
-name *Name*	Indicates that the window with the specified WM_NAME property should be selected for the window dump, without requiring the user to select a window with the pointer. This option is specific to X11R5.
-icmap	Forces the first installed colormap of the screen to be used to obtain RGB values. By default, the colormap of the chosen window is used. This option is specific to X11R5.
-screen	Indicates that the GetImage request used to obtain the image should be done on the root window, rather than directly on the specified window. In this way, you can obtain pieces of the other windows that overlap the specified window and, more importantly, capture menus or other popups that are independent windows but appear over the specified window. This option is specific to X11R5.
-xy	Selects xy format dumping instead of the default z format. This option applies to color displays only.

File

XWDFile.h	X Window dump file format definition file.

Related Information

The **xwud** command.

---

xwud Command

Purpose

Retrieves and displays the dumped image of an Enhanced X-Windows window.

Syntax

xwud [**-in** *FileName*] [**-noclick**] [**-geometry** *Geometry*] [**-display** *Display*] [**-new**] [**-std** *MapType*] [**-raw**] [**-vis** **visual_type** | **visual_id**] [**-help**] [**-rv**] [**-plane** *Number*] [**-fg** *Color*] [**-bg** *Color*]

Description

The **xwud** command retrieves the dumped image of an Enhanced X-Windows window. It does so by displaying in a window an image saved in a specially formatted dump file previously produced by the **xwd** command. The dump file format is determined by the **XWDFile.h** file.

You can use flags to specify color display, window size and position, input field, and visual class or identification. You can also select a single bit plane of the image to display.

Flags

-bg *Color*
Specifies the color to display for the **0** (zero) bits in the image if a bitmap image (or a single plane of an image) is displayed.

-display *Display*
Specifies the server to connect to; see the **X** command.

-fg *Color*
Specifies the color to display for the **1** bits in the image if a bitmap image (or a single plane of an image) is displayed.

-geometry *Geometry*
Specifies the size and position of the window. Typically, you will only specify the position and let the size default to the actual size of the image.

-help
Prints a short description of the allowable options.

-in *FileName*
Specifies the input file on the command line. If the input file is not specified, the standard input is assumed.

-new
Creates a new color map for displaying the image. If the image characteristics match those of the display, this flag can display the image on the screen faster, but at the cost of using a new color map (which on most terminals causes other windows to go technicolor).

-noclick
Prevents the application from ending when a button in the window is clicked. You can end the application by typing a q or Q character, or the Ctrl-C key sequence.

-plane *Number*
Selects a single bit plane of the image to display. Planes are numbered, with 0 (zero) being the least significant bit. Use this flag to determine which plane to pass to the **xpr** command for printing.

-raw
Displays the dumped image in whatever color values currently exist on the screen. This flag is useful when undumping an image back onto the same screen that the image originally came from, while the original windows are still on the screen. This results in getting the image on the screen faster.

-rv
Swaps the foreground and background colors if a bitmap image (or a single plane of an image) displays. This flag is useful when displaying a bitmap image that has the color sense of pixel values 0 and 1 reversed from what they are on the display.

-std *MapType*
Uses the specified Standard Colormap to display the image. You can obtain the map type by converting the type to uppercase letters, prepending **RGB_** and appending **_MAP**. Typical map types are **best**, **default**, and **gray**. See the **/usr/lpp/X11/Xamples/clients/xstdcmap** for information about creating Standard Colormaps.

-vis visual_type | **visual_id**
Specifies a particular visual type or visual id. The default picks the **best** one or you can specify **default**, which is the same class as the colormap of the root window.

You can specify a particular class: **StaticGray**, **GrayScale**, **StaticColor**, **PseudoColor**, **DirectColor**, **TrueColor**. Specify **Match** to use the same class as the source image.

Specify an exact visual id (specific to the server) as a hexadecimal number (prefixed with 0x) or as a decimal number. This string is not case sensitive.

Environment Variables

DISPLAY Gets the default display.

Example

To retrieve a specific file from the dump window, enter:

```
xwud -in FileName
```

Related Information

The **X** command, **xpr** command, **xwd** command.

---

yacc Command

Purpose

Generates an LALR(1) parsing program from input consisting of a context-free grammar specification.

Syntax

yacc [**-b** *Prefix*] [**-C**] [**-d**] [**-l**] [**-Nn***Number*] [**-Nm***Number*] [**-Nr***Number*] [**-p** *Prefix*] [**-s**] [**-t**] [**-v**] [**-y** *Path*] *Grammar*

Description

The **yacc** command converts a context-free grammar specification into a set of tables for a simple automaton that executes an LALR(1) parsing algorithm. The grammar can be ambiguous; specified precedence rules are used to break ambiguities.

You must compile the output file, **y.tab.c**, with a C language compiler to produce a **yyparse** function. This function must be loaded with the **yylex** lexical analyzer, as well as with the **main** subroutine and the **yyerror** error-handling subroutine (you must provide these subroutines). The **lex** command is useful for creating lexical analyzers usable by the **yyparse** subroutine. Simple versions of **main** and **yyerror** subroutines are available through the **yacc** library, **liby.a**. Also, **yacc** can be used to generate C++ output.

You can compile the **yacc**-generated C file **(y.tab.c)** with the **-DYACC_MSG** option to include code necessary to use the Message Facility. When you use this option during compilation, error messages generated by the **yyparse** subroutine and the **YYBACKUP** macro are extracted from the **yacc_user.cat** catalog.

This allows you to receive error messages in languages other than English in non-English locales. If the catalog cannot be found or opened, the **yyparse** and **YYBACKUP** subroutines display the default English messages.

The **yacc** command is affected by the **LANG**, **LC_ALL**, **LC_CTYPE**, and **LC_MESSAGES** environment variables.

Flags

-b *Prefix* Use *Prefix* instead of **y** as the prefix for all output file names. The code file **y.tab.c**, the header file **y.tab.h** (created when **-d** is specified), and the description file **y.output** (created when **-v** is specified) are changed to *Prefix*.**tab.c**, *Prefix*.**tab.h**, and *Prefix*.**output**, respectively.

-C Produces the **y.tab.C** file instead of the **y.tab.c** file for use with a C++ compiler. To use the I/O Stream Library for input and output, define the macro, **_CPP_IOSTREAMS**.

-d Produces the file **y.tab.h**. This contains the **#define** statements that associate the **yacc**-assigned token codes with your token names. This allows source files other than **y.tab.c** to access the token codes by including this header file.

-l Does not include any **#line** constructs in **y.tab.c**. Use this only after the grammar and associated actions are fully debugged.

-Nn*Number* Changes the size of the token and nonterminal names array to *Number*. The default value is 8000. Valid values are only those greater than 8000.

-Nm*Number* Changes the size of the memory states array to *Number*. Default value is 40000. Valid values are only those greater than 40000.

-Nr*Number* Changes the internal buffer sizes to handle large grammars. The default value is 2000. Valid values are only those greater than 2000.

-p *Prefix* Use *Prefix* instead of **yy** as the prefix for all external names created by the **yacc** command. External names affected include: **yychar**, **yylval**, **yydebug**, **yyparse( )**, **yylex( )**, and **yyerror( )**. (Previously, **-p** was used to specify an alternate parser; now, **-y***Path* can be used to specify an alternate parser.)

-s Breaks the **yyparse** function into several smaller functions. Since its size is somewhat proportional to that of the grammar, it is possible for the **yyparse** function to become too large to compile, optimize, or execute efficiently.

-t Compiles run-time debugging code. By default, this code is not included when **y.tab.c** is compiled. However, the run-time debugging code is under the control of the preprocessor macro, **YYDEBUG**. If **YYDEBUG** has a nonzero value, the C compiler (**cc**) includes the debugging code, regardless of whether the **-t** flag is used. **YYDEBUG** should have a value of 0 if you don't want the debugging code included by the compiler. Without compiling this code, the **yyparse** subroutine will have a faster operating speed.

The **-t** flag causes compilation of the debugging code, but it does not actually turn on the debug mode. To get debug output, the **yydebug** variable must be set either by adding the C language declaration, `int yydebug=1` to the declaration section of the **yacc** grammar file or by setting **yydebug** through **dbx**.

-v Prepares the file **y.output**. It contains a readable description of the parsing tables and a report on conflicts generated by grammar ambiguities.

-y *Path* Uses the parser prototype specified by *Path* instead of the default **/usr/lib/yaccpar** file. (Previously, **-p** was used to specify an alternate parser.)

Exit Status

This command returns the following exit values:

0 Successful completion.

>0 An error occurred.

Examples

1. The following command:

   ```
   yacc grammar.y
   ```

   draws **yacc** rules from the **grammar.y** file, and places the output in **y.tab.c**.

2. The following command:

   ```
   yacc  -d grammar.y
   ```

   functions the same as example 1, but it also produces the **y.tab.h** file which would contain C-style **#define** statements for each of the tokens defined in the **grammar.y** file.

Files

y.output	Contains a readable description of the parsing tables and a report on conflicts generated by grammar ambiguities.
y.tab.c	Contains an output file.
y.tab.h	Contains definitions for token names.
yacc.tmp	Temporary file.
yacc.debug	Temporary file.
yacc.acts	Temporary file.
/usr/ccs/lib/yaccpar	Contains parser prototype for C programs.
/usr/ccs/lib/liby.a	Contains a run-time library.

Related Information

The **lex** command.

Generating a Lexical Analyzer with the lex Command in *AIX 5L Version 5.3 General Programming Concepts: Writing and Debugging Programs*.

The Example program for the lex and yacc programs in *AIX 5L Version 5.3 General Programming Concepts: Writing and Debugging Programs*.

---

yes Command

Purpose

Outputs an affirmative response repetitively.

Syntax

yes [*charstring*]

Description

The **yes** command outputs an affirmative response repetitively. Use the **yes** command as piped input to another command that requires an affirmative response before it completes the specified action. For example, the **yes** command is useful when deleting multiple files from a directory. The Ctl-C key sequence terminates the continuous affirmative responses.

> **Note:** The current locale is determined by the **LC_MESSAGES** environment variable or the *charstring* parameter, if specified. The *charstring* parameter can be any single character or character stream. If you enter an *charstring* parameter after issuing the **yes** command, the *charstring* parameter displays to the screen until you type the Ctl-C key sequence.

Example

To display the word `first` to the screen, type:

```
yes first
```

This statement displays the word until you enter the Ctl-C key sequence.

File

/usr/bin/yes	Contains the **yes** command.

Related Information

The **environment** file.

Shells in *Operating system and device management*.

---

ypbind Daemon

Purpose

Enables client processes to bind, or connect, to an NIS server.

Syntax

/usr/lib/netsvc/yp/ypbind [**-s -ypset -ypsetme**]

Description

The **ypbind** daemon binds, or connects, processes on a Network Information Services (NIS) client to services on an NIS server. This daemon, which runs on every NIS client, is started and stopped by the following System Resource Controller (SRC) commands:

```
startsrc -s ypbind
stopsrc -s ypbind
```

When a client requests information from a Network Information Services (NIS) map, the **ypbind** daemon broadcasts on the network for a server. When the server responds, it gives the daemon the Internet address and port number of a host. This is the host that provides the information the client is seeking. The **ypbind** daemon stores this address information in the **/var/yp/binding** directory using a file name of **domainname.version**. Then, the next time the client wants to access an NIS map, the client's **ypbind** daemon refers to the addresses in the **domainname.version** file.

The **ypbind** daemon can maintain bindings to several domains and their servers **-ypsetme** simultaneously. The default domain is the one specified by the **domainname** command at startup time.

Notes:

1. If a domain becomes unbound (usually when the server crashes or is overloaded), the **ypbind** daemon broadcasts again to find another server.
2. To force a client to bind to a specific server, use the **ypset** command.
3. To find out which server a client is bound to, use the **ypwhich** command.
4. If the **/var/yp/binding/domainname/ypservers** file exists, **ypbind** will attempt to contact the servers listed in that file before broadcasting. The file should contain a list of server IP addresses, one per line.
5. By default, the NIS client will wait indefinitely for the NIS server, during which time, logins to the client system are not possible. It is possible, however, to limit the length of this wait. If the **YPBIND_MAXWAIT** environment variable is set (usually in **/etc/environment**) before the **ypbind** daemon is started, this value (in seconds) will limit the amount of time the NIS client will wait for the NIS server. If this limit is exceeded, the client behaves as if NIS were unavailable and continues using local files. This will allow local logins, such as root.
6. If a domain becomes unbound and it is listed in the **/var/yp/binding/domainnanme/ypservers** file, by default **ypbind** daemon attempts to contact the server that is currently down; however, if the **YPBIND_SKIP** environment variable is set to 1 (usually set in the **/etc/environment** file) before the **ypbind** daemon is started, the server that is currently down will not be contacted again.

Flags

-s	Runs the **ypbind** daemon in a secure mode on privileged communications ports.
-ypset	Indicates the local host accepts **ypset** commands from local or remote hosts.

-ypsetme Indicates that the local host accepts **ypset** commands only from the local host. This flag overrides the **-ypset** flag if both are specified.

Notes:

1. If neither the **-ypset** or **-ypsetme** flags are specified, the local host rejects all **ypset** commands from all hosts. This is the most secure mode because the NIS server cannot change.
2. If neither the **-ypset** or **-ypsetme** flags are specified, the local host rejects all **ypset** commands from all hosts. This is the most secure mode because the NIS server cannot change. However, if no NIS servers exist on the networks directly connected to the client machine, then the **-ypsetme** flag must be used and the NIS server should be specified with the **ypset** command.

Files

/var/yp/binding directory	Contains Internet addresses and port numbers for NIS servers.
/var/yp/binding/domainname/ypservers	Contains a list of internet addresses, one per line, of servers to attempt to contact before broadcasting.
domainname.version	Binary file that contains the address and port number of the current NIS server.

Related Information

The **domainname** command, **makedbm** command, **mkclient** command, **mkmaster** command, **mkslave** command, **ypcat** command, **ypinit** command, **ypmatch** command, **yppoll** command,**yppush** command, **ypset** command, **ypwhich** command, **ypxfr** command.

System Resource Controller in *Operating system and device management*.

Network File System (NFS) Overview for System Management in *Networks and communication management*.

Network Information Services (NIS) Overview for System Management in *AIX 5L Version 5.3 Network Information Services (NIS and NIS+) Guide*.

NIS Reference.

---

ypcat Command

Purpose

Prints out a Network Information Services (NIS) map.

Syntax

To Display the Network Information Services Database

/usr/bin/ypcat [**-k**] [**-t**] [**-d** *DomainName*] *MapName*

To Display the Nickname Translation Table

/usr/bin/ypcat -x

Description

The **ypcat** command prints out the Network Information Services (NIS) map you specify with the *MapName* parameter. You can specify either a map name or a map nickname. Because the **ypcat** command uses the NIS service, you do not need to specify a server.

Flags

-k	Displays the keys for those maps in which the values are null or for which the key is not part of the value. (None of the maps derived from files that have an ASCII version in the **/etc** directory fall into this class.)
-t	Indicates that the name specified by the *MapName* parameter is *not* a nickname. This flag causes the **ypcat** command to bypass the nickname translation table and search only for the map specified by the *MapName* parameter.
-d *DomainName*	Searches the specified domain for the specified map.
-x	Displays the nickname translation table. This table lists the map nicknames the command knows of and indicates the map name (as specified by the *MapName* parameter) associated with each nickname.

Examples

1. To look at the networkwide password map, **passwd.byname**, type:

   ```
   ypcat passwd
   ```

   In this example, `passwd` is the nickname for the **passwd.byname** map.
2. To locate a map, type:

   ```
   ypcat -t passwd
   ```

   In this example, the **ypcat** command bypasses any maps with the nickname of `passwd` and searches for a map with the full name of `passwd`.
3. To display a map in another domain, type:

   ```
   ypcat -d polaris passwd
   ```

   In this example, the **ypcat** command locates the map named `passwd` in the domain named `polaris`.
4. To display the map nickname translation table, type:

   ```
   ypcat -x
   ```

   In this example, the **ypcat** command displays a list of map nicknames and their associated map names.

Related Information

The **domainname** command, **ypmatch** command.

The **ypserv** daemon.

Network File System (NFS) Overview for System Management in *Networks and communication management*.

Network Information Services (NIS) Overview for System Management in *AIX 5L Version 5.3 Network Information Services (NIS and NIS+) Guide*.

NIS Reference.

---

ypinit Command

Purpose

Sets up NIS maps on a Network Information Services (NIS) server.

