Expanded personal computing power and capability

by P. A. Korn
J. P. McAdaragh
C. L. Tondo

Discussed is the XENIX™ Operating System for the IBM Personal Computer AT. The operating system incorporates capabilities of a mainframe operating system—multiusage, multitasking, file management and security, program compilation, and networking. The XENIX shell structure is introduced. Pipes and pipelining are presented. The XENIX file structure is explained and illustrated with examples. Software development and text formatting are treated in detail. The ability to compile C program code developed under XENIX and run it on the IBM Personal Computer Disk Operating System is explained.

This paper describes the IBM Personal Computer XENIX Operating System and options. (XENIX is a trademark of the Microsoft Corporation.) Discussed are the basic features of the XENIX Operating System, primarily for users who are new to UNIX, and enhancements of IBM Personal Computer XENIX. Since the first announcement of personal computers, users have sought operating systems to help them increase productivity. The XENIX system brings many of the features normally found on mainframe computers to the IBM Personal Computer AT. The IBM Personal Computer XENIX, developed in conjunction with the Microsoft Corporation, is an enhanced version of the UNIX Operating System originally developed at Bell Laboratories. (UNIX is a trademark of the AT&T Bell Telephone Laboratories, Inc.) The movement toward UNIX has resulted from its versatility and from the portability of its programs to compatible systems.

To the software developer, the prospect of developing an application on one computer system and running that application on other compatible systems from mainframes to microcomputers makes software development on the XENIX system quite attractive. The IBM Personal Computer XENIX is compatible with UNIX System III, which allows the IBM Personal Computer AT to take advantage of a large base of existing application software.

The XENIX operating system has multiuser and multitasking capabilities. The term *multiuser* on the PC AT currently means that up to three users can use the system at the same time. Two users can access the IBM Personal Computer AT through terminals or computers, and the third uses the system console. The multiuser capability also allows multiple usage of the same file. XENIX file management controls individual file execution permissions, read permissions, and write permissions for file security and integrity. This capability makes possible the immediate updating and sharing of critical data from several sources. *Multitasking* means the concurrent running of several programs, including the programs of one or more users. The XENIX system allows programs and tasks to be initiated as background processes, freeing the terminal to do other work while a time-consuming task is being completed in the background. This increases productivity because tasks are run simultaneously rather than in sequence.

Enhancements to the XENIX system from IBM include DOS-XENIX compatibility features. DOS is the IBM Personal Computer Disk Operating System. Compatibility supports transferring files between DOS and XENIX and XENIX sharing a fixed disk with DOS. Another enhancement is a compiler for the C language[1] that can compile programs to run under either DOS or XENIX. Other IBM enhancements are designed to improve user friendliness for the inexperienced user.

Microsoft enhancements are improvements in performance and matching the capabilities of XENIX with the Intel 80286 microprocessor used in the IBM Personal Computer AT. The addition of file locking and the *micnet network* by the Microsoft Corporation increases the versatility of the system.

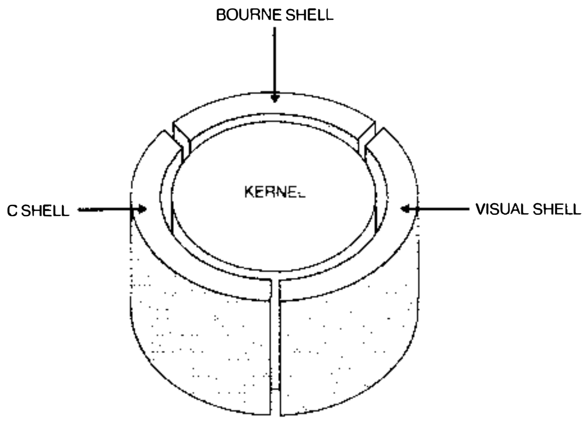
Enhancements developed at the University of California at Berkeley are *vi*, *more*, *termcap*, *style*, and *curses*. These enhancements are discussed later in this paper.

Matched hardware and software

The IBM Personal Computer XENIX system was designed to take advantage of the Intel 80286 processor used in the IBM Personal Computer AT. The processor has an advanced *protected mode* feature. During the development of the IBM Personal Computer XENIX Operating System, much attention was placed on this feature, which was designed to protect the operating system from its users and to protect the users from one another. The protected-mode processing capability allows the operating system to support multiple users. This support protects the programs of each user against other programs' executing, overlaying, or even viewing their allocated storage space. Careful and thorough testing of the operating system has resulted in an extremely stable and secure system environment.

The 80286 processor supports up to 16M bytes of memory, and IBM Personal Computer XENIX architecturally can support system memory sizes from 512K bytes to 16M bytes. This support permits users to create and execute very large programs and to have access to large amounts of data in system memory.

Figure 1 Kernel-and-shell structure of XENIX



An optional Intel 80287 math coprocessor is also supported, which means that any program using floating point arithmetic will experience significant time performance improvements. To ensure compatibility with all users of IBM Personal Computer XENIX, if the coprocessor is not present, the 80287 instruction set is automatically emulated by the operating system.

The XENIX system supports the IBM hardware available for the IBM Personal Computer AT, including up to two 20M-byte fixed disks, 1.2M-byte and 360K-byte slimline diskette drives, IBM color and graphics printers, IBM monochrome and color displays, as well as IBM enhanced color monitors. Additionally, XENIX supports a facility to add device drivers to the system. This enables both IBM and other manufacturers to support new devices as they become available.

XENIX Operating System concepts

The internal workings of the operating system can be described as a center, or *kernel*, surrounded by a layer of interfacing, or *shells*, as illustrated in Figure 1.

When a UNIX-based system is changed to run on another system, or *ported*, the major part of the work is to adapt the kernel to the new hardware requirements. The newly developed kernel accepts standard UNIX interfaces and adapts system calls to the new hardware. It is the responsibility of the kernel in the operating system to decide how to access devices such as printers and disk files. This internal structure frees the program and the developer from machinedependent code. Because the operating system kernel changes while the interface remains the same, software developed on this system is portable to other UNIX systems.

Commands and usage. Approximately 150 commands are available with the XENIX Operating System. These commands create and maintain files and directories, create file systems, mount and unmount file systems, link files across directories, and communicate with other users. The capabilities of linking and combining these commands with shell scripts, redirection, and pipelines are among the XENIX system's most powerful features.

The Bourne shell is a full programming language, in addition to being a command interpreter.

Bourne shell. The XENIX *Bourne shell* is a command language interpreter that serves as the interface between all users and the rest of the system. It interprets commands, calls the corresponding programs into memory, and executes them. This shell is a full programming language, in addition to being a command interpreter. This permits the creation of new commands simply by writing shell scripts (often called *shell procedures*). The shell language contains flow-control constructs such as **while, if, else, for,** and **case**. It also allows the definition of variables, the passing of parameters, and the processing of software signals (or software interrupts).

Input/output redirection. The XENIX shell structure provides a uniform and very flexible means for *redirecting* standard input and output. The standard input is the keyboard and the standard output is the display. By using this feature, each physical I/O device—including the keyboard, the display, and other peripherals—can be treated as any normal file. The input to a command can come from the keyboard, from a file stored on a disk, or from another program and still look like the standard input. Similarly, the output from a command can be redirected from the display, to the printer, to a disk file, or to another command. With these capabilities, a single XENIX command can often perform several distinct tasks.

Pipelines. XENIX has the ability to *pipe* data from one command to another; two or more connected programs are called a *pipeline*. The shell can thus execute a sequence of commands, with the output of each command becoming the input to the next command in the sequence. This ability to pipe commands often eliminates the need to develop new programs for new applications. For example, an application that requires sorting a file of words and printing the results can use two existing XENIX programs piped together to perform this new action as follows:

```
sort words | lpr
```

The vertical bar is the pipe symbol. The meaning of this pipeline is (1) **sort** the file named *words*, and then (2) print the output of **sort** on the printer using the **lpr** command.

The XENIX file system. The XENIX file system consists of files and directories; it allows filenames up to fourteen alphanumeric characters in length and differentiates between uppercase and lowercase. Directories help to group files for easy reference or separation. The XENIX file system, which is a *hierarchical* file system, consists of many levels or hierarchies, i.e., directories within directories. Figure 2 is an example of a hierarchical file system in which the names bin, etc, usr, you, other, and dir are directories, and the names file1 and file2 are text or source files.

The XENIX file system has a unique path for each file in the system. This unique path—also called the *absolute filename*—is the name of the file in relation to the root of the file system. The root of the file system is the slash symbol /. Within the XENIX file system some directories are standard and contain XENIX command files. The directory /bin contains most of the common XENIX commands. The directory /etc has the super-user (system administrator) commands. There are other standard directories such as /usr.