Syntax

To Set up NIS on an NIS Master Server

/usr/sbin/ypinit [**-o**] [**-n**] [**-q**] **-m** [*SlaveName ...*]

To Set up NIS on an NIS Slave Server

/usr/sbin/ypinit -s *MasterName*

Description

The **ypinit** command sets up NIS maps on a Network Information Services (NIS) master server or NIS slave server. Only users with root user authority can use the **ypinit** command.

By default, the **ypinit** command uses the ASCII system files as input files for the map being created.

Flags

-m [*SlaveName...*]	Indicates that the local host is to be the NIS master. If the **-q** flag is used the **-m** flag can be followed by the names of the machines that will be the NIS slave servers.
-n	Indicates that the **ypinit** command is not to stop if it finds errors.
-o	Allows any existing maps for the current NIS domain to be overwritten.
-q	Indicates that the **ypinit** command is to get arguments from the command line instead of prompting for input.
-s *MasterName*	Copies NIS maps from the server workstation you specify in the *MasterName* parameter.

Examples

1. To set up an NIS master server that functions as the master for all NIS maps, type the following command on the command line:

   ```
   ypinit -m
   ```

   This command invokes the **make** procedure, which follows the instructions in the **/var/yp/Makefile** file.
2. To set up an NIS slave server, type:

   ```
   ypinit -s zorro
   ```

   In this example, the **ypinit** command copies the NIS maps onto your workstation from the NIS server named `zorro`, making your workstation an NIS slave server.
3. To set up an NIS master server without being prompted for input, type:

   ```
   ypinit -o -n -q -m slave
   ```

   Note: If the system has previously been configured as an NIS master server, ensure that the directory, **/var/yp/binding**, is removed before executing **ypinit**. If old information is stored in **/var/yp/binding**, it may cause errors to occur during configuration of the NIS master server.

Files

/etc/bootparams	Lists clients that diskless clients can use for booting.
/etc/passwd	Contains an entry for each user that has permission to log on to the machine.
/etc/group	Contains an entry for each user group allowed to log on to the machine.
/etc/hosts	Contains an entry for each host on the network.
/var/yp/Makefile	Contains rules for making NIS maps.
/etc/networks	Contains the name of each network in the DARPA Internet.
/etc/netmasks	Lists network masks used to implement IP standard subnetting.

/etc/netid	Contains identification information for machines, hosts, and groups.
/etc/rpc	Contains map information for RPC programs.
/etc/services	Contains an entry for each server available through the Internet.
/etc/protocols	Defines Internet protocols used on the local host.
/etc/netgroup	Contains information about each user group on the network.
/etc/ethers	Contains the Ethernet addresses of hosts on the Internet network.
/etc/publickey	Contains public or secret keys for NIS maps.

Related Information

The **chmaster** command, **chslave** command, **lsmaster** command, **makedbm** command, **mkmaster** command, **mkslave** command, **yppush** command, **ypxfr** command.

The **ypserv** daemon.

Network Information Services (NIS) Overview for System Management in *AIX 5L Version 5.3 Network Information Services (NIS and NIS+) Guide*.

NIS Reference.

---

ypmatch Command

Purpose

Displays the values of given keys within a Network Information Services (NIS) map.

Syntax

To Display Key Values for an NIS Map

/usr/bin/ypmatch [**-d** *Domain*] [**-k**] [**-t**] *Key*... *MapName*

To Display the NIS Map Nickname Table

/usr/bin/ypmatch -x

Description

The **ypmatch** command displays the values associated with one or more keys within a Network Information Services (NIS) map. Use the *MapName* parameter to specify either the name or nickname of the map you want to search.

When you specify multiple keys in the *Key* parameter, the system searches the same map for all of the keys. Because pattern matching is not available, match the capitalization and length of each key exactly. If the system does not find a match for the key or keys you specify, a diagnostic message is displayed.

Flags

-d *Domain*	Specifies a domain other than the default domain.
-k	Prints a key followed by a colon before printing the value of the key. This is useful only if the keys are not duplicated in the values or if you have specified so many keys that the output could be confusing.
-t	Inhibits translation of nickname to map name.
-x	Displays the map nickname table. This lists the nicknames (as specified by the *MapName* parameter) the command knows of and indicates the map name associated with each nickname.

Examples

To display the value associated with a particular key, type:

```
ypmatch  -d ibm  -k host1 hosts
```

In this example, the **ypmatch** command displays the value of the `host1` key from the `hosts` map in the `ibm` domain.

Related Information

The **ypcat** command.

Network Information Services (NIS) Overview for System Management in *AIX 5L Version 5.3 Network Information Services (NIS and NIS+) Guide*.

NIS Reference.

---

yppasswd Command

Purpose

Changes your network password in Network Information Services (NIS).

Syntax

yppasswd [**-f** [*Name*] | **-s** [*Name* [*ShellProg*]]]

Description

The **yppasswd** command changes (or installs) a network password and associates it with the name you specify in the *UserName* parameter. To create or change a password, you must be the owner of the password you want to change. The Network Information Services (NIS) password can be different from the one on your own machine. Root users on an NIS server can change the password of another user without knowing the user's original password. To do this, the Root user enters their password in place of the user's original password. Root users on an NIS client, however, do not have this privilege.

When you enter the **yppasswd** command on the command line, the system prompts you to enter the old password. When you do this, the system prompts you to enter the new password. The password you enter can be as small as four characters long if you use a mixture of uppercase and lowercase characters. Otherwise, the password has to be six characters long or longer. These rules are relaxed if you are insistent enough.

If you enter the old password incorrectly, you have to enter the new password before the system will give you an error message. The system requires both passwords because the **update** protocol sends them to the server at the same time. The server catches the error and notifies you that you entered the old password incorrectly.

To verify the new password, the system prompts you to enter it again. For this new password to take effect, the **yppasswdd** daemon must be running on your NIS server.

Note: The **yppasswd** command cannot establish rules for passwords as does the **passwd** command.

Flags

-f [*Name*] — Changes user *Name*'s gecos information in the NIS maps. Gecos information is general information stored in the **/etc/passwd** file.

-s [*Name* [*ShellProg*]] Changes user *Name*'s login shell in the NIS maps.

Example

1. To change a user's NIS password, enter:

   ```
   yppasswd Joe
   ```

   This example demonstrates how to change the NIS password for the user named `Joe`. The system prompts you to enter Joe's old password and then his new password.
2. To change the login shell to **/bin/ksh** for the user named `Joe`, if the **yppasswdd** daemon has not been started with the **-noshell** flag, enter:

   ```
   yppasswd  -s Joe /bin/ksh
   ```
3. To change the gecos information in the **passwd** file for the user named `Joe`, if the **yppasswdd** daemon has not been started with the **-nogecos** flag, enter:

   ```
   yppasswd  -f Joe
   Old NIS password:
   Joe's current gecos:
   John Doe Test User Id
   Change (yes) or (no)? >y
   To?>Joe User Test User Id
   ```

Related Information

The **yppasswdd** daemon.

Network Information Services (NIS) Overview for System Management in *AIX 5L Version 5.3 Network Information Services (NIS and NIS+) Guide*.

NIS Reference.

---

yppasswdd Daemon

Purpose

Receives and executes requests from the **yppasswd** command.

Syntax

rpc.yppasswdd *FileName* [**-nogecos**] [**-nopw**] [**-noshell**] [**-r** | **-m** [*Argument*...]]

Description

The **yppasswdd** daemon is a server that receives and executes requests for new passwords from the **yppasswd** command. These requests require the daemon to verify the user's old password and change it. The daemon changes the password in the file you specify in the *FileName* parameter, which has the same format as the **/etc/passwd** file.

To make it possible to update the Network Information Services (NIS) password map from remote machines, the **yppasswdd** daemon must be running on the master server that contains the NIS password map.

Note: The **yppasswdd** daemon is not run by default, nor can it be started up from the **inetd** daemon like other Remote Procedure Call (RPC) daemons.

The **yppasswdd** daemon can be started and stopped with the following System Resource Controller (SRC) commands:

```
startsrc -s yppasswdd
stopsrc -s yppasswdd
```

Flags

-m	Runs the **make** command using the makefile in the **/var/yp** directory. This adds the new or changed password to the NIS password map. Any arguments that follow the **-m** flag are passed to the **make** command.
-nogecos	Indicates the server will not accept changes for gecos information from the **yppasswd** command.
-nopw	Indicates that the server will not accept password changes from the **yppasswdd** command.
-noshell	Indicates the server will not accept changes for user shells from the **yppasswd** command.
-r	Directly updates the **/var/yp/***domainname***/passwd.byname** and **/var/yp/***domainname***/passwd.byuid** database files on the Master server as well as any Slave servers with new or changed passwords. This option is faster than the **-m** flag because the **make** command is not run. The **-r** flag is useful when the database files are large (several thousand entries or more).

Note: The System Resource Controller (SRC) starts the **yppasswdd** daemon with the **-m** flag specified by default. Use the **chssys** command to change the default to the **-r** flag.

Example

To propagate updated passwords immediately, invoke the **yppasswdd** daemon as follows:

```
startsrc -s yppasswdd
```

Files

/etc/inetd.conf	Defines how the **inetd** daemon handles Internet service requests.
/var/yp/Makefile	Contains rules for making NIS maps.
/etc/rc.nfs	Contains the startup script for the NFS and NIS daemons.
/etc/security/passwd	Stores password information.

Related Information

The **chssys** command, **domainname** command, **make** command, **passwd** command, **startsrc** command, **yppasswd** command.

The **inetd** daemon.

The **/etc/security/passwd** file.

System Resource Controller in *Operating system and device management*.

Network File System (NFS) Overview for System Management in *Networks and communication management*.

Remote Procedure Call (RPC) Overview for Programming in *AIX 5L Version 5.3 Communications Programming Concepts*.

Network Information Services (NIS) Overview for System Management in *AIX 5L Version 5.3 Network Information Services (NIS and NIS+) Guide*.

NIS Reference.

yppoll Command

Purpose

Displays the order number (ID number) of the Network Information Services (NIS) map currently in use on the server.

Syntax

/usr/sbin/yppoll [**-h** *Host*] [**-d** *Domain*] *MapName*

Description

The **yppoll** command uses the **ypserv** daemon to display the order number of the map you specify in the *MapName* parameter. An order number is a map's ID number and is assigned by the system. This number changes whenever a map is updated. Use the **yppoll** command whenever you want to make sure your servers are using the most current version of a particular map.

The **yppoll** command can run on systems that have either version 1 or version 2 of the Network Information Services (NIS) protocol installed. Be aware, however, that each version of the protocol has its own set of diagnostic messages.

Note: When specifying a *MapName*, be sure to enter the map's full name. The **yppoll** command does not recognize map nicknames.

Flags

-h *Host*	Enables you to specify a server other than the default server. To find out which server the command defaults to, use the **ypwhich** command.
-d *Domain*	Enables you to specify a domain other than the default domain. To find out which domain the command defaults to, use the **domainname** command.

Examples

1. To look at a map located on a particular host, type:

   ```
   /usr/sbin/yppoll -h thor netgroups.byuser
   ```

   In this example, the **yppoll** command displays the order number for the `netgroups.byuser` map located on the host named `thor`.
2. To look at a map on a domain, type:

   ```
   /usr/sbin/yppoll -d atlantis hosts.byname
   ```

   In this example, the **ypoll** command displays the order number for the `hosts.byname` map located in the domain `atlantis`.

Related Information

The **domainname** command, **ypwhich** command.

The **ypserv** daemon.

Network Information Services (NIS) Overview for System Management in *AIX 5L Version 5.3 Network Information Services (NIS and NIS+) Guide*.

NIS Reference.

yppush Command

Purpose

Prompts the Network Information Services (NIS) slave servers to copy updated NIS maps.

Syntax

/usr/sbin/yppush [**-v**] [**-d** *Domain*] *MapName*

Description

The **yppush** command, which is issued from the **/usr/etc/yp** directory, prompts the Network Information Services (NIS) slave servers to make copies of updated NIS maps. The *MapName* variable specifies that map to be transferred to the slave servers of the master servers. To get a list of the servers it needs to prompt, the **yppush** command reads the **ypservers** map, specified by the *Domain* parameter or the current default domain. When prompted, each slave server uses the **ypxfr** command to copy and transfer the map back to its own database.

You can use the System management interface tool (SMIT) to run this command. To use SMIT, type:

```
smit yppush
```

Note: If your system uses version 1 of the NIS protocol, the **ypxfr** command is not the transfer agent.

Flags

-d *Domain*	Specifies a domain other than the default domain. The maps for the specified domain must exist.
-v	Displays messages as each server is called and then displays one message for each server's response, if you are using the version 2 protocol. If this flag is omitted, the command displays error messages only. **Note:** Version 1 of the NIS protocol does not display messages. If your system uses version 1, use the **yppoll** command to verify that the transfer took place.

Examples

1. To copy a map from another domain to the slave servers, type:

   ```
   /usr/sbin/yppush -d atlantis netgroup
   ```

   In this example, the **yppush** command copies the `netgroup` map from the `atlantis` domain.
2. To display the in-progress status of the **yppush** command as it calls each slave server, type:

   ```
   /usr/sbin/yppush -v -d atlantis netgroup
   ```

   In this example, the **yppush** command displays in-progress messages as it copies the `netgroup` map from the `atlantis` domain onto each of the network's slave servers.

Files

/var/yp/*DomainName***/ypservers.{dir, pag}**	Lists servers that the **yppush** command prompts to make copies of updated NIS maps.

Related Information

The **yppoll** command, **ypxfr** command.

The **ypserv** daemon.

System Management Interface Tool (SMIT) in in *AIX 5L Version 5.3 General Programming Concepts: Writing and Debugging Programs*.

Network File System (NFS) Overview for System Management in *Networks and communication management* and NIS **Maps** in *AIX 5L Version 5.3 Network Information Services (NIS and NIS+) Guide*.

Network Information Services (NIS) Overview for System Management in *AIX 5L Version 5.3 Network Information Services (NIS and NIS+) Guide*.

NIS Reference.

---

ypserv Daemon

Purpose

Looks up information in local Network Information Services (NIS) maps.

Syntax

/usr/lib/netsvc/yp/ypserv

Description

The **ypserv** daemon looks up information in its local Network Information Services (NIS) maps. The operations performed by the **ypserv** daemon are defined for the implementor by the NIS Protocol Specification and for the programmer by the **/usr/include/rpcsvc/yp_prot.h** header file. Communication with the **ypserv** daemon is by means of Remote Procedure Calls (RPC).

The **ypserv** daemon runs only on server machines. The **ypserv** daemon is started and stopped by the following System Resource Controller (SRC) commands:

```
startsrc -s ypserv
stopsrc -s ypserv
```

The **ypserv** daemon performs the following operations on a specified map within an NIS domain:

Match	Takes a key and returns the associated value.
Get_first	Returns the first key-value pair from the map.
Get_next	Enumerates the next key-value pair in the map.
Get_all	Ships the entire NIS map to a requestor in response to a single RPC request.
Get_order_number	Supplies information about a map instead of map entries. The order number actually exists in the map as a key-value pair, but the server does not return it through the normal lookup functions. However, the pair will be visible if you examine the map with the **makedbm** command.
Get_master_name	Supplies information about a map instead of map entries. The master name actually exists in the map as a key-value pair, but the server does not return it through the normal lookup functions. However, the pair will be visible if you examine the map with the **makedbm** command.

Log information is written to the **/var/yp/ypserv.log** file if it exists when the **ypserv** daemon starts running.

If the **/var/yp/securenets** file exists, the **ypservr** command only responds to hosts within the ip range specified in this file.

Files

/etc/rc.nfs	Contains the startup script for the NFS and NIS daemons.

/var/yp/ypserv.log Contains the log for the **ypserv** daemon.

Related Information

The **chmaster** command, **chslave** command, **domainname** command, **makedbm** command, **mkmaster** command, **mkslave** command, **ypcat** command, **ypinit** command, **ypmatch** command, **yppoll** command, **yppush** command, **ypset** command, **ypwhich** command, **ypxfr** command.

System Resource Controller in *Operating system and device management*.

Network File System (NFS) Overview for System Management in *Networks and communication management*, NIS Maps in *AIX 5L Version 5.3 Network Information Services (NIS and NIS+) Guide*.

Remote Procedure Call Overview for Programming in *AIX 5L Version 5.3 Communications Programming Concepts*.

How to Configure NIS in *Networks and communication management*.

Network Information Services (NIS) Overview for System Management in *AIX 5L Version 5.3 Network Information Services (NIS and NIS+) Guide*.

NIS Reference.

---

ypset Command

Purpose

Directs a client machine to a specific server.

Syntax

/usr/sbin/ypset [**-V1**] [**-d** *Domain*] [**-h** *Host*] *Server*

Description

The **ypset** command directs the **ypbind** daemon on the client to the **ypserv** daemon on the server. The **ypbind** daemon goes to the server you specify in the *Server* parameter to get Network Information Services (NIS) services for the domain you specify in the *Domain* parameter. The **ypbind** daemon gets the NIS services from the **ypserv** daemon on the server.

After the binding is set, it is not tested until a client process (such as the **ypcat** command or the **ypwhich** command) tries to get a binding for the domain. If the attempt to bind fails (the specified server is down or is not running the **ypserv** daemon), the **ypbind** daemon makes another attempt to bind for the same domain.

Specify either a name or an Internet Protocol (IP) address in the *Server* parameter. If you specify a name, the **ypset** command attempts to resolve the name to an IP address through the use of the NIS service. This works only if your machine has a current valid binding for the domain in question. In most cases, you should specify the server as an IP address.

In cases where several hosts on the local network are supplying NIS services, the **ypbind** daemon can rebind to another host. If a server is down or is not running the **ypserv** daemon, the **ypbind** daemon rebinds the client to another server. In this way, the network information service balances the load among the available NIS servers.

Use the **ypset** command if the network:

- Does not support broadcasting.
- Supports broadcasting but does not have an NIS server.
- Accesses a map that exists only on a particular NIS server.

An alternative to using **ypset** is to use the **/var/yp/binding/***domain_name***/ypservers** file. This file, if present, should contain a list of NIS servers to attempt to bind to, one server per line. If the **ypbind** daemon cannot bind to any of the servers in the **ypservers** file, then it will attempt to use the server specified by **ypset**. If that fails, it will broadcast on the subnet for a NIS server.

Flags

-d *Domain*	Specifies a domain other than the default domain.
-h *Host*	Sets the binding for the **ypbind** daemon on the specified host instead of on the local host. The host can be specified as a name or as an IP address.
-V1	Binds the specified server for the (old) version 1 NIS protocol.

Example

To set a server to bind on a host in a particular domain, enter:

```
ypset -d ibm -h venus mars
```

In this example, the **ypset** command causes the host named `venus` to bind to the server named `mars`.

Related Information

The **domainname** command, **ypcat** command, **ypwhich** command,

The **ypbind** daemon, **ypserv** daemon.

Network File System (NFS) Overview for System Management in *Networks and communication management*.

How to Configure NIS in *AIX 5L Version 5.3 Network Information Services (NIS and NIS+) Guide*.

Network Information Services (NIS) Overview for System Management in *AIX 5L Version 5.3 Network Information Services (NIS and NIS+) Guide*.

NIS Reference.

---

ypupdated Daemon

Purpose

Updates information in Network Information Services (NIS) maps.

Syntax

/usr/lib/netsvc/yp/rpc.ypupdated [**-i** | **-s**]

Description

The **ypupdated** daemon updates information in Network Information Services (NIS) maps. Before it can update information, however, the daemon consults the **updaters** file in the **/var/yp** directory to determine which NIS maps should be updated and how they should be changed.

By default, the **ypupdated** daemon requires the most secure method of authentication available to it, either DES (secure) or UNIX (insecure).

The **ypupdated** daemon is started and stopped by the following System Resource Controller (SRC) commands:

```
startsrc -s ypupdated

stopsrc -s ypupdated
```

Flags

-s Accepts only calls authenticated using the secure Remote Procedure Call (RPC) mechanism (AUTH_DES authentication). This disables programmatic updating of NIS maps unless the network supports these calls.

-i Accepts RPC calls with the insecure AUTH_UNIX credentials. This allows programmatic updating of NIS maps in all networks.

Examples

To start the **ypupdated** daemon from the command line, type:

```
startsrc -s ypupdated
```

File

/var/yp/updaters A makefile for updating NIS maps.

Related Information

The **startsrc** command.

The **keyserv** daemon.

System Resource Controller in *Operating system and device management*.

Remote Procedure Call Overview for Programming in *AIX 5L Version 5.3 Communications Programming Concepts*.

Network Information Services (NIS) Overview for System Management in *AIX 5L Version 5.3 Network Information Services (NIS and NIS+) Guide*.

NIS Reference.

---

ypwhich Command

Purpose

Identifies either the Network Information Services (NIS) server or the server that is the master for a given map.

Syntax

To Identify the NIS Server

/usr/bin/ypwhich [**-d** *Domain*] [**-V1** | **-V2**] [*HostName*]

To Identify the Master NIS Server for a Map

/usr/bin/ypwhich [**-t**] [**-d** *Domain*] [**-m** [*MapName*]]

To Display the Map Nickname Table

/usr/bin/ypwhich -x

Description

The **ypwhich** command identifies which server supplies Network Information Services (NIS) services or which server is the master for a map, depending on how the **ypwhich** command is invoked. If invoked without arguments, this command displays the name of the NIS server for the local machine. If you specify a host name, the system queries that host to find out which master it is using.

Flags

-d *Domain*	Uses the specified domain instead of the default domain.
-V1	Indicates which server is serving the old version 1 NIS protocol client processes.
-V2	Indicates which server is serving the current version 2 NIS protocol client processes. If neither version is specified, the **ypwhich** command attempts to locate the server that supplies the version 2 services. If there is no version 2 server currently bound, the **ypwhich** command then attempts to locate the server supplying version 1 services. Because servers and clients are both backward-compatible, the user need seldom be concerned about which version is currently in use.
-t	Inhibits nickname translation, which is useful if there is a map name identical to a nickname.
-m *MapName*	Finds the master NIS server for a map. No host can be specified with the **-m** flag. The *MapName* variable can be a map name or a nickname for a map. When the map name is omitted, the **-m** flag produces a list of available maps.
-x	Displays the map nickname table. This lists the nicknames (*MapName*) the command knows of and indicates the map name associated with each nickname.

Examples

1. To find the master server for a map, type:

   ```
   ypwhich -m passwd
   ```

   In this example, the **ypwhich** command displays the name of the server for the `passwd` map.
2. To find the map named `passwd`, rather than the map nicknamed `passwd`, type:

   ```
   ypwhich -t -m passwd
   ```

   In this example, the **ypwhich** command displays the name of the server for the map whose full name is `passwd`.
3. To find out which server serves clients that run the old version 1 of the NIS protocol, type:

   ```
   ypwhich -V1
   ```
4. To display a table of map nicknames, type:

   ```
   ypwhich -x
   ```

Related Information

The **ypset** command.

The **ypserv** daemon.

Network File System (NFS) Overview for System Management in *Networks and communication management*.

Network Information Services (NIS) Overview for System Management in *AIX 5L Version 5.3 Network Information Services (NIS and NIS+) Guide*.

NIS Reference.

---

ypxfr Command

Purpose

Transfers a Network Information Services (NIS) map from an NIS server to a local host.

Syntax

/usr/sbin/ypxfr [**-f**] [**-c**] [**-d** *Domain*] [**-h** *Host*] [**-s** *Domain*] [**-C** *TID Program IPAddress Port*] [**-S**] *MapName*

Description

The **ypxfr** command transfers a Network Information Services (NIS) map from an NIS server to the local host as follows:

1. Creates a temporary map in the **/var/yp/***Domain* directory (which must already exist) on the client.
2. Fetches the map entries from the server and fills in the map on the client, one at a time.
3. Gets and loads the map parameters (order number and server).
4. Deletes any old versions of the map.
5. Moves the temporary map to the real map name.

If the **/var/yp/securenets** file exists, the **ypxfr** command only responds to hosts that are listed in this file.

The *MapName* variable specifies the name of a map that will be transferred from an NIS server.

If run interactively, the **ypxfr** command sends output to the user's terminal. If invoked without a controlling terminal, the **ypxfr** command appends its output to the **/var/yp/ypxfr.log** file (if the file already exists). This file records each transfer attempt and its results. The **ypxfr** command is most often invoked from the root user's **crontab** file or by the **ypserv** daemon.

To maintain consistent information between servers, use the **ypxfr** command to update every map in the NIS database periodically. Be aware though that some maps change more frequently than others and therefore need to be updated more frequently. For instance, maps that change infrequently, such as every few months, should be updated at least once a month. Maps that change frequently, such as several times a day, should be checked hourly for updates. The **services.byname** map, for example, may not change for months at a time, while the **hosts.byname** map may change several times a day.

To perform periodic updates automatically, use a **crontab** entry. To update several maps at one time, group commands together in a shell script. Examples of a shell script can be found in the **/usr/etc/yp** directory in the following files: **ypxfr_1perday**, **ypxfr_2perday**, **ypxfr_1perhour**.

You can use the System management interface tool (SMIT) to run this command. To use SMIT, enter:

```
smit ypxfr
```

Flags

-C *TID Program IPAddress Port*
Tells the **ypxfr** command where to find the **yppush** command. The **ypserv** daemon invokes the **ypxfr** command to call back a **yppush** command to the host. Use the parameters to indicate the following:

TID Specifies the transaction ID of the **yppush** command.

Program
Specifies the program number associated with the **yppush** command.

IPAddress
Specifies the Internet Protocol address of the port where the **yppush** command resides.

Port Specifies the port that the **yppush** command is listening on.

Note: This option is only for use by the **ypserv** daemon.

-c
Prevents sending of a request to `Clear Current Map` to the local **ypserv** daemon. Use this flag if the **ypserv** daemon is not running locally at the time you are running the **ypxfr** command. Otherwise, the **ypxfr** command displays an error message and the transfer fails.

-d *Domain*
Specifies a domain other than the default domain. The maps for the specified domain must exist.

-f
Forces the transfer to occur even if the version at the master is not more recent than the local version.

-h *Host*
Gets the map from host specified, regardless of what the map says the master is. If a host is not specified, the **ypxfr** command asks the NIS service for the name of the master and tries to get the map from there. The *Host* variable can contain a name or an Internet address in the form `a.b.c.d`.

-S
Requires the **ypserv** server, from which it obtains the maps to be transferred, use *privileged* IP ports. Because only root user processes are typically allowed to use privileged ports, this feature adds an extra measure of security to the transfer. If the map being transferred is a secure map, the **ypxfr** command sets the permissions on the map to `0600`.

-s *Domain*
Specifies a source domain from which to transfer a map that should be the same across domains (such as the **services.byname** map).

Examples

To get a map from a host in another domain, enter:

```
/usr/sbin/ypxfr -d ibm -h venus passwd.byname
```

In this example, the **ypxfr** command gets the `passwd.byname` map from the host name `venus` in the `ibm` domain.

Files

/var/yp/ypxfr.log Contains the log file.
/usr/sbin/ypxfr_1perday Contains the script to run one transfer each day, for use with the **cron** daemons.
/usr/sbin/ypxfr_2perday Contains the script to run two transfers each day.
/usr/sbin/ypxfr_1perhour Contains the script for hourly transfers of volatile maps.

Related Information

The **crontab** command, **yppush** command.

The **cron** daemon, **ypserv** daemon.

System Management Interface Tool (SMIT) in in *AIX 5L Version 5.3 General Programming Concepts: Writing and Debugging Programs*.

Network File System (NFS) Overview for System Management in *Networks and communication management*, NIS **Maps** in *AIX 5L Version 5.3 Network Information Services (NIS and NIS+) Guide*.

Network Information Services **(NIS) Overview for System Management** in *AIX 5L Version 5.3 Network Information Services (NIS and NIS+) Guide*.

NIS **Reference**.

---

zcat Command

Purpose

Expands a compressed file to standard output.

Syntax

zcat [**-n**] [**-V**] [*File ...*]

Description

The **zcat** command allows the user to expand and view a compressed file without uncompressing that file. The **zcat** command does not rename the expanded file or remove the **.Z** extension. The **zcat** command writes the expanded output to standard output.

Flags

-n Omits the compressed file header from the compressed file.
Note: Use the **-n** option if the file was compressed using the **-n** option.

-V Writes the current version and compile options to standard error.

Parameters

File ... Specifies the compressed files to expand.

Return Values

If the **zcat** command exits with a status of 1 if any of the following events occur:

- The input file was not produced by the **compress** command.
- An input file cannot be read or an output file cannot be written.

If no error occurs, the exit status is 0.

Exit Status

0 Successful completion.

>0 An error occurred.

Examples

To view the `foo.Z` file without uncompressing it, enter:

```
zcat foo.Z
```

The uncompressed contents of the `foo.Z` file are written to standard output. The file is not renamed.

Related Information

The **compress** command, **pack** command, **uncompress** command, **unpack** command.

Commands in *Operating system and device management*.

---

zdump Command

Purpose

Dumps the time zone information.

Syntax

zdump [**-v**] [**-c** *CutOffYear*] *TimeZoneName* ...

Description

The **zdump** command prints the current time for each zone name specified on the command line.

Flags

-c *CutOffYear*
Stops the verbose output near the start of the year *CutOffYear*.

-v
For each TimeZoneName listed on the command line, the following information is printed:

- The current time.
- The time at the lowest possible time value.
- The time one day after the lowest possible time value.
- The times both one second before, and exactly at each time, when the rules change for computing local time.
- The time at the highest possible time value
- The time at one day less than the highest possible time value.

Each line ends with `isdst=1` if the given time is Daylight Saving Time, or `isdst=0` otherwise.

Parameters

TimeZoneName Represents the name of the zone whose time zone information is displayed.

Exit Status

0 The command completed successfully.

>0 An error occurred.

Examples

1. To report time zone information for Singapore , enter:

   ```
   zdump -v Singapore
   ```

2. To report verbose time zone information for Turkey with verbose output stopping near the start of the year 2035, enter:

```
zdump -v -c 2035 Turkey
```

Files

/usr/sbin/zdump	Contains the SystemV **zdump** command.
/usr/share/lib/zoneinfo	Contains the standard time zone directory.

Related Information

The **zic** command.

---

zic Command

Purpose

Provides a time zone compiler.

Syntax

zic [**-v**] [**-d** *Directory*] [**-l** *LocalTime*] [**-y** *YearIsType*] [*FileName* ...]

Description

The zic **command** processes text from the files named on the command line and creates the time conversion binary files specified in this input. The time zone information is processed from the standard input if file name is specified as - (hyphen).

Input lines in the specified files are made up of fields. Field separators are be any number of white space characters. A pound sign (#) in the input file indicates a comment that extends to the end of the line on which the pound sign appears. White space characters and pound signs can be enclosed in double quotes (″) if they are to be used as part of a field. Blank lines are ignored.

A rule line has the form:

```
Rule  NAME  FROM   TO   TYPE   IN   ON   AT   SAVE   LETTER/S
```

The fields that make up the rule line are as follows:

NAME Provides a random name for the set of rules for which this Rule is applicable.

FROM Provides the first year in which the rule applies. Specifying `minimum` (min) indicates the minimum year with a representable time value. Specifying `maximum` (max) indicates the maximum year with a representable time value.

TO Provides the final year in which the rule applies. This is indicated as a valid time value or by specifying `minimum` and `maximum`. Specifying `only` is used to repeat the value of the **FROM** field.

TYPE Provides the type of year in which the rule applies.

The **TYPE** field has the following values:

'-' The rule applies in all years between **FROM** and **TO**, inclusively.

uspres
The rule applies in U.S. Presidential election years.

nonpres
The rule applies in years other than U.S. Presidential election years.

If the **TYPE** field has a value other than what is mentioned above, the **zic** command runs the **/usr/sbin/yearistype year type** command to check the type of year.

The **yearistype** command accepts two parameters; the year and the type of year. An exit status of 0 is taken to mean that the year is of the given type. Otherwise, 1 is returned as exit status.

IN Represents the month in which the rule takes effect. Month names may be abbreviated.

ON Prepresents the day on which the rule takes effect. Recognized forms include:

- **lastFri** represents the last Friday in the month.
- **lastMon** represents the last Monday in the month.
- A number representing the day of the month. For example, 5 represents the fifth of the month.
- **lastSun** represents the last Sunday in the month.
- lastMon represents the last Monday in the month.
- Sun>=8 represents first Sunday on or after the eighth.
- Sun<=25 represents last Sunday on or before the 25th.

Names of days of the week can be abbreviated or spelled out in full. Note that there must be no spaces within the **ON** field.