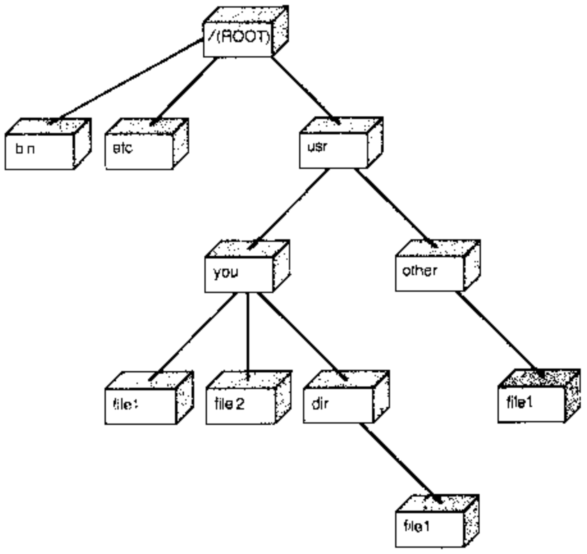
Each user of the XENIX system is assigned a *login* or *home* directory. When a system user logs in using the assigned login name, he is automatically in his own directory, which is a directory with the same name as his login name. Starting at his home directory, a user can create and delete files and directories as needed. As files are created, the user assigns read, write, and execute permissions to them. There are three sets of permissions for each file or directory: (1) those that apply to the owner; (2) those that apply to the group; (3) those that apply to anyone else with a login in the system. Regarding (2), group permissions, a system login is always assigned to some group, such as programming or planning.

If the user's login is *you*, as in the file system in Figure 2, the user has files file1, file2, and a directory dir. The file named file1 is distinct from file1 in the login *other* because they are in different directories; i.e., the files must have unique names only within a given directory. Another way to look at it is that *you* and *other* have different paths from the root, as shown in the following example taken from Figure 2:

/usr/you/file1

/usr/other/file1

Figure 2 An example illustrating the XENIX hierarchical file system



Communications. Among the several ways to communicate with other users on the XENIX system, the two basic facilities are **write** and **mail**.

The **write** command allows communications with any user currently logged in to the system. One user can write to another user, then wait for the other user to write back. Following the execution of the **write** command, whatever is typed on the first user's terminal appears on the other user's terminal and vice versa. The users take turns reading and writing messages until one decides to end the conversation.

The other basic way to communicate with a user is with the **mail** command. **Mail** sends a message or a previously created file to the other user. The receiving user does not have to be logged in or even be a user on the same system to receive mail. Mail is delivered to the user's mailbox directory and, when the receiving user logs in, a message indicates that mail has arrived. The user can then **read, delete, save,** or **transfer** the received mail to other users.

Networks. If one wants to communicate with a receiving user who is not in the same system (i.e., the login or home directory is in another system), additional communication facilities may be needed. First, the two systems must be able to communicate with one another. If the systems are not far apart—e.g., across the room—serial lines and the RS-232 ports on the IBM Personal Computer AT can be used to form a *micnet network.* If the other system is across town or across the country, a modem and the XENIX **uucp** communication system can be used to complete the network. In XENIX, the command **uucp** retains its UNIX meaning of UNIX-to-UNIX copy. IBM Personal Computer XENIX micnet is a limited local network that connects personal computers via the serial ports. Once a form of communication has been established between the two systems, the **mail** command is used to communicate with users in the other system. Commands within micnet allow copying or transferring files and accomplishing tasks such as executing commands on a remote XENIX system. The **uucp** communication system can combine the usefulness of local micnet systems by connecting them via modems. The **uucp** system also allows file transfer and mail activities.

Software development

The XENIX Software Development System provides a large set of tools, including a C compiler,[1] a debugger, and such productivity tools as the command **make** and the Source Code Control System (SCCS).

Most of the software in the XENIX system is written in the general-purpose programming language C.

This language has been used for a wide variety of applications, including operating systems and application programs.

The C compiler[2] can create programs for small, middle-sized, and large types of memory models. Each type of model is distinguished by its use of available memory space and how it uses programs and data in that space.

Small-model programs can be pure or impure with regard to space usage. Impure-text, small-model programs occupy one 64K-byte physical segment. The program and data areas are combined in one segment. Pure-text, small-model programs occupy two 64K-byte physical segments. One segment is used for data and the other for text. The second is a read-only segment. This means that a pure-text program can be shared by several processes at a time, although only one copy is loaded in memory. The total program size for a pure small version is 128K bytes.

Middle-model and large-model programs generate pure-text programs. Middle-model programs use one segment for data and as many segments as necessary for the program. There is no limit on the program (text) size. The only limitation is that the space required for variables (data area) cannot exceed 64K bytes. Large-model programs use as many segments as necessary for data and programs. There is no limit on the size of either.