AT Represents the time of day at which the rule takes effect. Recognized forms include:

- A number representing time in hours. For example, 2 indicates two hours.
- 2:00 indicates two o'clock in hours and minutes.
- 15:00 represents 3 o'clock in the afternoon using the 24-hour format time format.
- 1:28:14 indicates one o'clock, twenty-eight minutes and fourteen seconds, using the hours, minutes, seconds format.

Any of these forms may be followed by the letter `w` if the given time is local *wall-clock* time or `s` if the given time is local *standard* time. In the absence of `w` or `s`, wall-clock time is assumed.

Regions with more than two types of local time are required to use the local standard time in the **AT** field of the earliest transition time's rule to ensure the accuracy of the earliest transition time that is stored in the resulting time-zone binary.

SAVE Prepresents the amount of time to be added to local standard time when the rule is in effect. This field has the same format as the **AT** field The `w` and `s` suffixes are not valid with this field.

LETTER/S
Provides the *variable part* of the time-zone abbreviations that are used when this rule is in effect. When this field contains - (hyphen), the variable is null. The `S` character is used to indicate `EST` and the `D` character is used to indicate `EDT`.

A zone line has the form:

```
Zone   NAME   GMTOFF   RULES/SAVE   FORMAT   [UNTIL]
```

The fields that make up a zone line are:

NAME Indicates the name of the time zone. This is the name used to create the time conversion information file for the zone.

GMTOFF
Indicated the amount of time to add to GMT to get standard time in this zone. This field has the same format as the **AT** and **SAVE** fields of rule lines. Begin the field with a minus sign if time must be subtracted from GMT.

RULES/SAVE
Indicated the name of the rules that apply in the time zone or, alternately, an amount of time to add to local standard time. If value of this field is - (hyphen), then standard time always applies in the time zone.

FORMAT
Indicates the format for time zone abbreviations in this time zone. The `%s` characters are used to show where the variable part of the time zone abbreviation goes.

UNTIL Indicates the time at which the GMT offset or the rules change for a location. It is specified as year, month, day, and time of day. If this is specified, the time zone information is generated from the given GMT offset and rule change until the time specified.

The next line must be a *continuation* line. The continuation line places information starting at the time specified in the **UNTIL** field of the previous line into the file used by the previous line. This line has the same format as a zone line, except that the `Zone` string and the name are omitted. Continuation lines can contain an **UNTIL** field, just as zone lines do, indicating that the next line is a further continuation.

A link line has the form:

```
Link    LINK-FROM    LINK-TO
```

The **LINK-FROM** field should appear as the **NAME** field in a zone line; the **LINK-TO** field is used as an alternate name for that zone.

Except for continuation lines, lines can appear in any order in the input.

The **zic** command has a limitation of compiling input containing a date before 14 December 1901 because dates before this time cannot be represented by a 32-bit **time_t** data type.

Flags

-d *Directory*	Creates time conversion information files in the *Directory* directory, instead of the**/usr/share/lib/zoneinfo/** standard directory .
-l *TimeZone*	Use the *TimeZone* time zone as local time. The **zic** command acts as if the file contained a link similar to the following: `Link  timezone  localtime`
-v	Provides a message if a year that appears in a data file is outside the range of years representable by system time values (0:00:00 AM GMT, January 1, 1970, to 3:14:07 AM GMT, January 19, 2038).
-y *YearIsType*	Uses the given **yearistype** command rather than **/usr/sbin/yearistype** command when used to check year types.

Parameters

FileName	A file containing input lines that specify the time conversion information files to be created. If *FileName* is - (hyphen), then standard input is read.

Examples

1. A rule line can have the following format:

   ```
   Rule  USA  1970    max    -    Sep   Sun<=14 3:00   0     S
   ```

2. A zone line can have the following format:

   ```
   Zone    Turkey          3:00    Turkey          EET%s
   ```

3. A link line can have the following format:

   ```
   Link    MET     CET
   ```

4. To compile a **timezone.infile** file containing input time zone information and place the binaries into the standard time zone **/usr/share/lib/zoneinfo/** directory, type:

   ```
   zic timezone.infile
   ```

5. To compile a **timezone.infile** file containing input time zone information and place the binaries into a directory specified with **-d** option , type:

   ```
   zic -d tzdir timezone.infile
   ```

6. To report warnings during compilation of the time zone input file when the range of years are incorrect , type:

   ```
   zic -v timezone.infile
   ```

7. To compile a **timezone.infile** file that contains input time zone information using the **yearistype** file specified with **-y** flag to check year types, type:

   ```
   zic -y year timezone.infile
   ```

Exit Status

0 The command completed successfully.

>0 An error occurred.

Files

/usr/sbin/yearistype	Contains the **yearistype** command used to check year types.
/usr/sbin/zic	Contains the SystemV **zic** command.
/usr/share/lib/zoneinfo	Standard directory used for files create by the **zic** command.

Related Information

The **zdump** command.

Appendix A. Command Support for Files Larger than 2 Gigabytes

AIX provides support for files greater than 2 gigabytes so that users can store large quantities of data in a single file. Many, but not all, commands support the use of files larger than 2 gigabytes. Additionally, some commands have large file support with limitations.

Commands That Do Not Support Files Larger Than 2 Gigabytes

In many cases, commands that do not support large files do not utilize files of any size to begin with, such as the **date**, **echo**, **nice**, **kill** commands and others.

This support also does not extend to specific system-controlled files, such as **/etc/passwd**, **/etc/inittab**, files in **/etc/security**, system accounting files, etc. Consequently, commands that only utilize these system files, such as commands to administer users and system security (**mkuser**, **su**), system accounting commands (**acctcom**, **prdaily**), and general system controlling commands (**init**, **penable**) do not have large file support.

Other commands do not support large files because they work with files of a specific format defined to have a maximum of less than or equal to 2 gigabytes. These include the XCOFF file format, defining the format of object files and executable files. The file headers that define XCOFF do not have fields defined to support files this large, and the system would not be able to load an executable file of this size. Commands that utilize these files, such as **ld**, **as**, **m4**, **strip** and so on, do not have large file support.

The header format of the **pack**, **unpack**, and **pcat** commands does not have enough characters to store a file size over 2 gigabytes.

Additional file formats also prevent files of their type from being larger than 2 gigabytes. These include some archiving utilities restricted in format by industry standards such as the **/usr/bin/cpio** command (although the**/usr/sysv/bin/cpio** command does not have this restriction), and the object file archive format, restricting the **ar** command. You can archive large files with **backup**.

The print spooling subsystem has been enabled on the frontend to support the submission, manipulation, and cancelation of files larger than 2 gigabytes. However, the default printer backend, the **piobe** command, does not support files of this size. This means print jobs larger than 2 gigabytes can either be sent to a remote printer or print server that can handle these large files, or an alternate user or vendor-supplied backend that comprehends large files could be used.

> **Note:** A print job larger than 2 gigabytes would likely take several days to complete.

Finally, there are commands for which the user files used are not reasonably expected to ever be larger than 2 gigabytes. For example, although a directory may contain large files, the directory file itself may not exceed 2 gigabytes. Hence, commands such as **mkdir** and **rmdir** do not support large directories. Other examples in which support is unnecessary would be using the **wall** command to broadcast the contents of extremely large files to all terminals, or using the **nroff** command to process over 2 gigabytes of written text in a single file.

Appendix B. Functional List of Commands

This appendix lists commands by function.

- Communications
  - Asynchronous Terminal Emulation
  - Basic Networking Utilities
  - General Communications Facilities
  - Mail Facilities
  - Message Handler
  - Network File System
  - Network Information Service
  - Network Management
  - STREAMS
  - Transmission Control Protocol/Internet Protocol
  - Network Computing System (NCS)
- Files and Directories
  - Directories
  - Editors
  - Files
  - File Contents
  - Text Formatting
  - Text Formatting Macro Packages
- General Operations
  - Devices and Terminals
  - Documentation and Education
  - File Systems
  - Games
  - iFOR/LS
  - Logical Volumes
  - Network Installation Management (NIM)
  - Numerical Data
  - Performance Tuning
  - Processes and Commands
  - Queues
  - Screen Output
  - Security and System Access
  - Shells
  - System Accounting and Statistics
  - acct/* Commands
  - System Resources
  - Software Installation
  - User Interface
  - Macros
- Programming Tools

- Debuggers
- Messages
- Source Programs
- Object Files
- Miscellaneous Languages
- C Tools
- Assemblers and Compilers
- Object Data Manager (ODM)

Communications

Commands List: Asynchronous Terminal Emulation

ate
Starts the Asynchronous Terminal Emulation (ATE) Program.

xmodem
Transfers files with the **xmodem** protocol, which detects data transmission errors during asynchronous transmission.

Commands List: Basic Networking Utilities

ct
Dials an attached terminal and issues a login process.

cu
Connects directly or indirectly to another system.

cut
Writes out selected bytes, characters, or fields from each line of a file.

rmail
Handles remote mail received through Basic Networking Utilities (BNU).

tip
Connects to a remote system.

uucheck
Checks for files and directories required by the BNU.

uucico
Transfers Basic Networking Utilities (BNU) command, data, and execute files to remote systems.

uuclean
Removes files from the BNU spool directory.

uucleanup
Deletes selected files from the Basic Networking Utilities (BNU) spooling directory.

uucp
Copies files from one operating system to another.

uucpadm
Enters basic BNU configuration information.

uucpd
Handles communications between BNU and TCP/IP.

uudecode
Encodes or decodes a binary file for transmission using electronic mail.

uudemon.admin

	Provides periodic information on the status of BNU file transfers.
uudemon.cleanu	Cleans up BNU spooling directories and log files.
uudemon.hour	Initiates file transport calls to remote systems using the BNU program.
uudemon.poll	Polls the systems listed in the BNU **Poll** file.
uuid_gen	Generates Universal Unique Identifiers (UUIDs) for objects, types, and interfaces.
uuencode	Encodes or decodes a binary file for transmission using electronic mail.
uukick	Uses debugging mode to contact a specified remote system.
uulog	Provides information about BNU file-transfer activities on a system.
uuname	Provides information about other systems accessible to the local system.
uupick	Completes the transfer and handles files sent by the **uuto** command.
uupoll	Forces a poll of a remote BNU system.
uuq	Displays the BNU job queue and deletes specified jobs from the queue.
uusched	Schedules work for the Basic Networking Utilities (BNU) file transport program.
uusend	Sends a file to a remote host.
uusnap	Displays the status of BNU contacts with remote systems.
uustat	Reports the status of and provides limited control over BNU operations.
uuto	Copies files from one system to another.
uutry	Contacts a specified remote system with debugging turned on and allows the user to override the default retry time.
Uutry	Contacts a specified remote system with debugging turned on and saves the debugging output in a temporary file.
uux	Runs a command on another UNIX-based system.
uuxqt	Executes Basic Networking Utilities (BNU) remote command requests.

Commands List: General Communications Facilities

connect
Connects to a remote computer.

enroll
Sets up a password used to implement a secure communication channel.

getty
Sets the characteristics of ports.

mesg
Permits or refuses write messages.

no
Configures network options.

pdelay
Enables or reports the availability of delayed login ports.

pdisable
Disables login ports.

penable
Enables or reports the availability of login ports.

phold
Disables or reports the availability of login ports on hold.

pshare
Enables or reports the availability of shared login ports.

rdist
Maintains identical copies of files on multiple hosts.

rdump
Backs up files onto a remote machine's device.

wall
Writes a message to all users that are logged in.

write
Sends messages to other users on the system.

writesrv
Allows users to send messages to and receive messages from a remote system.

Commands List: Mail Facilities

bellmail
Sends messages to system users and displays messages from system users.

bffcreate
Creates installation image files in backup format.

biff
Enables or disables mail notification during the current session.

comsat
Notifies users of incoming mail.

from
Determines who mail is from.

imapd
Starts the Internet Message Access Protocol (IMAP) server process.

Mail or mail
Sends and receives mail.

mailq
Prints the contents of the mail queue.

mailstats

	Displays statistics about mail traffic.
mailx	Sends and receives mail.
newaliases	Builds a new copy of the alias database from the **/etc/aliases** file.
pop3d	Starts the Post Office Protocol Version 3 (POP3) server process.
rmail	Handles remote mail received through Basic Networking Utilities (BNU).
sendmail	Routes mail for local or network delivery.
smdemon.cleanu	Cleans up the **sendmail** queue for periodic housekeeping.
xget	Receives secret mail in a secure communication channel.
xsend	Sends secret mail in a secure communication channel.

Commands List: Message Handler

ali	Lists mail aliases and their addresses.
anno	Annotates messages.
ap	Parses and reformats addresses.
bugfiler	Automatically stores bug reports in specified mail directories.
burst	Explodes digests into messages.
comp	Composes a message.
conflict	Searches for alias and password conflicts.
dist	Redistributes a message to additional addresses.
dp	Parses and reformats dates.
folder	Selects and lists folders and messages.
folders	Lists all folders and messages in mail directory.
forw	Forwards messages.
inc	Incorporates new mail into a folder.
install_mh	Sets up mailbox directories.
mark	Creates, modifies, and displays message sequences.
mhl	Produces formatted listings of messages.
mhmail	

	Sends or receives mail.
mhpath	Prints full path names of messages and folders.
msgchk	Checks for messages.
msh	Creates an MH shell.
next	Shows the next message.
packf	Compresses the contents of a folder into a file.
pick	Selects messages by content, and creates and modifies sequences.
post	Routes a message.
prev	Shows the previous message.
prompter	Invokes a prompting editor.
rcvdist	Sends a copy of incoming messages to additional recipients.
rcvpack	Saves incoming messages in a packed file.
rcvstore	Incorporates new mail from standard input into a folder.
rcvtty	Notifies the user of incoming messages.
refile	Moves files between folders.
repl	Replies to a message.
rmf	Removes folders and the messages they contain.
rmm	Removes messages from active status.
scan	Produces a one line per message scan listing.
send	Sends a message.
sendbug	Mails a system bug report to a specified address.
show	Shows messages.
slocal	Processes incoming mail.
sortm	Sorts messages.
spost	Routes a message.
vmh	Invokes a visual interface for use with MH commands.
whatnow	Invokes a prompting interface for draft disposition.
whom	Manipulates Message Handler (MH) addresses.

Commands List: Network File System

automount	Mounts NFS file systems automatically.
biod	Handles client requests for files.
bootparamd	Provides information for booting to diskless clients.
chnfs	Changes the configuration of the system to invoke a specified number of **biod** and **nfsd** daemons.
chnfsexp	Changes the options used to export a directory to NFS clients.
chnfsmnt	Changes the options used to mount a directory from an NFS server.
exportfs	Exports and unexports directories to NFS clients.
lockd	Processes lock requests.
mknfs	Configures the system to run NFS.
mknfsexp	Exports a directory to NFS clients.
mknfsmnt	Mounts a directory from an NFS server.
mountd	Answers requests from clients for file system mounts.
nfsd	Starts client requests for file system operations.
nfso	Configures Network File System (NFS) network options.
nfsstat	Displays statistical information about the Network File System (NFS) and Remote Procedure Call (RPC) calls.
on	Executes commands on remote systems.
portmap	Converts RPC program numbers into Internet port numbers.
rexd	Executes programs for remote machines.
rmnfs	Changes the configuration of the system to stop invoking the NFS daemons.
rmnfsexp	Unexports a directory from NFS clients.
rmnfsmnt	Removes an NFS mount.
rpcgen	Generates C code to implement an RPC protocol.
rpcinfo	Reports the status of Remote Procedure Call (RPC) servers.
rpc.pcnfsd	Handles service requests from PC-NFS (Personal Computers Network File System) clients.
rstatd	Returns performance statistics obtained from the kernel.

rup

Shows the status of a remote host on the local network.

rusers

Reports a list of users logged in remote machines.

rusersd

Responds to queries from the **rusers** command.

rwall

Sends messages to all users on the network.

rwalld

Handles requests from the **rwall** command.

showmount

Displays a list of all clients that have remotely mounted file systems.

spray

Sends a specified number of packets to a host and reports performance statistics.

sprayd

Receives packets sent by the **spray** command.

statd

Provides crash and recovery functions for the locking services on NFS.