A significant feature of the IBM Personal Computer XENIX C compiler is DOS cross-compilation, which lets the programmer create code to be run under IBM Personal Computer DOS (Version 2.00, 2.10, or 3.00). The programmer creates and debugs C programs using the productivity tools available in XENIX, then recompiles the program for DOS. The IBM Personal Computer XENIX provides a library for XENIX C programs and another library for DOS C programs. The C compiler selects the correct library and generates the appropriate load module. The programmer can then use DOS utilities available in XENIX, such as **doscp** or **doscat**, to copy the load module to a DOS diskette.

The C compiler is also capable of generating code for XENIX or DOS in such a way that if an 80287 or 8087 chip is present, the floating point operations are performed in hardware. If no math coprocessor is present, the floating point operations are emulated. Sometimes a programmer determines that certain tasks cannot be done in C or will be better performed when written in assembler. The XENIX Software Development System provides an assembler for the 80286/80287, 8088/8087, 80186, and 8086. The assembler is similar to the IBM Personal Computer DOS assembler except for macro support.

The programmer does not have to remember files to be recompiled and relinked when a change occurs.

Development facilities. Because large programming projects tend to have multiple source files, there may be too many dependencies among files to be remembered at all times. A modification in one file might affect other files. The **make** command is a program maintainer that coordinates these dependencies. This command reads a file that specifies the file dependencies in the project. Usually a programmer specifies that a load module depends on certain object modules, and each object module depends on a source file. If a source file has been changed (i.e., its date is more recent than that for the object file), the object file is recompiled. The new object file is then linked with other object files to produce a new load module. This way the programmer does not have to remember the files that have to be recompiled and relinked when a change occurs.

The Source Code Control System (SCCS) controls the maintenance of any files of text or source code. The maintenance process starts by running the SCCS **admin** command on the files containing the text. SCCS provides other commands to retrieve and update the files, add new releases, and recreate the complete source of a release at any point in time. SCCS saves the original files and maintains a list of modifications (called *deltas*) that have been applied to each of the files. To obtain a specific release, SCCS starts with the original file and applies the correct deltas to the files involved.

Some components of the XENIX Software Development System are an archiver, a C beautifier, a C-shell, a lexical analyzer called *lex*, preprocessors, and a compiler-compiler.

Text formatting

The IBM Personal Computer XENIX Text Formatting System is composed of a collection of tools that complement writing productivity. In combination with any of the system's text editors, the text formatting system can be especially valuable in simplifying the generation of technical reports, formal papers, and other professional-quality documents.

Most of the commands to format text appear at the beginning of text lines and consist of one or two letters preceded by a dot. This makes XENIX text formatting unlike typical microcomputer word processing programs, because the formatting of text takes place all at once rather than interactively. The desired output is generated by executing a formatter program that uses a text file as input to create the desired results. This allows the text formatting programs to provide great capabilities in text formatting.

At the heart of the Text Formatting System are the **nroff** and **troff** text formatters. The **nroff** formatter produces output on either matrix- or line-printer-type devices. The **troff** formatter, which is compatible with **nroff**, offers more sophisticated capabilities because it supports phototypesetters. For example, **nroff** uses underlining, approximate spacing, and text size determined by the characteristics of a particular printer, and **troff** supports italics, variable spacing, and point size.

To use **nroff** or **troff**, a series of commands are inserted directly into the text to specify in detail the appearance desired for the final output. The multiple capabilities of **nroff** and **troff** include margin justification, vertical spacing, line and page length, indentation, automatic hyphenation, page headers and footers, page numbering, nested input files, macros with substitutable arguments, multicolumn output, font type (**troff** only), point size (**troff** only), and many other features.

Although these formatters are very powerful, they can also be difficult to master. Therefore, to reduce the learning time required and to increase user productivity, a series of aids have been introduced.

Memorandum macros (**mm**) provide a simplified interface for translating input to **nroff/troff** specifications. With this macro support, the user can specify the style of paragraphs, titles, footnotes, bullets, and headings. Support is provided for user-defined macros, hyphenation, multicolumn output, and tables of contents.

Additional formatting facilities may be used in conjunction with these programs for building sophisticated mathematical equations and for designing complex tables. The equation (**eqn**) preprocessor can format complicated mathematical symbols and equations using commands that resemble English words. Some common **eqn** commands are **over** and **union**, Greek letters such as alpha and gamma, keywords such as inf and int (for infinity and integral symbols), and shorthand notation such as < and =. **Eqn** also supports subscripts, superscripts, piles, matrices, and a host of other mathematical features. Another version of this preprocessor, termed **neqn**, is available for use on matrix- or line-printer-type devices.

Cut and paste utilities rearrange text blocks within a document.