Commands List: Network Information Service

chkey

Changes your encryption key.

chmaster

Executes the **ypinit** command and restarts the NIS daemons to change a master server.

chslave

Re-executes the **ypinit** command to retrieve maps from a master server and restarts the **ypserv** daemon to change the slave server.

chypdom

Changes the current domainname of the system.

domainname

Displays or sets the name of the current NIS domain.

keyenvoy

Acts as an intermediary between user processes and the **keyserv** daemon.

keylogin

Decrypts and stores the user's secret key.

keyserv

Stores public and private keys.

lsmaster

Displays the characteristics for the configuration of an NIS master server.

lsnfsexp

Displays the characteristics of directories that are exported with the Network File System (NFS).

lsnfsmnt

Displays the characteristics of NFS mountable file systems.

makedbm

Makes a Network Information Service (NIS) map.

mkclient

Uncomments the entry in the **/etc/rc.nfs** file for the **ypbind** daemon and starts the **ypbind** daemon to configure a client.

mkkeyserv	Uncomments the entry in the **/etc/rc.nfs** file for the **keyserv** daemon and invokes the daemon by using the startsrc command.
mkmaster	Invokes the **ypinit** command and starts the NIS daemons to configure a master server.
mkslave	Executes the **ypinit** command to retrieve maps from an NIS master server and starts the **ypserv** daemon to configure a slave server.
mk_niscachemgr	Uncomments the entry in the **/etc/rc.nfs** file for the **nis_cachemgr** daemon and invokes the daemon by using the **startsrc** command.
mk_nisd	Uncomments the entry in the **/etc/rc.nfs** file for the **rpc.nisd** daemon and invokes the daemon by using the **startsrc** command.
mk_nispasswdd	Uncomments the entry in the **/etc/rc.nfs** file for the **rpc.nispasswdd** daemon and invokes the daemon by using the **startsrc** command.
newkey	Creates a new key in the **/etc/publickey** file.
nis_cachemgr	Starts the NIS+ cache manager daemon..
nisaddcred	Creates NIS+ credential information.
nisaddent	Creates NIS+ tables from corresponding **/etc** files or NIS maps.
niscat	Displays the contents of an NIS+ table.
nischgrp	Changes the group owner of a NIS+ object.
nischmod	Changes the access rights on a NIS+ object.
nischown	Changes the owner of one or more NIS+ objects or entries.
nisclient	Initializes NIS+ credentials for NIS+ principals.
nisdefaults	Displays the seven default values currently active in the namespace.
niserror	Displays NIS+ error messages.
nisgrep	Utility for searching NIS+ tables.
nisgrpadm	Creates, deletes, and performs miscellaneous administration operations on NIS+ groups.
nisinit	Initializes a workstation to be a NIS+ client.
nisln	Creates symbolic links between NIS+ objects and table entries.
nislog	The **nislog** command displays the contents of the transaction log.
nisls	Lists the contents of an NIS+ directory.
nismatch	

	Utility for searching NIS+ tables.
nismkdir	Creates non-root NIS+ directories.
nismkuser	Creates a new NIS+ user account.
nispopulate	Populates the NIS+ tables in a NIS+ domain.
nisrm	Removes NIS+ objects from the namespace.
nisrmdir	Removes NIS+ objects from the namespace.
nisrmuser	Removes a NIS+ user account.
nisserver	Sets up NIS+ servers.
nissetup	Initializes an NIS+ domain.
nisshowcache	Prints out the contents of the shared cache file.
nisstat	Reports NIS+ server statistics.
nistbladm	Administers NIS+ tables.
nistest	Returns the state of the NIS+ namespace using a conditional expression.
nisupdkeys	Updates the public keys in NIS directory objects.
revnetgroup	Reverses the listing of users and hosts in network group files in NIS maps.
rm_niscachemgr	Stops the **rpc.nisd** daemon and comments the entry in the **/etc/rc.nfs** file.
rm_nisd	Stops the **nis_cachemgr** daemon and comments the entry in the **/etc/rc.nfs** file.
rm_nispasswdd	Stops the **rpc.nispasswdd** daemon and comments the entry in the **/etc/rc.nfs** file.
rmkeyserv	Stops the **keyserv** daemon and comments the entries for the **keyserv** daemon in the **/etc/rc.nfs** file.
rmyp	Removes the configuration for NIS.
rpc.nispasswd	NIS+ password update daemon.
ypbind	Enables client processes to bind, or connect, to an NIS server.
ypcat	Prints out an NIS map.
ypinit	Sets up NIS maps on an NIS server.
ypmatch	Displays the values of given keys within an NIS map.
yppasswd	

	Changes your network password in NIS.
yppasswdd	Receives and executes requests from the **yppasswd** command.
yppoll	Displays the order number (ID number) of the NIS map currently in use on the server.
yppush	Prompts the NIS slave servers to copy updated NIS maps.
ypserv	Looks up information in local NIS maps.
ypset	Directs a client machine to a specific server.
ypupdated	Updates information in NIS maps.
ypwhich	Identifies either the NIS server or the server that is the master for a given map.
ypxfr	Transfers an NIS map from an NIS server to a local host.

Commands List: Network Management

mosy	Converts the ASN.1 definitions of Structure and Identification of Management Information (SMI) and Management Information Base (MIB) modules into objects definition files for the **snmpinfo** command.
snmpd	Starts the Simple Network Management Protocol (SNMP) agent daemon as a background process.
snmpinfo	Requests or modifies values of Management Information Base (MIB) variables managed by a Simple Network Management Protocol (SNMP) agent.

Commands List: STREAMS

autopush	Configures lists of automatically pushed STREAMS modules.
scls	Produces a list of module and driver names.
strace	Prints STREAMS trace messages.
strchg	Changes stream configuration.
strload	Loads and configures Portable Streams Environment (PSE).
strconf	Queries stream configuration.
strclean	Cleans up the STREAMS error logger.
strerr (Daemon)	Receives error log messages from the STREAMS log driver.

Commands List: Transmission Control Protocol/Internet Protocol

arp

Displays and modifies address resolution.

chnamsv

Changes TCP/IP-based name service configuration on a host.

chprtsv

Changes a print service configuration on a client or server machine.

f

Shows user information.

finger

Shows user information.

fingerd

Provides server function for **finger** command.

ftp

Transfers files between a local and a remote host.

ftpd

Provides the server function for the Internet FTP protocol.

gated

Provides gateway routing functions for the RIP, EGP, HELLO, and SNMP protocols.

gettable

Gets NIC format host tables from a host.

host

Resolves a host name into an Internet address or an Internet address into a host name.

hostent

Directly manipulates address-mapping entries in the system configuration database.

hostid

Sets or displays the identifier of the current local host.

hostname

Sets or displays the name of the current host system.

htable

Converts host files to the format used by network library routines.

ifconfig

Configures or displays network interface parameters for a network that is using TCP/IP.

inetd

Provides Internet service management for a network.

ipreport

Generates a packet trace report from the specified packet trace file.

iptrace

Provides interface-level packet tracing for Internet protocols.

lpd

Provides the remote print server on a network.

lsnamsv

Shows name service information stored in the database.

lsprtsv

Shows print service information stored in the database.

mkhosts

Generates the host table file.

mknamsv

Configures TCP/IP-based name service on a host for a client.

mkprtsv

Configures TCP/IP-based print service on a host.

mktcpip	Sets the required values for starting TCP/IP on a host.
named	Provides the server function for the Domain Name Protocol.
namerslv	Directly manipulates domain name server entries for local resolver routines in the system configuration database.
netstat	Shows network status.
nslookup	Queries Internet domain name servers.
ping	Sends an echo request to a network host.
rcp	Transfers files between a local and a remote host or between two remote hosts.
remsh	Executes the specified command at the remote host or logs into the remote host.
rexec	Executes commands one at a time on a remote host.
rexecd	Provides the server function for the **rexec** command.
rlogin	Connects the local host with a remote host.
rlogind	Provides the server function for the **rlogin** command.
rmnamsv	Unconfigures TCP/IP-based name service on a host.
rmprtsv	Unconfigures a print service on a client or server machine.
route	Manually manipulates the routing tables.
routed	Manages network routing tables.
rsh	Executes the specified command at the remote host or logs into the remote host.
rshd	Provides the server function for remote command execution.
ruptime	Shows the status of each host on a network.
ruser	Directly manipulates entries in three separate system databases that control foreign host access to programs
rwho	Shows which users are logged in to hosts on the local network.
rwhod	Provides the server function for the **rwho** and **ruptime** commands.
securetcpip	Enables the operating system network security feature.
setclock	Sets the time and date for a host on a network.
slattach	Attaches serial lines as network interfaces.
sliplogin	

Configures a standard-input terminal line as a Serial Line Internet Protocol (SLIP) link to a remote host.

talk

Converse with another user.

talkd

Provides the server function for the **talk** command.

tcpdump

Prints out packet headers.

telinit

Initializes and controls processes.

telnet

Connects the local host with a remote host using the TELNET interface.

telnetd

Provides the server function for the TELNET protocol.

tftp

Transfers files between hosts using the Trivial File Transfer Protocol (TFTP).

tftpd

Provides the server function for the Trivial File Transfer Protocol.

timed

Invokes the time server daemon at system startup time.

timedc

Returns information about the **timed** daemon.

tn

Connects the local host with a remote host using the TELNET interface.

tn3270

Connects the local host with a remote host using the TELNET interface.

traceroute

Prints the route that IP packets take to a network host.

trpt

Performs protocol tracing on TCP sockets.

utftp

Transfers files between hosts using the Trivial File Transfer Protocol (TFTP).

Commands List: Network Computing System (NCS)

lb_admin Monitors and administers Location Broker registrations.
llbd Manages the information in the Local Location Broker database.
nrglbd Manages the Global Location Broker database.

Files and Directories

Commands List: Directories

cd

Changes the current directory.

chgrp

Changes the group ownership of a file or directory.

chmod

	Changes permission modes.
chroot	Changes the root directory of a command.
delete	Removes (unlinks) files or directories.
dircmp	Compares two directories and the contents of their common files.
dirname	Writes to standard output all but the last part of a specified path.
dosdir	Lists the directory for DOS files.
fdformat	Formats diskettes.
ls	Displays the contents of a directory.
mkdir	Creates one or more new directories.
mvdir	Moves (renames) a directory.
pathchk	Checks pathnames.
pwd	Displays the pathname of the working directory.
rm	Removes (unlinks) files or directories.
rmdir	Removes a directory.
which_fileset	Searches the **/usr/lpp/bos/AIX_file_list** file for a specified file name or command.

Commands List: Editors

ctags	Makes a file of tags to help locate objects in source files.
ed	Edits text by line.
edit	Provides a simple line editor for the new user.
ex	Edits lines interactively, with a screen display.
red	Edits text by line.
sed	Provides a stream editor.
tvi	Provides a trusted editor with a full-screen display.
vedit	Edits files with a full-screen display.
vi	Edits files with a full-screen display.
view	Starts the vi editor in read-only mode.

Commands List: Files

ar

Maintains the indexed libraries used by the linkage editor.

backup

Backs up files and filesystems.

cat

Concatenates or displays files.

chgrp

Changes the group ownership of a file or directory.

chlang

Changes the language (LANG) environment variable in the **/etc/environment** file.

chmod

Changes permission modes.

chtz

Changes the language (TZ) environment variable in the **/etc/profile** file.

cksum

Changes the checksum and byte count of a file.

copy

Copies files.

cp

Copies files.

cpio

Copies files into and out of archive storage and directories.

dd

Converts and copies a file.

defragfs

Increases a file system's contiguous free space.

delete

Removes (unlinks) files or directories.

dosdel

Deletes DOS Files.

dosread

Copies DOS files.

doswrite

Copiesthis operating system files to DOS files.

file

Determines the file type.

find

Finds files with a matching expression.

link

Performs a **link** subroutine.

ln

Links files.

mv

Moves files.

nulladm

Creates the file specified with read and write permissions to the file owner and group and read permissions to other users.

pax

Extracts, writes, and lists members of archive files; copies files and directory hierarchies.

pg

Formats files to the display.

restore

Copies previously backed-up file systems or files, created by the **backup** command, from a local device.

rm

Removes (unlinks) files or directories.

rmvfs

Removes entries in the **/etc/vfs** file.

split

Splits a file into pieces.

sum

Displays the checksum and block count of a file.

tar

Manipulates archives.

tee

Displays the output of a program and copies it into a file.

touch

Updates the access and modification times of a file.

umask

Displays or sets the file mode creation mask.

unlink

Performs an **unlink** subroutine.

Commands List: File Contents

awk

Finds lines in files matching patterns and then performs specified actions on them.

bdiff

Uses the **diff** command to find differences in very large files.

bfs

Scans files.

cmp

Compares two files.

colrm

Extracts columns from a file.

comm

Selects or rejects lines common to two sorted files.

comp

Composes a message.

compress

Compresses and expands data.

csplit

Splits files by context.

cut

Writes out selected bytes, characters, or fields from each line of a file.

diff

Compares text files.

diff3

Compares three files.

dircmp

Compares two directories and the contents of their common files.

egrep

Searches a file for a pattern.

expand

	Writes to standard output with tabs changed to spaces.
fgrep	Searches a file for a literal string.
fold	Folds long lines for finite-width output device.
genxlt	Generates a code set conversion table for use by the **Iconv** library.
grep	Searches a file for a pattern.
head	Display the first few lines or bytes of a file or files.
iconv	Converts the encoding of characters from one code page encoding scheme to another.
join	Joins the data fields of two files.
localedef	Processes locales and character map files to produce a locale database.
look	Finds lines in a sorted file.
more	Displays continuous text one screen at a time on a display screen.
paste	Merges the lines of several files or subsequent lines in one file.
pcat	Unpacks files and writes them to standard output.
pack	Compresses files.
page	Displays continuous text one screen at a time on a display screen.
rev	Reverse characters in each line of a file.
sdiff	Compares two files and displays the differences in a side-by-side format.
sort	Sorts files, merges files that are already sorted, and checks files to determine if they have been sorted.
spell	Finds English-language spelling errors.
spellin	Creates a spelling list.
spellout	Verifies that a word is not in the spelling list.
tab	Changes spaces into tabs.
tail	Writes a file to standard output, beginning at a specified point.
tr	Translates characters.
trbsd	Translates characters (BSD version).
tsort	Sorts an unordered list of ordered pairs (a topological sort).
uncompress	

	Compresses and expands data.
unexpand	Writes to standard output with tabs restored.
uniq	Deletes repeated lines in a file.
unpack	Expands files.
untab	Changes tabs into spaces.
wc	Counts the number of lines, words, and bytes in a file.
what	Displays identifying information in files.
zcat	Compresses and expands data.

Commands List: Text Formatting

addbib	Creates or extends a bibliographic database.
apropos	Locates commands by keyword lookup.
canonls	Processes **troff** command output for the Canon LASER SHOT in LIPS III mode.
catman	Creates the cat files for the manual.
checkcw	Prepares constant-width text for the **troff** command.
checkeq	Checks documents formatted with memorandum macros.
checkmm	Checks documents formatted with memorandum macros.
checknr	Checks **nroff** and **troff** files.
col	Filters for standard output text having reverse linefeeds and forward/reverse half-linefeeds.
colcrt	Filters **nroff** command output for CRT previewing.
cw	Prepares constant-width text for the **troff** command.
deroff	Removes **nroff**, **troff**, **tbl**, and **eqn** command constructs from files.
diction	Highlights unclear or wordy sentences.
diffmk	Marks differences between files.
enscript	Converts text files to PostScript format for printing.
eqn	Formats mathematical text for the **troff** command.
expand	Writes to standard output with tabs changed to spaces.
explain	

Provides an interactive thesaurus.

fmt

Formats mail messages prior to sending.

grap

Typesets graphs to be processed by the **pic** command.

greek

Converts English-language output from a Teletype 37 workstation to output for other workstations.

hp

Handles special functions for the HP2640- and HP2621-series terminals.

hplj

Post-processes the **troff** command output for the HP LaserJet Series printers.

hyphen

Finds hyphenated words.

ibm3812

Post-processes the **troff** command output for the 3818 Pageprinter and the 3812 Model 2 Pageprinter.

ibm3816

Post-processes the **troff** command output for the 3816 Pageprinter and the 3812 Model 2 Pageprinter.

ibm5587G

Post-processes **troff** command output for the 5587G printer with the (32x32/24x24) cartridge installed.

indxbib

Builds an inverted index for a bibliography.

lookbib

Finds references in a bibliography.

macref

Produces cross-reference listing of macro files.

makedev

Creates binary description files suitable for reading by the **troff** command and its preprocessors.

managefonts

Provides the user with a simple menu-based interface to update or change the set of installed font families on the system.

mant

Typesets manual pages.

mm

Prints documents formatted with memorandum macros.

mmt

Typesets documents.

mvt

Typesets English-language view graphs and slides.

ndx

Creates a subject-page index for a document.

neqn

Formats mathematical text for the **nroff** command.

newform

Changes the format of a text file.

nl

Numbers lines in a file.

nroff

Formats text for printing on typewriter-like devices and line printers.

pic

	Preprocesses **troff** command input for the purpose of drawing pictures.
proff	Formats text for printers with personal printer data streams.
ps630	Converts Diablo 630 print files to PostScript format.
ps4014	Converts a Tektronix 4014 files to PostScript format.
psc	Converts **troff** intermediate format to PostScript format.
psdit	Converts **troff** intermediate format to PostScript format.
psplot	Converts files in plot format to PostScript format.
psrev	Reverses the page order of a PostScript file and selects a page range for printing.
psroff	Converts files from **troff** format to PostScript format.
ptx	Generates a permuted index.
refer	Finds and inserts literature references in documents.
roffbib	Prints a bibliographic database.
soelim	Processes **.so** requests in **nroff** command files.
sortbib	Sorts a bibliographic database.
spell	Finds English-language spelling errors.
spellin	Creates a spelling list.
spellout	Verifies that a word is not in the spelling list.
style	Analyzes surface characteristics of a document.
subj	Generates a list of subjects from a document.
tbl	Formats tables for the **nroff** and **troff** commands.
tc	Interprets text in the **troff** command output for the Tektronix 4015 system.
troff	Formats text for printing on typesetting devices.
ul	Performs underlining.
vgrind	Formats listings of programs that are easy to read.
xpreview	Displays **troff** files on an X display.