A **tbl** preprocessor gives control over material that must appear in tabular form, and it automatically calculates the information necessary to line up complicated columnar data even if elements have varying widths. It also centers a table, expands a table to the width of the current line, and encloses a table in a box.

Because the text formatting package is a set of building blocks that work together, the user can combine formatting commands like **eqn, tbl, mm,** and **troff** into one file and then process it all at once.

Text analysis. Another series of text formatting system utilities for analyzing writing style and correctness are *style, diction,* and *explain.* These utilities were developed at the University of California at Berkeley. The style utility analyzes writing style and reports on readability, sentence length and structure, word length and usage, verb type, and sentence openers. The diction utility finds all sentences in a document that contain phrases from a data base containing bad or wordy diction. The explain utility interactively suggests alternatives to phrases flagged by the diction utility.

A *spell* utility is useful in finding misspelled words in a file. It has a large dictionary of words, and it supports adding a list of additional words. It also has an option that checks for British spellings such as colour and centre.

Text modification. *Cut* and *paste* utilities rearrange text blocks within a document. The *cut* utility can extract or copy fields of information from a file and provide the user with a facility to delete either columns from a table or fields from each line of a file. *Paste* can merge "cut" information into a specified file. Paste can also merge lines of a single file horizontally.

The XENIX Operating System installation procedure was designed to combine flexibility and simplicity.

XENIX enhancements

Several enhancements to the XENIX operating system were added by IBM during the development of the IBM Personal Computer XENIX. One enhancement is modular packaging of the total system, whereby the IBM XENIX is split into three parts: (1) the XENIX Operating System; (2) the XENIX Software Development System; and (3) the XENIX Text Formatting System. The main advantage of splitting the systems is cost reduction for the end user. The prospect of purchasing the XENIX Operating System as a standalone package to run application software is a significant saving to users not requiring software development or text formatting capabilities.

The system is distributed on large-capacity (1.2M-byte) diskettes supported by the IBM Personal Computer AT, thereby permitting the operating system along with the system installation diskette to be shipped on just four diskettes. The entire XENIX system is provided on eight diskettes. By utilizing a simple-to-follow installation procedure, the system can be easily and quickly installed even by inexperienced users.

The ability to coreside with and transfer files to IBM Personal Computer DOS adds a dimension of compatibility with the existing personal computer operating system. This can be a great advantage to users with existing applications running under DOS.

Installation and coresidence. The XENIX Operating System installation procedure was designed to combine the flexibility needed to handle system variations and the simplicity needed by a novice. The installer loads a diskette-based version of the XENIX Operating System to start the installation procedure. A bad-track program automatically scans the surface of the fixed disk and records the locations of any bad or unusable tracks. This information is then passed to the operating system so that these sectors are not used to store information.

The system then prompts the installer for the type of installation. The installation procedure supports the following three installation options: (1) dedicating the entire disk to the XENIX system; (2) installing the system with a space at the end of the fixed disk to install an additional operating system; and (3) installing the XENIX system on a disk that already contains another operating system. The installation procedure automatically calculates disk space allocation and partition definitions.

When the initial procedures and checking are complete, the installation program copies the XENIX system kernel to the fixed disk along with other system files from the installation diskette. Once the installation diskette files are copied to the fixed disk, the installer is prompted to insert the operating system diskettes. This procedure continues until all the operating system and options diskettes are installed on the fixed disk.

Administrative utilities. Some new system administration commands that are unique to XENIX are called *super-user commands*, because they facilitate the maintenance of the users' logins. The system administrator is the super-user and is the only one permitted to use these commands. The **mkuser** command is used to add a user login to the system. This command handles all directory creation and updating of shell assignments and passwords. When the super-user creates a user login, he assigns one of the shells (Bourne shell, C shell, or visual shell) as the default shell. The **rmuser** command does the opposite by making sure that the user's mailbox is empty and that all files belonging to that user have been

A user-friendly shell interface was added to the operating system.

deleted. **Rmuser** then deletes the user's name from the password file and removes his home directory. The **chsh** command allows the super-user to change a user's default shell.

DOS utilities. XENIX provides the capability of transferring files between IBM Personal Computer DOS diskettes and the XENIX file system. Eight utilities access files and directories on DOS diskettes. These utility commands have a "dos" prefix and include support for copying files, listing files and directories, erasing files, and creating and deleting directories. As an example, one of these commands is **doscp**, which copies files between a DOS diskette and the XENIX file system. **Doscp** takes two arguments, file1 and file2, and copies the contents of file1 to file2. Either filename can be made a DOS file by using the form

```
device:file
```

where device is the path name of the special device containing the DOS file.

A new, simplified way to specify the DOS device is provided in XENIX by a set of capital letters that represent the DOS devices. For example, if a DOS diskette is 48 tracks per inch (TPI), and it is in drive A, to copy the DOS file file1 to the XENIX file system and keep the same filename, the syntax is the following:

```
doscp A:file1 file1
```

To copy file1 from the XENIX file system to a 48-TPI DOS diskette in drive B and rename it to file2, the following syntax is used:

```
doscp file1 B:file2
```

Visual shell. A user-friendly shell interface was added to the operating system. The visual shell is a full-screen shell interface that optionally replaces the Bourne shell as the command interpreter for inexperienced users. The visual shell replaces operating system commands, which may be cryptic, with common terms selected by cursor movement.

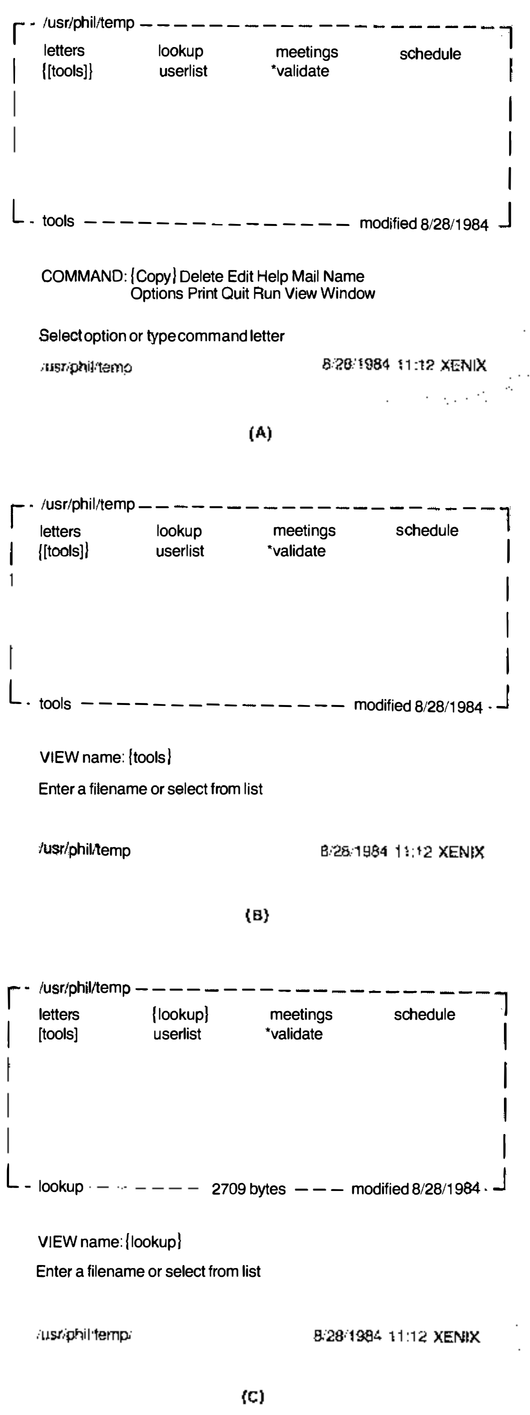
A typical visual shell screen might look as shown in Figure 3A. At the bottom of the screen are a series of English language words representing system commands. Using the space bar, the backspace key, or the first letter of any of the words, the user selects a command to be executed. This set of command words, or menu, permits the user to display files, copy files, delete files, rename files, send and receive mail, and send output to the printer; it offers several additional facilities including help documentation.

At the top of the screen is a window in which all the files in the current directory are displayed. The user can select file(s) to use with the command word by moving the cursor (via the cursor movement keys) to the filename(s) desired. To change directories, the user simply types an equal (=) to go down one level or a minus (−) to go up one level. Optionally, the user can type the filename on the command line. The current directory name is displayed at the top left of the window. At the bottom of the window is information about the file pointed to by the cursor. Several filenames appear in the window in Figure 3A. The filename in brackets indicates that the entry displayed is actually a directory. The filenames preceded by an asterisk are files marked as executable. The cursor is currently at the directory "tools."

We now give an example of a user's displaying the file "lookup." By pressing v on the terminal (meaning "view"), the screen in Figure 3A is changed to look as shown in Figure 3B. The file "tools" is initially inserted on the command line because the window cursor is currently pointing at the "tools" filename. By using the cursor movement keys, the file "lookup" is selected (Figure 3C).