Text Formatting Macro Packages

man

Provides a formatting facility for manual pages.

me

Provides a formatting facility for creating technical papers in various styles.

mm

Provides a formatting facility for business documents such as memos, letters, and reports.

mptx

Formats a permuted index produced by the **ptx** command.

ms

Provides a formatting facility for various styles of articles, theses, and books.

mv

Simplifies typesetting of view graphs and projection slides.

General Operations

Commands List: Devices and Terminals

adfutil

Provides the capability to merge Micro Channel information through AIX 5.1 only for PS/2 adapters with the Configuration Database.

bterm

Emulates terminals in bidirectional bus (BIDI) mode.

cancel

Cancels requests to a line printer.

captoinfo

Converts a termcap file to a terminfo descriptor file.

cfgmgr

Configures devices by running the programs specified in the Configuration Rules object class.

chcons

Redirects the system console to a specified device or file to be effective on the next start of the system.

chdev

Changes the characteristics of a device.

chdisp

Changes the display used by the low function terminal (LFT) subsystem .

chfont

Changes the default font for a display.

chkbd

Changes the default keyboard map used by the high function terminal Subsystem at system startup.

clear

Clears the terminal screen.

devnm

Names a device.

diag

Performs hardware problem determination.

digest

	Converts the ASCII form of the **/etc/qconfig** file into the **/etc/qconfig.bin** file, a binary version of the queue configuration used by the **qdaemon** command.
dscreen	Starts the Dynamic Screen utility.
enable	Enables a printer queue
fdformat	Formats diskettes.
flcopy	Copies to and from diskettes.
fold	Folds long lines for finite-width output device.
format	Formats diskettes.
getty	Sets the characteristics of ports.
hplj	Post-processes the **troff** command output for the HP LaserJet Series printers.
ibm3812	Post-processes the **troff** command output for the 3816 Pageprinter and the 3812 Model 2 Pageprinter.
ibm3816	Post-processes the **troff** command output for the 3816 Pageprinter and the 3812 Model 2 Pageprinter.
ibm5587G	Post-processes **troff** command output for the 5587G printer with the (32x32/24x24) cartridge installed.
iconv	Converts the encoding of characters from one code page encoding scheme to another.
infocmp	Manages **terminfo** descriptions.
iostat	Reports Central Processing Unit (CPU) statistics and input/output statistics for tty, disks, and CD-ROMs.
keycomp	Compiles a keyboard mapping file into an input method keymap file.
lp	Sends requests to a line printer.
lpr	Enqueues print jobs.
lpstat	Displays line printer status information.
lptest	Generates the line printer ripple pattern.
lsattr	Displays attribute characteristics and possible values of attributes for devices in the system.
lscfg	Displays diagnostic information about a device.
lsconn	Displays the connections a given device, or kind of device, can accept.
lscons	

	Writes the name of the console device to standard output.
lsdev	Displays devices in the system and their characteristics.
lsdisp	Lists the displays currently available on the system.
lsfont	Lists the fonts available for use by the display.
lskbd	Lists the keyboard maps currently available to the Low Function Terminal (LFT) subsystem.
lsparent	Displays the possible parent devices that accept a specified connection type or device.
mkdev	Adds a device to the system.
mkfont	Adds the font code associated with a display to the system.
mknod	Creates a special file.
mt (BSD)	Gives subcommands to streaming tape device.
panel20	Diagnoses activity between an HIA and the 5080 Control Unit.
pdelay	Enables or reports the availability of delayed login ports.
pdisable	Disables login ports.
penable	Enables or reports the availability of login ports.
phold	Disables or reports the availability of login ports on hold.
pioattred	Provides a way to format and edit attributes in a virtual printer.
piobe	Print job manager for the printer backend.
pioburst	Generates burst pages (header and trailer pages) for printer output.
piocnvt	Expands or contracts a predefined definition or virtual printer definition.
piodigest	Digests attribute values for a virtual printer definition into memory image and stores the memory image in a file.
piofontin	Copies fonts from a multilingual font diskette.
pioformat	Drives a printer formatter.
piofquote	Converts certain control characters destined for PostScript printers.
pioout	Printer backend's device driver interface program.
piopredef	Creates a predefined printer data stream definition.
portmir	

	Allows one TTY stream (monitor) to attach to another TTY stream (target) and monitor the user session that is taking place on that stream.
pr	Writes a file to standard output.
pshare	Enables or reports the availability of shared login ports.
pstart	Enables or reports the availability of login ports (normal, shared, and delayed).
pstat	Interprets the contents of the various system tables and writes it to standard output.
reset	Initializes terminals.
rmdev	Removes a device from the system.
rmt	Allows remote access to magnetic tape devices.
script	Makes a typescript of a terminal session.
setmaps	Sets terminal maps or code setmaps.
splp	Changes or displays printer driver settings.
stty	Sets, resets, and reports workstation operating parameters.
stty-cxma	Sends and reports the terminal options for 128-port asynchronous controllers.
swapon	Specifies additional devices for paging and swapping.
swcons	Redirects, temporarily, the system console output to a specified device or file.
sysdumpdev	Changes the primary or secondary dump device designation in a running system.
tabs	Sets tab stops on terminals.
tapechk	Performs consistency checking of the streaming tape device.
tcopy	Copies a magnetic tape.
tctl	Gives commands to a streaming tape device.
termdef	Queries terminal characteristics.
tput	Queries the terminal descriptor files in the **terminfo** database.
tset	Initializes terminals.
tsm	Provides terminal state management.
tty	Writes to standard output the full pathname of your terminal.

Commands List: Documentation and Education

apropos
Locates commands by keyword lookup.

catman
Creates the cat files for the manual.

explain
Provides an interactive thesaurus.

help
Provides information for new users.

learn
Provides computer-aided instruction courses and practice for using files, editors, macros, and other features.

man
Displays manual entries online.

Commands List: File Systems

automount
Mounts NFS file systems automatically.

chfs
Changes attributes of a file system.

chps
Changes attributes of a paging space.

chvfs
Changes entries in the **/etc/vfs** file.

crfs
Adds a file system.

crvfs
Creates entries in the **/etc/vfs** file.

defragfs
Increases a file system's contiguous free space.

df
Reports information about space on file systems.

dfsck
Checks file system consistency and interactively repairs the file system.

dosformat
Formats a DOS diskette.

dumpfs
Dumps file system information.

ff
Lists the file names and statistics for a file system.

fsck
Checks file system consistency and interactively repairs the file system.

fsdb
Debugs file systems.

istat
Examines i-node numbers.

lsfs
Displays the characteristics of file systems.

mkfs

	Makes a file system.
mklost+found	Creates a lost and found directory for the **fsck** command.
mkproto	Constructs a prototype file system.
mount	Makes a file system available for use.
ncheck	Generates path names from i-node numbers.
proto	Constructs a prototype file for a file system.
rmfs	Removes a file system, any logical volume on which it resides, and the associated stanza in the **/etc/filesystems** file.
rrestore	Copies previously backed up file systems from a remote machine's device to the local machine.
skulker	Cleans up file systems by removing unwanted files.
umount	Unmounts a previously mounted file system, directory, or file.
unmount	Unmounts a previously mounted file system, directory, or file.
update	Periodically updates the super block.

Commands List: Games

arithmetic	Tests arithmetic skills.
bj	Starts the blackjack game.
craps	Starts the craps game.
fish	Plays the go fish card game.
fortune	Displays a random fortune from a database of fortunes.
hangman	Starts the hangman word-guessing game.
moo	Starts the number-guessing game.
number	Displays the written form of a number.
quiz	Tests your knowledge.
ttt	Starts the tic-tac-toe game.
turnoff	Sets the permission codes off for files in the /**usr**/**games** directory.
turnon	Sets the permission codes on for the files in the /**usr**/**games** directory.
wump	Starts the hunt the wumpus game.

Commands List: License Use Management

drm_admin

Administers servers based on the Data Replication Manager (DRM), such as **glbd**, the replicated version of the global location broker (GLB).

glbd

Manages the global location broker database.

lb_admin

Monitors and administers Location Broker registrations.

lb_find

Gets a list of global location broker (GLB) server daemons and their attributes.

llbd

Manages the information in the Local Location Broker database.

monitord

Communicates with the License Use Management server and requests an AIX Version 4 concurrent-use license for each countable login.

nrglbd

Manages the Global Location Broker database.

Commands List: Logical Volumes

chlv

Changes only the characteristics of a logical volume.

chpv

Changes the characteristics of a physical volume in a volume group.

chvg

Sets the characteristics of a volume group.

cplv

Copies the contents of a logical volume to a new logical volume.

exportvg

Exports the definition of a volume group from a set of physical volumes.

extendlv

Increases the size of a logical volume by adding unallocated physical partitions from within the volume group.

extendvg

Adds physical volumes to a volume group.

importvg

Imports a new volume group definition from a set of physical volumes.

lslv

Displays information about a logical volume.

lspv

Displays information about a physical volume within a volume group.

lsvg

Displays information about volume groups.

migratepv

Moves allocated physical partitions from one physical volume to one or more other physical volumes.

mirrorvg

Mirrors all the logical volumes that exist on a given volume group.

mklv

Creates a logical volume.

mklvcopy

Provides copies of data within the logical volume.

mkvg

Creates a volume group.

mkvgdata

Creates a file containing information about a volume group for use by the **savevg** and **restvg** commands.

redefinevg

Redefines the set of physical volumes of the given volume group in the device configuration database.

reducevg

Removes physical volumes from a volume group.

reorgvg

Reorganizes the physical partition allocation for a volume group.

restvg

Restores the user volume group and all it containers and files, as specified in the **/tmp/vgdata/vgname/vgname.data** file contained within the backup image created by the **savevg** command.

rmlv

Removes logical volumes from a volume group.

rmlvcopy

Removes copies from a logical volume.

savevg

Finds and backs up all file belonging to a specified volume group.

synclvodm

Synchronizes or rebuilds the logical volume control block, the device configuration database, and the volume group descriptor areas on the physical volumes.

syncvg

Synchronizes logical volume copies that are not current.

unmirrorvg

Removes the mirrors that exist on volume groups or specified disks.

varyoffvg

Deactivates a volume group.

varyonvg

Activates a volume group.

Commands List: Network Installation Management (NIM)

lsnim

Displays information about the Network Installation Management (NIM) environment.

nim

Performs operations on Network Installation Management (NIM) objects.

nimclient

Allows Network Installation Management (NIM) operations to be performed from a NIM client.

nimconfig

Initializes the Network Installation Management (NIM) client package.

niminit

Displays information about the Network Installation Management (NIM) environment.

Commands List: Numerical Data

bc

Provides an interpreter for arbitrary-precision arithmetic language.

dc

Provides an interactive desk calculator for doing arbitrary-precision integer arithmetic.

factor

Factors a number.

number

Displays the written form of a number.

units

Converts units in one measure to equivalent units in another measure.

Commands List: Performance Tuning

acctcms	Produces command usage summaries from accounting records.
acctcom	Displays selected process accounting record summaries.
accton	Performs process-accounting procedures.
filemon	Monitors and reports performance of file system.
fileplace	Displays the placement of file's blocks within logical or physical volumes.
gprof	Displays call graph profile data.
iostat	Reports Central Processing Unit (CPU) statistics and input/output statistics for tty, disks, and CD-ROMs.
lsattr	Displays attribute characteristics and possible values of attributes for devices in the system.
lslv	Displays information about a logical volume.
mmtu	Displaying, adding, and deleting maximum transfer unit (MTU) values used for path MTU discovery.
netpmon	Monitors activity and reports statistics on network usage.
netstat	Shows network status.
nfsstat	Displays statistical information about the Network File System (NFS) and Remote Procedure Call (RPC) calls.
nice	Runs a command at a specified priority.
no	Configures network options.
nulladm	Creates the file specified with read and write permissions to the file owner and group and read permissions to other users.
ps	Shows current status of processes.
renice	Alters priority of running processes.
reorgvg	Reorganizes the physical partition allocation for a volume group.
rmss	Simulates system with various sizes of real memory.
sar	Collects, reports, or saves system activity information.
stripnm	Displays the symbol information of a specified object file.
svmon	Captures and analyzes a snapshot of virtual memory.
time	Prints the time of the execution of a command.
timex	Reports, in seconds, the elapsed time, user time, and system execution time for a command.
tprof	Specifies the user program to be profiled, executes it, and produces reports.
trcnm	Generates a kernel name list.
trcrpt	Formats a report from the trace log.
trcstop	Stops the trace function.
vmstat	Reports virtual memory statistics.

Commands List: Processes and Commands

apply Applies a command to a set of parameters.
cron Runs commands automatically.
cronadm Lists or removes **crontab** or **at** jobs.
crontab Submits, lists, or removes **cron** job files.
env Displays the current environment or sets the environment for the execution of a command.
fuser Identifies processes using a file or file structure.
install Installs a command.
installbsd Installs a command (BSD version of the **install** command).
ipcs Reports interprocess communication facility status.
kill Sends a signal to running processes.
killall Cancels all processes except the calling process.
lastcomm Displays information about the last commands executed.
nice Runs a command at a specified priority.
nohup Runs a command without hangups.
ps Shows current status of processes.
renice Alters priority of running processes.
sleep Suspends execution for an interval.
time Prints the time of the execution of a command.
timex Reports, in seconds, the elapsed time, user time, and system execution time for a command.
wait Waits until the termination of a process ID.
whatis Describes what function a command performs.
xargs Constructs parameter lists and runs commands.

Commands List: Queues

at Runs commands at a later time.
atq Displays the queue of jobs waiting to be run.
atrm Removes jobs spooled by the **at** command.
batch Runs jobs when the system load level permits.
chprtsv Changes a print service configuration on a client or server machine.
chque Changes the queue name.
chquedev Changes the printer or plotter queue device names.
chvirprt Changes the attribute values of a virtual printer.
digest Converts the ASCII form of the **/etc/qconfig** file into the **/etc/qconfig.bin** file, a binary version of the queue configuration used by the **qdaemon** command.
disable Disables a printer queue.
enq Enqueues a file.
lpq Examines the spool queue.
lpr Enqueues print jobs.
lprm Removes jobs from the line printer spooling queue.
lsallq Lists the names of all configured queues.
lsallqdev Lists all configured printer and plotter queue device names within a specified queue.
lsprtsv Shows print service information stored in the database.
lsque Displays the queue stanza name.
lsquedev Displays the device stanza name.
lsvirprt Displays the attribute values of a virtual printer.
mkprtsv Configures TCP/IP-based print service on a host.
mkque Adds a printer queue to the system.
mkquedev Adds a printer queue device to the system.
mkvirprt Makes a virtual printer.
piodmgr Compacts the Object Data Manager (ODM) database in the **/var/spool/lpd/pio/@local/smit** directory.
piolsvp Lists virtual printers on a system.

piomgpdev Manages printer pseudo-devices.
piomkapqd Builds a SMIT dialog to create print queues and printers.
piomkpq Creates a printer queue.
piomsg Sends a printer backend message to the user.
qadm Performs system administration functions for the print spooling system.
qcan Cancels a print job.
qchk Displays the status of a print queue.
qdaemon Schedules jobs enqueued by the **enq** command.
qhld Holds a spooled print job.
qmov Moves spooled print jobs to another queue.
qpri Prioritizes a job in the print queue.
qprt Starts a print job.
qstatus Provides printer status for the printer spooling system.
rembak Sends a print job to a queue on a remote server.
rmprtsv Unconfigures a print service on a client or server machine.
rmque Removes a printer queue from the system.
rmquedev Removes a printer or plotter queue device from the system.
rmvirprt Removes a virtual printer.