The file named "lookup" is now automatically inserted on the command line because the window cursor is currently pointing to that filename. When the Enter key is pressed, the contents of "lookup"

Figure 3 Example visual shell procedure to display a file named "lookup" showing (A) a typical screen; (B) the screen after the terminal key "v" has been pressed, and (C) the screen after the file named "lookup" has been selected



```
┌ - /usr/phil/temp ---------------------------------------┐
│  letters        lookup        meetings        schedule  │
│  {[tools]}      userlist      *validate                 │
│                                                         │
│                                                         │
│                                                         │
└ - tools ----------------------------- modified 8/28/1984 ┘

   COMMAND: {Copy} Delete Edit Help Mail Name
                Options Print Quit Run View Window

   Select option or type command letter

   /usr/phil/temp                        8/28/1984 11:12 XENIX
```

(A)

```
┌ - /usr/phil/temp ---------------------------------------┐
│  letters        lookup        meetings        schedule  │
│  {[tools]}      userlist      *validate                 │
│                                                         │
│                                                         │
│                                                         │
└ - tools ----------------------------- modified 8/28/1984 ┘

   VIEW name: {tools}

   Enter a filename or select from list


   /usr/phil/temp                        8/28/1984 11:12 XENIX
```

(B)

```
┌ - /usr/phil/temp ---------------------------------------┐
│  letters        {lookup}      meetings        schedule  │
│  [tools]        userlist      *validate                 │
│                                                         │
│                                                         │
│                                                         │
└ - lookup --------- 2709 bytes ------- modified 8/28/1984 ┘

   VIEW name: {lookup}

   Enter a filename or select from list


   /usr/phil/temp                        8/28/1984 11:12 XENIX
```

(C)

are displayed in the window. If the file contains more lines than will fit in the window, control keys can be used to move up or down within the file.

Another major feature of the visual shell, especially valuable to software developers, is the ability to change the shell easily. By developing appropriate shell scripts or programs and changing the words on the menu, application developers can make a simple application interface for the novice or occasional system user. This flexibility enables complicated data processing tasks to be performed with minimal computer knowledge.

The Microsoft Corporation added several enhancements to the standard UNIX system while producing XENIX. These enhancements help take advantage of the processing power of the IBM Personal Computer AT.

File locking. Multiple processes may access a file for reading and writing. When a process is reading information from a file, that information must not change until the reading has ended. Therefore, it becomes necessary to lock different parts of a file. Similarly, when a process is writing to a file, another process should not access the information until the writing has been completed.

Locking a file synchronizes file usage when several processes require access to the same file. One or more bytes can be locked in any region of a file. When one is writing to a file, all other users can be locked out of that area. Because only a specific region of a file is locked, the rest of the file can be accessed or locked by other users.

Semaphores. Semaphores are signals that allow the IBM Personal Computer XENIX system to control access to its resources. Semaphores are named like files and are created by a process that synchronizes the access of other processes to the resource. The control of a semaphore passes from process to process. The process that controls the semaphore has control of the system resource.

Shared data. A *shared data segment* (also known as a *shared memory segment*) improves speed and facilitates communication among processes. It works like a temporary file, except that it is kept in memory. A process can create a shared data segment. Other processes can request access, wait if necessary, and gain access to a shared segment. Once a process has finished accessing a shared data segment, it signals other processes. The sharing of data allows processes to pool information without creating temporary files. The process that creates a shared data segment may allow simultaneous access to the segment by multiple processes and can free the segment when the space is no longer needed.

The IBM Personal Computer XENIX includes extensions written at the University of California at Berkeley. Among the most significant are the C shell, vi, and termcap/curses.

C shell. The C shell (csh) is a command language interpreter. Like the IBM Personal Computer XENIX Bourne shell (sh), the C shell is an interface between the user and the system that translates command lines typed at a terminal into corresponding system actions. It is packaged as part of the IBM Personal Computer XENIX Operating System. The C shell offers features other than those specifically provided by the Bourne shell. For example, the C shell can maintain a history list in which it places the text of previously entered commands. By using a special notation, commands or words from previously typed commands can be used to form new commands. This mechanism is useful for repeating commands or correcting typing mistakes.

An *alias* facility simplifies commands, supplies default arguments, and performs transformations on commands and their arguments. This mechanism enables the user to substitute, for example, a personally preferred name for a system command, for a system command and its arguments, or even for multiple commands or pipelines.

Also provided by the C shell are control structures similar to those of the C language, as well as features to help trace the actions of the C shell. These control structures make it easy for users to write shell procedures.

Full-screen editor. The vi editor is a full-screen text editor based on a set of mnemonics commands. Most commands are single keystrokes that perform editing functions. The vi editor, which combines line-oriented and screen-oriented features to increase editing productivity, is packaged as part of the IBM Personal Computer XENIX Operating System. The capabilities of the vi editor include inserting characters, deleting lines, searching for a character, replacing a string wherever it is found in a file, and splitting and rejoining lines. Cursor control permits movement in any direction, to any line, or to a word, sentence, line, or paragraph boundary.

Multiple files may be edited by vi simultaneously, and users may be permitted to leave the editor temporarily to execute any system command. This can be especially helpful when writing or debugging programs or shell scripts. The editor vi also supports a series of options to tailor its execution to individual needs. These options include features for displaying hidden and end-of-line characters in files, set tabs, and display line numbers, and for preventing messages from other users from interrupting screen displays. Additionally, vi allows options to be predefined in a file so that they are automatically set each time vi is invoked.

Another system feature is a database that describes terminals.

Terminal support. Another system feature developed at the University of California at Berkeley is *termcap*, which is a database that describes terminals. It is used by programs such as vi to interface with a user's terminal. Termcap in IBM Personal Computer XENIX supports many ASCII terminals, including terminals manufactured by vendors other than IBM. Termcap is packaged as part of the IBM Personal Computer XENIX Operating System. Termcap support can also be easily added for new terminals as they become available. Some of the terminal features that are described in a termcap database entry are number of lines, number of columns in each line, how the screen is cleared, backspace capability, cursor motions, highlighting, underlining, and insert/delete character keys.

As a complement to termcap, *curses* is a screen-updating and cursor-movement set of library functions that are packaged as part of the IBM Personal Computer XENIX Software Development System. Curses relies on the information in the termcap database to tailor terminal commands, thereby allowing the user to write terminal-independent programs. These functions provide a simple and efficient way to use the capabilities of the terminal attached to the program's standard input and output files.

Concluding remarks

Because of the increasing power and complexity of computer hardware, more sophisticated operating systems are needed. The XENIX Operating System takes advantage of the hardware capabilities of the IBM Personal Computer AT and also brings compatibility to many different hardware systems. This paper has discussed some of the features of the IBM Personal Computer XENIX Operating System. Because of the number of multifaceted commands and utilities, the XENIX system allows the user to tailor a unique working environment to facilitate his computing tasks. This operating system is an integral part of the microcomputing industry and a step toward increasing the power and usefulness of the personal computer.

Cited references

1. *Bell System Technical Journal* **57**, No. 6, Part 2, whole issue (1978).
2. R. R. Ryan and H. Spiller, "The C programming language and a C compiler," *IBM Systems Journal* **29**, No. 1, 37–48 (1985, this issue).

Reprint Order No. G321-5235.

Philip A. Korn *IBM Entry Systems Division, P.O. Box 1328, Boca Raton, Florida 33432.* Mr. Korn is currently a senior programmer in the Engineering/Scientific Microsystems Development Department, where he is involved in the development and testing of the IBM Personal Computer implementation of XENIX. In 1970, he joined IBM in East Fishkill, New York, where he was a systems programmer providing support for OS/MVT and MVS users. In 1975, Mr. Korn joined the IBM System Products Division, where he has worked on the Audio Distribution System and the RPS operating system on the Series/1 computer system. He received his B.S. degree in mathematics from Hunter College of the City University of New York.

John P. McAdaragh *IBM Entry Systems Division, P.O. Box 1328, Boca Raton, Florida 33432.* Mr. McAdaragh joined the Office Products Division in 1977. While a member of that division, he received the IBM Means Service Award. He transferred to the Entry Systems Division in 1983 and there joined the Information Development Department. His work there has included project planning and team leading in personal systems software information development. He was responsible for the IBM Personal Computer XENIX library. Mr. McAdaragh is currently doing research in on-line graphics and tutorial information, along with other planning responsibilities. Mr. McAdaragh received his B.A. degree in management in 1983 from Governors State University, Park Forest South, Illinois.

Clovis L. Tondo *IBM Entry Systems Division, P.O. Box 1328, Boca Raton, Florida 33432.* Mr. Tondo is currently a staff programmer in the Engineering/Scientific Microsystems Development Department. There he is involved in the development and testing of the IBM Personal Computer implementation of XENIX. Prior to this assignment, he worked in compiler construction for COBOL and FORTRAN on the Series/1 computer. Before joining IBM, Mr. Tondo worked with UNIX at the Western Electric Company and the Bell Telephone Laboratories on the Number 5 Electronic Switching System. He has a B.S. degree in civil engineering from the Federal University of Santa Maria in Brazil, and an M.S. degree in computer science from Southern Illinois University at Carbondale.