Commands List: Screen Output

banner Writes ASCII character strings in large letters to standard output.
cal Displays a calendar.
calendar Writes reminder messages to standard output.
echo Writes character strings to standard output.
leave Reminds you when you have to leave.
more Displays continuous text one screen at a time on a display screen.
news Writes system news items to standard output.
page Displays continuous text one screen at a time on a display screen.
tail Writes a file to standard output, beginning at a specified point.
vacation Returns a message to the sender that the mail recipient is on vacation.

Commands List: Security and System Access

acledit Edits the access control information of a file.
aclget Displays the access control information of a file.
aclput Sets the access control information of a file.
audit Controls system auditing.
auditbin Manages bins of audit information.
auditcat Writes bins of audit records.
auditpr Formats bin or stream audit records to a display device or printer.
auditselect Selects audit records for analysis according to defined criteria.
auditstream Creates a channel for reading audit records.
chfn Changes a user's gecos information.
chgroup Changes attributes for groups.
chgrp Changes the group ownership of a file or directory.
chgrpmem Changes the administrators or members of a group.
chmod Changes permission modes.
chown Changes the user associated with a file.
chrole Changes role attributes.
chsec Changes attributes in the security stanza files.
chsh Changes a user's login shell.
chtcb Changes or queries the **trusted computing base** attribute of a file.
chuser Changes attributes for the specified user.

groups Displays group membership.
grpck Verifies the correctness of a group definition.
last Displays information about previous logins.
lastlogin Updates the **/var/adm/acct/sum/loginlog** file to show the last date each user logged in.
lssec Lists the attributes in the security stanza files.
lock Reserves a terminal.
login Initiates a user session.
logname Displays login name.
logout Stops all processes on a port.
lsgroup Displays the attributes of groups.
lslicense Displays the maximum number of users that can be logged in concurrently.
lsrole Displays role attributes.
lsuser Displays attributes of user accounts.
makekey Generates an encryption key.
mkgroup Creates a new group.
mkpasswd Creates a hashed look-aside version of the user database.
mkrole Creates new roles.
mkuser Creates a new user account.
mkuser.sys Customizes a new user account.
newgrp Changes your primary group identification.
nulladm Creates active accounting data files.
passwd Changes a user's password.
pwdadm Administers users' passwords.
pwdck Verifies the correctness of local authentication information.
rmgroup Removes a group.
rmrole Removes a role.
rmuser Removes a user account.
Rsh Invokes the restricted version of the Bourne shell.
setgroups Resets the supplementary group ID for the session.
setsenv Resets the protected state environment of a user.
shell Executes a shell with the user's default credentials and environment.
su Changes the user ID associated with a session.
sysck Checks the inventory information during installation and update procedures.
tcbck Audits the security state of the system.
usrck Verifies the correctness of a user definition.
xss Improves the security of unattended workstations.

Commands List: Shells

alias Defines or displays aliases.
basename Returns the base file name of a string parameter.
bg Runs jobs in the background.
bsh Invokes the Bourne shell.
chsh Changes a user's login shell.
command Executes a simple command.
csh Invokes the C shell.
expr Evaluates arguments as expressions.
false Returns an exit value of zero **(true)** or a nonzero exit value (**false**).
fc Processes the command history list.
fg Runs jobs in the foreground.
getopt Parses command line flags and parameters.
hash Remembers or reports command path names.
jobs Displays status of jobs in the current session.
ksh Invokes the Korn shell.

line	Reads one line from the standard input.
patch	Applies changes to files.
read	Reads one line from standard input.
rsh	Executes the specified command at the remote host or logs into the remote host.
Rsh	Invokes the restricted version of the Bourne shell.
sh	Invokes the default shell.
shell	Executes a shell with the user's default credentials and environment.
tee	Displays the output of a program and copies it into a file.
test	Evaluates conditional expressions.
true	Returns an exit value of zero (**true**) or a nonzero exit value (**false**).
tsh	Interprets commands in a trusted shell.
type	Writes a description of the command type.
ulimit	Sets or reports user resource limits.
unalias	Removes alias definitions.
xargs	Constructs argument lists and runs commands.
yes	Outputs an affirmative response repetitively.

Commands List: System Accounting and Statistics

accton	Performs process-accounting procedures.
date	Displays or sets the date or time.
diag	Performs hardware problem determination.
dp	Parses and reformats dates.
du	Summarizes disk usage.
dump	Dumps selected parts of an object file.
errclear	Deletes entries from the error log.
errdead	Extracts error records from a system dump.
errdemon	Starts the error-logging daemon and writes entries to the error log.
errinstall	Installs messages in the error logging message sets.
errlogger	Logs an operator message.
errmsg	Adds a message to the error logging message catalog.
errpt	Processes a report of logged errors.
errstop	Terminates the error-logging daemon.
errupdate	Updates the Error Record Template Repository.
getconf	Writes system configuration variable values to standard output.
id	Displays the system identifications of a specified user.
iostat	Reports Central Processing Unit (CPU) statistics and input/output statistics for tty, disks, and CD-ROMs.
ipcs	Reports interprocess communication facility status.
ipreport	Generates a packet trace report from the specified packet trace file.
iptrace	Provides interface-level packet tracing for Internet protocols.
last	Displays information about previous logins.
locale	Writes information about current locale or all public locales.
logger	Makes entries in the system log.
pac	Prepares printer/plotter accounting records.
pstat	Interprets the contents of the various system tables and writes it to standard output.
sa	Summarizes accounting records.
sa1	Collects and stores binary data in the **/var/adm/sa/sa***dd* file.
sa2	Writes a daily report in the **/var/adm/sa/sar***dd* file.
sadc	Provides a system activity report package.
sar	Collects, reports, or saves system activity information.
snap	Gathers system configuration information.
sysdumpstart	Provides a command line interface to start a kernel dump to the primary or secondary dump device.
sysline	Displays system status on the status line of a terminal.

syslogd	Logs system messages.
tput	Queries the terminal descriptor files in the **terminfo** database,
uname	Displays the name of the current operating system.
uptime	Shows how long the system has been up.
users (BSD)	Displays a compact list of users currently on the system.
vmstat	Reports virtual memory statistics.
w	Prints a summary of current system activity.
watch	Observes a program that may be untrustworthy.
who	Identifies the users currently logged in.
whoami	Displays your login name.
whois	Identifies a user by user ID or alias.

acct/* Commands

ac	Prints connect-time records.
acctcms	Produces command usage summaries from accounting records.
acctcom	Displays selected process accounting record summaries.
acctcon1	Performs connect-time accounting.
acctcon2	Performs connect-time accounting.
acctdisk	Performs disk-usage accounting.
acctdusg	Performs disk-usage accounting.
acctmerg	Merges total accounting files into an intermediary file or a daily report.
acctprc1	Performs process-accounting procedures.
acctprc2	Performs process-accounting procedures.
accton	Performs process-accounting procedures.
acctwtmp	Manipulates connect-time accounting records to change formats and to make corrections in the records.
chargefee	Charges users for the computer resources they use.
ckpacct	Checks data file size for process accounting.
diskusg	Generates disk accounting data by user ID.
dodisk	Initiates disk-usage accounting.
fwtmp	Manipulates connect-time accounting records to change formats and to make corrections in the records.
lastlogin	Reports the last login date for each user on the system.
monacct	Performs monthly or periodic accounting.
nulladm	Creates active accounting data files.
prctmp	Displays session record files.
prdaily	Creates an ASCII report of the previous day's accounting data.
prtacct	Formats and displays files in **tacct** format.
remove	Deletes files from **var/adm/acct** subdirectories.
runacct	Runs daily accounting.
shutacct	Turns off processing accounting.
startup	Turns on accounting functions at system startup.
turnacct	Provides an interface to the **accton** command to turn process accounting on or off.
wtmpfix	Manipulates connect-time accounting records to change formats and to make corrections in the records.

Commands List: System Resources

chps	Changes attributes of a paging space.
chserver	Changes a subserver definition in the subserver object class.
chssys	Changes a subsystem definition in the subsystem object class.
compress	Compresses and expands data.
lslicense	Displays the range of users that can be logged in concurrently.

lsps	Displays the characteristics of paging spaces.
lssrc	Gets status of a subsystem, a group of subsystems, or a subserver.
mknotify	Adds a notify method definition to the Notify object class.
mkps	Add an additional paging space to the system.
mkserver	Adds a subserver definition to the subserver object class.
mkssys	Adds a subsystem definition to the subsystem object class.
pack	Compresses files.
pagesize	Displays the system page size.
pcat	Unpacks files and writes them to standard output.
rmnotify	Removes a notify method definition from the Notify object class.
rmps	Removes a paging space from the system along with any logical volume on which it resides.
rmserver	Removes a subserver definition from the Subserver Type object class.
rmssys	Removes a subsystem definition from the subsystem object class.
srcmstr	Starts the System Resource Controller.
startsrc	Starts a subsystem, a group of subsystems, or a subserver.
stopsrc	Stops a subsystem, a group of subsystems, or a subserver.
swapon	Specifies additional devices for paging and swapping.
tracesoff	Turns off tracing of a subsystem, a group of subsystems, or a subserver.
traceson	Turns on tracing of a subsystem, a group of subsystems, or subserver.
uncompress	Compresses and expands data.
zcat	Compresses and expands data.

Commands List: Software Installation

bootlist	Alters the list of IPL devices or the ordering of devices on the list) available to the system.
bootparamd	Provides information for booting to diskless clients.
bosboot	Creates boot device.
chitab	Changes records in the **/etc/inittab** file.
ckprereq	Verifies that all prerequisite software is available and at appropriate revision levels.
fastboot	Restarts the system.
fasthalt	Stops the processor.
halt	Stops the processor.
init	Initializes and controls processes.
installp	Installs available software products in a compatible installation package.
inudocm	Displays contents of files containing supplemental information.
inurecv	Recovers all files saved by the **inusave** command.
inurest	Performs simple archive and restore operations for the **installp** command and shell scripts.
inusave	Saves files that are installed or updated during an installation procedure.
inuumsg	Displays specific error or diagnostic messages provided by a software products installation procedures.
logger	Make entries in the system log.
lppchk	Verifies files of an installable software product.
lsitab	Lists records in the **/etc/inittab** file.
lslpp	Lists software products.
mkboot	Creates the boot image, the boot record and the service record.
mkitab	Makes records in the **/etc/inittab** file.
rc	Performs normal startup initialization.
reboot	Restarts the system.
refresh	Requests a refresh of a subsystem or group of subsystems.
rmitab	Removes records in the **/etc/inittab** file.
shutdown	Ends system operation.
smit	Performs system management.
sync	Updates the i-node table and writes buffered files to the hard disk.
sysck	Checks the inventory information during installation and update procedures.

Commands List: User Interface

AIXwindows:

custom	Allows users to customize X applications.
dtscript	Builds simple dialogs used in the X Window System environment.
mwm	Runs the AIXwindows Window Manager.
uil	The command that starts the User Interface Language Compiler for the AIXwindows system.
xmbind	Configures virtual key bindings.

Enhanced X-Windows:

addX11input	Adds an X11 input extension record into the ODM database.
aixterm	Initializes an Enhanced X-Windows terminal emulator.
bdftopcf	A font compiler that converts fonts from Bitmap Distribution format to Portable Compiled format.
deleteX11input	Deletes an X11 input extension record from the ODM database.
listX11input	Lists X11 input extension records entered into the ODM database.
mkfontdir	Creates a **fonts.dir** file from a directory of font files.
resize	Sets the **TERMCAP** environment variable and terminal settings to the current window size.
rgb	Reads lines from standard input and inserts them into a database to associate color names with specific rgb values.
startx	Initializes an X session.
uil	Starts the User Interface Language Compiler for the AIXwindows system.
X	Starts the X Server.
xauth	Edits and displays the authorization information used in connecting to the X server.
xclock	Continuously displays the current time of day.
xcmsdb	Loads, queries, or removes Screen Color Characterization Data stored in properties on the root window of the screen.
xdm	X Display Manager with support for XDMCP.
xfs	Supplies fonts to X Window System display servers.
xhost	Controls who can have access to Enhanced X-Windows on the current host machine.
xinit (Enhanced X-Windows)	
xinit (X11R5)	Initializes the X Window System.
xlock	Locks the local X display until a password is entered.
xlsfonts	Displays the font list for X.
xmodmap	Modifies keymaps in the X server.
xpr	Formats a window dump file for output to a printer.
xrdb	Performs X server resource database utilities.
xset (X-Windows)	Sets options for your X-Windows environment.
xsetroot	The root window parameter setting utility for the **x** command.
xterm	Provides a terminal emulator for the X Window System.
xwd	Dumps the image of an Enhanced X-Window.
xwud	Retrieves the dumped image of an Enhanced X-Windows window.

Commands List: Macros

add_netopt	Adds a network option structure to the list of network options.
assert	Verifies a program assertion.
auth_destroy	Destroys authentication information.

clnt_call	Calls the remote procedure associated with the *clnt* parameter.
clnt_control	Changes or retrieves various information about a client object.
clnt_destroy	Destroys the client's RPC handle.
clnt_freeres	Frees data that was allocated by the RPC/XDR system.
clnt_geterr	Copies error information from a client handle.
DTOM	Converts an address anywhere within an **mbuf** structure to the head of that **mbuf** structure.
del_netopt	Deletes a network option structure from the list of network options.
feof, **ferror**, **clearerr**, or**fileno**	Checks the status of a stream.
M_HASCL	Determines if an **mbuf** structure has an attached cluster.
MTOCL	Converts a pointer to an **mbuf** structure to a pointer to the head of an attached cluster.
MTOD	Converts a pointer to an **mbuf** structure to a pointer to the data stored in the **mbuf** structure.
M_XMEMD	Returns the address of an **mbuf** cross-memory descriptor.
man	Provides a formatting facility for manual pages.
m_copy	Creates a copy of all or part of a list of **mbuf** structures.
m_clget	Allocates a page-sized **mbuf** structure cluster.
me	Provides a formatting facility for creating technical papers in various styles.
m_getclust	Allocates an **mbuf** structure from the **mbuf** buffer pool and attaches a page-sized cluster.
mm	Provides a formatting facility for business documents such as memos, letters, and reports.
mptx	Formats a permuted index produced by the **ptx** command.
ms	Provides a formatting facility for various styles of articles, theses, and books.
mv	Simplifies typesetting of view graphs and projection slides.
svc_destroy	Destroys a Remote Procedure Call (RPC) service transport handle.
svc_freeargs	Frees data allocated by the RPC/XDR system.
svc_getargs	Decodes the arguments of an RPC request.
svc_getcaller	Gets the network address of the caller of a procedure.
varargs	Handles a variable-length parameter list.
xdr_destroy	Destroys the XDR stream pointed to by the *xdrs* parameter.
xdr_getpos	Returns an unsigned integer that describes the current position in the data stream.
xdr_inline	Returns a pointer to the buffer of a stream pointed to by the *xdrs* parameter
xdr_setpos	Changes the current position in the XDR stream.

Programming Tools

Commands List: Debuggers

adb
Provides a general purpose debug program.

dbx
Provides an environment to debug and run programs.

od
Displays files in a specified format.

prof
Displays object file profile data.

savecore
Saves a core dump of the operating system.

syscall
Performs a specified subroutine call.

trace
Records selected system events.

trcdead
Extracts the trace buffer from a system dump image.

trcnm
Generates a kernel name list.

trcrpt
Formats a report from the trace log.

trcstop
Stops the trace function.

trcupdate
Adds, replaces, or deletes trace report format templates.

Commands List: Messages

dspcat
Displays all or part of a message catalog.

dspmsg
Displays a selected message from a message catalog.

gencat
Creates and modifies a message catalog.

mkcatdefs
Preprocesses a message source file.

mkstr
Creates an error message file.

runcat
Pipes the output data from the **mkcatdefs** command to the **gencat** command.

xstr
Extracts strings from C programs to implement shared strings.

Commands List: Source Programs

admin
Creates and controls SCCS files.

asa
Prints FORTRAN files.

cdc
Changes the comments in a SCCS delta.

comb
Combines SCCS deltas.

ctags
Makes a file of tags to help locate objects in source files.

delta
Creates a delta in a SCCS file.

get
Creates a specified version of a SCCS file.

prs
Displays a Source Code Control System (SCCS) file.

rmdel
Removes a delta from a SCCS file.

sact
Displays current SCCS file-editing status.

sccs
Administration program for SCCS commands.

sccsdiff
Compares two versions of a SCCS file.

sccshelp
Provides information about a SCCS message or command.

unget
Cancels a previous **get** command.

unifdef
Removes ifdef'ed lines from a file.

val
Validates SCCS files.

vc
Substitutes assigned values for identification keywords.

vgrind
Formats listings of programs that are easy to read.

whereis
Locates source, binary, or manual for program.

which
Locates a program file, including aliases and paths (the **csh** (C shell) command only).

Commands List: Object Files

ld
Links object files.

lorder
Finds the best order for member files in an object library.

make
Maintains up-to-date versions of programs.

nm
Displays the symbol table of an object file.

prof
Displays object file profile data.

size
Displays the section sizes of the Extended Common Object File Format (XCOFF) object files.

slibclean
Removes any currently unused modules in kernel and library memory.

strings

	Finds the printable strings in an object or binary file.
strip	Reduces the size of an Extended Common Object File Format (XCOFF) object file by removing information used by the binder and symbolic debug program.

Commands List: Miscellaneous Languages

bc	Provides an interpreter for arbitrary-precision arithmetic language.
bs	Compiles and interprets modest-sized programs.
m4	Preprocesses files, expanding macro definitions.
sno	Provides a SNOBOL interpreter.

Commands List: C Tools

cb	Puts C source code into a form that is easily read.
cflow	Generates a C flow graph of external references.
cpp	Performs file inclusion and macro substitution on C Language source files.
cxref	Creates a C program cross-reference listing.
execerror	Writes error messages to standard error.
indent	Reformats a C Language program.
ipcrm	Removes message queue, semaphore set, or shared memory identifiers.
lex	Generates a C Language program that matches patterns for simple lexical analysis of an input stream.
lint	Checks the C Language programs for potential problems.
m4	Preprocesses files, expanding macro definitions.
mkstr	Creates an error message file.
regcmp	Compiles patterns into C Language **char** declarations.
tic	Translates the **terminfo** descriptor files from source to compiled format.
xstr	Extracts strings from C programs to implement shared strings.
yacc	Generates a LR(1) parsing program from input consisting of a context-free grammar specification.

Commands List: Assemblers and Compilers

Assembler:

as
Assembles a source file.

FORTRAN:

asa
Prints FORTRAN files.

fpr
Prints FORTRAN files.

fsplit
Splits FORTRAN source code into separate routine files.

struct
Translates a FORTRAN program into a RATFOR program.

Commands List: Object Data Manager (ODM)

odmadd
Adds objects to created object classes.

odmchange
Changes the contents of a selected object in the specified object class.

odmcreate
Produces the **.c** (source) and **.h** (include) files necessary for ODM application development and creates empty object classes.

odmdelete
Deletes selected objects from specified object classes.

odmdrop
Removes an object class.

odmget
Retrieves objects from the specified object classes into an **odmadd** format.

odmshow
Displays an object class definition on the screen.

restbase
Reads the base customized information from the boot image and restores it into the Device Configuration database used during system boot phase 1.

savebase
Saves information about base-customized devices in the Device Configuration database onto the boot device.

Appendix C. Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
Dept. LRAS/Bldg. 003
11400 Burnet Road
Austin, TX 78758-3498
U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrates programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. You may copy, modify, and distribute these sample programs in any form without payment to IBM for the purposes of developing, using, marketing, or distributing application programs conforming to IBM's application programming interfaces.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurement may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at Copyright and trademark information at www.ibm.com/legal/copytrade.shtml

Java and all Java-based trademarks and logos are registered trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Other company, product, or service names may be the trademarks or service marks of others.

Index

A

accounting system
  correcting format problems 123
acct/* commands 307
add entries to
  product, lpp, history databases
    using vpdadd command 51
  vendor databases
    using vpdadd command 51
aliases
  locating 81
argument lists
  constructing
    using xargs command 140
ARPANET
  user of whois command 88
assemblers
  functional list of commands 314
ATE program
  functional list of commands 274
  transferring files with xmodem 181
  xmodem command 181
authorization information
  editing and displaying
    using xauth command 143

B

binding
  to NIS server
    using ypbind daemon 247
bindings
  configures virtual key
    using xmbind command 178
BNU
  functional list of commands 274

C

characters
  counting the number of
    using wc command 72
client machine
  directing to a specific server
    using ypset command 258
command
  X 125
command lines
  running
    using xargs command 140
commands
  event response resource manager (ERRM)
    wallevent 69
  vacation 1
  vpdadd 51
  vsdchgserver 55
  vsdelnode 56
commands *(continued)*
  vsdnode 59
  vsdsklst 61
  w 66
  wall 68
  wallevent 69
  who 83
  wlmassign 91
  wlmcntrl 95
  wlmmon 98
  wlmperf 98
  wlmstat 102
  wsm 119
  wsmaccess 121
  xargs 140
  xdm 149
  xmbind 178
  xmodmap 183
  xntpdc 192
  xterm 217
  ypcat 248
  ypinit 249
  yppasswd 252
  yppoll 255
  yppush 256
  ypset 258
  ypwhich 260
  ypxfr 262
communication channel
  receiving mail in a secure
    using xget command 166
  sending mail in a secure
    using xsend command 209
communications facilities
  functional list of commands 276
compilers
  functional list of commands 314
configuring
  virtual key bindings
    using xmbind command 178
connecting
  to NIS server
    using ypbind daemon 247

D

daemons
  ypbind 247
  yppasswdd 253
  ypserv 257
  ypupdated 259
debuggers
  functional list of commands 311
devices
  functional list of commands for 294
directing a client machine
  to specific server
    using ypset command 258

directories
functional list of commands 286
documentation
functional list of commands for 298
dump file
formatting for printer output
using xpr command 200

E

editors
functional list of commands 287
education
functional list of commands for 298
emulations
xterm command 218
ERRM commands
elogevent 69
ERRM scripts
wallevent 69
event response resource manager (ERRM)
commands
wallevent 69
scripts
wallevent 69

F

file contents
functional list of commands 289
file systems
functional list of commands for 298
files
compression 265
executable
locating 81
expansion 264
functional list of commands 288
locating sections 80
SCCS
displaying identifying information 74
validating 3
font directories
adding 138
font path element
removing 139
font servers
adding 137
fonts
supplying to X Window display servers 164

G

games
functional list of commands 299
hunt the wumpus 124

H

help
describing command functions 75
host machine
controlling access
using xhost command 168

I

ID number, displaying
of NIS map
using yppoll command 255
idcmds 276, 279, 283, 298, 302, 303, 304
identifying server
for a given map, master
using ypwhich command 260
image
displaying
using xwud command 242
dumping
using xwd command 241
retrieving
using xwud command 242

K

key bindings
configures virtual
using xmbind command 178
keymaps
modifying
using xmodmap command 183
keywords
SCCS
substituting values 9

L

lines
counting the number of
using wc command 72
locking X display until password is entered
using xlock command 175
logged in, users
identifying
using who command 83
logical volume
functional list of commands 300
looks up information
in NIS maps
using ypserv daemon 257

M

macros
functional list of 309
text formatting 294
mail
receiving in a securing communication channel
using xget command 166

mail *(continued)*
sending in a secure communication channel
using xsend command 209
sending vacation message
using vacation command 1
managing a collection of X displays
with support for XDMCP
using xdm command 149
map nickname table
displaying
using ypwhich command 260
master server
identifying for a given map
using ypwhich command 260
memory management
reporting virtual memory statistics 38
messages
functional list of commands 311
listing the addresses of recipients of
using whom command 89
prompting for the disposition of
using whatnow command 76
receiving from a remote system
using writesrv command 119
sending from a remote system
using writesrv command 119
sending to other users
using write command 115
verifying the addresses of recipients of
using whom command 89
writing to all users
using wall command 68
MH
functional list of commands 277
MH commands
invoking a visual interface for use with
using vmh command 31

N

NCS commands
functional list 286
network information service 247
network manager
functional list of commands 283
network password
changing in NIS
using yppasswd command 252
Network Time Protocol command
xntpd 189
Network Time Protocol daemon
starting xntpdc
using xntpdc command 192
NFS commands
functional list 279
NIS commands
functional list 280
ypcat 248
ypinit 249
ypmatch 251
yppasswd 252
NIS commands *(continued)*
yppoll 255
yppush 256
ypset 258
ypwhich 260
ypxfr 262
NIS daemons
ypbind 247
yppasswdd 253
ypserv 257
ypupdated 259
NIS maps
displaying ID number
using yppoll command 255
finds information in
using ypserv daemon 257
printing
using ypcat command 248
prompting NIS slave servers to copy
using yppush command 256
sets up on NIS server 249
transfers to an NIS server
using ypxfr command 262
updating
using ypupdated daemon 259
using ypinit command 249
NIS network password
changing
using yppasswd command 252
NIS server
binding to
using ypbind daemon 247
sets up NIS maps on 249
transfers NIS map to
using ypxfr command 262
using ypinit command 249
numerical data
functional list of commands for 302

O

object files
functional list of commands 312
ODM (Object Data Manager)
functional list of commands 314
order number, displaying
of NIS map
using yppoll command 255

P

parameter lists
constructing
using xargs command 140
parser
creating with the yacc command 244
path names
executable files
finding 81
performance tuning
functional list of commands 302

printing
activity summary
using w command 66
NIS maps
using ypcat command 248
process suspension
suspending execution 67
processes
functional list of commands 303
program
monitoring
using watch command 70
program listing
formatting
using vgrind command 11
programming languages
functional list of commands
Assembler 314
C 313
FORTRAN 314
miscellaneous 313
programs
creating a Makefile from an Imakefile 179

Q

query/control program, starting
for xntpdc daemon
using xntpdc command 192
queue
functional list of commands for 303

R

realtime messages
sending to other users
using write command 115
receives and executes requests
from the yppasswd command
using yppasswdd daemon 253
repetitive responses
generating 246
resource database
performing utilities for X server
using xrdb command 206

S

SCCS
files
displaying identifying information 74
validating 3
keywords
substituting values 9
SCCS commands
val 3
vc 9
what 74
screen color characterization data
loading from root window
using xcmsdb command 148
screen color characterization data *(continued)*
queries
using xcmsdb command 148
removing from root window
using xcmsdb command 148
screen lock
controlling
using xss Command 215
screen output
functional list of commands 304
scripts
event response resource manager (ERRM)
wallevent 69
wallevent 69
security
functional list of commands 304
locking workstation screens
using xss Command 215
shell scripts
repetitive responses
generating 246
shells
locating executable files in C 81
slave servers, NIS
prompting to copy NIS maps
using yppush command 256
software installation
functional list of commands 308, 309
source programs
functional list of commands 311
STREAMS commands
functional list 283
strings, shared
using xstr command 216
system
printing a summary of activity for the
using w command 66
system access 305
functional list of commands 304
system accounting
functional list of commands 306
system resources
functional list of commands 307
system statistics
functional list of commands 306

T

TCP/IP commands
functional list 284
terminal emulator
providing for X Window System
using the xterm Command 217
terminals
functional list of commands for 294
text formatting
functional list of commands 291
list of macro packages 294
time
displaying the current
using xclock command 146

U

updating NIS maps
  using ypupdated daemon 259
user name directory
  searching for ID
    using whois command 88
  searching for nickname
    using whois command 88
users
  displaying login name 86
  identifying those logged in
    using who command 83
  writing messages to all
    using wall command 68

V

vacation command 1
vacation message
  sends to mail recipient
    using vacation command 1
varyoffvg command 4
varyonvg command 5
vgrind command 11
vi command 13
  limitations 14
vi editor
  command mode 14
  customizing 15
  defining macros 18
  interrupting and ending 30
  last line mode 14
  mapping keys 19
  procedures
    scrolling 25
  scrolling 25
  set command
    using 15
  starting 31
  subcommands 24
  text input mode 14
view command 31
Virtual Frame Buffer (VFB)
  starting X Server with 125
virtual key bindings
  configuring
    using xmbind command 178
virtual memory
  reporting statistics 38
vmh command 31
vmo command 33
volume group
  activating
    using varyonvg command 5
  deactivating
    using varyoffvg command 4
vpdadd command 51
vsdchgserver command 55
vsdelnode command 56
vsdnode command 59
vsdsklst command 61

W

w command 66
wall command 68
wallevent command 69
wallevent script 69
watch command 70
wc command 72
whatis command 75
whatnow command 76
which_fileset command 81
who command 83
whodo command 86
whois command
  ARPANET
    use on 88
  description of 88
  example of 89
WLM
  analyzing 98
wlmassign command 91
wlmcheck command 93
wlmcntrl command 95
wlmmon 98
wlmperf 98
wlmstat command 102
wol command 107
words
  counting the number of
    using wc command 72
Workload Manager
  see WLM 98
workstation screens
  locking
    using xss Command 215
wparexec
  creates an application workload partition or
    specification file 108
write command 115
writesrv command 119
wsm 119
wsmaccess 121
wtmpfix command 123

X

x command 125
X display
  displaying troff files on
    using xpreview command 202
  locking until password is entered
    using xlock command 175
X displays
  managing a collection of
    using xdm command 149
X render extension
  starting X Server with 125

X Server
modifying keymaps in
using xmodmap command 183
starting 125
x_add_fs_fpe command 137
x_add_nfs_fpe command 138
x_rm_fpe command 139
X-Windows
setting environment options 210
setting root window parameters 214
xargs command 140
xauth command 143
xclock command 146
setting the defaults 147
xcmsdb Command 148
xdm command 149
XDMCP
manages collection of X displays supporting
using xdm command 149
xfs command 164
xget command 166
xhost command 168
xinit command 169
xlock command 175
xlsfonts command 177
xmbind command 178
xmodem command 181
xmodem protocol 181
xmodmap command 183
xmpeek command 185
xmscheck command 187
xmtopas command 188
xmwlm command 180
xntpd daemon 189
xntpdc command 192
xpr command 200
xpreview command 202
xprofiler 204
xrdb command 206
xsend command 209
xsetroot command 214
xss Command 215
xstr command 216
xterm command 217
actions 231
bugs 235
character classes 230
control sequences
definitions 235
VT100 Mode 235
xterm description limitation 240
emulations 218
environment 234
menus 229
mouse tracking 240
pointer usage 227
providing terminal emulation
for X Windows system 217
resources 222
security 229
xwd command 241
xwud command 242

Y

yacc command 244
ypbind daemon 247
ypcat command 248
ypinit command 249
ypmatch command 251
yppasswd command 252
receives and executes requests from
using yppasswdd daemon 253
yppasswdd daemon 253
yppoll command 255
yppush command 256
slave servers, NIS
prompting to copy NIS maps 256
ypserv daemon 257
ypset command 258
ypupdated daemon 259
ypwhich command 260
ypxfr command 262

Z

zcat command 264
zdump command 265
zic command 266

Readers' Comments — We'd Like to Hear from You

AIX Version 6.1
Commands Reference, Volume 6, v - z

Publication No. SC23-5248-02

We appreciate your comments about this publication. Please comment on specific errors or omissions, accuracy, organization, subject matter, or completeness of this book. The comments you send should pertain to only the information in this manual or product and the way in which the information is presented.

For technical questions and information about products and prices, please contact your IBM branch office, your IBM business partner, or your authorized remarketer.

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you. IBM or any other organizations will only use the personal information that you supply to contact you about the issues that you state on this form.

Comments:

Thank you for your support.

Submit your comments using one of these channels:

- Send your comments to the address on the reverse side of this form.
- Send your comments via e-mail to: pserinfo@us.ibm.com

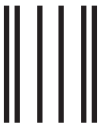
If you would like a response from IBM, please fill in the following information:

Name	Address
Company or Organization	
Phone No.	E-mail address

Readers' Comments — We'd Like to Hear from You
SC23-5248-02

Cut or Fold
Along Line

Fold and Tape | **Please do not staple** | Fold and Tape



NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

IBM Corporation
Information Development
Department 04XA-905-6B013
11501 Burnet Road
Austin, TX 78758-3400

Fold and Tape | **Please do not staple** | Fold and Tape

Cut or Fold
Along Line

Printed in U.S.A.

SC23-5248-02