Digital UNIX

Technical Overview

Order Number: AA-QTLLA-TE

March 1996

Product Version: Digital UNIX Version 4.0 or higher

This document describes the functionality in Digital UNIX Version 4.0 or higher. Digital UNIX was formerly called DEC OSF/1.

Digital Equipment Corporation
Maynard, Massachusetts

Contents

4 File System

5 Virtual Memory

6 I/O Subsystem

7 Development Environment

8 Windowing Environment

9 System V Functionality

10 Internationalization

11 Security

A Conformance to Internet Host Requirements

Index

Figures

Tables

About This Manual

This document provides a brief technical overview of the functionality in Digital UNIX ® Version 4.0.

Note

This book is in no way intended to supersede the *Software Product Description* (SPD), which is the definitive legal document describing the functionality in Digital UNIX Version 4.0 that Digital supports.

Audience

This manual is for anyone who is interested in the functionality in Digital UNIX Version 4.0.

Organization

This document contains the following chapters and appendixes:

Related Documents

You should have access to the *Software Product Description* (SPD), the *Systems and Options Catalog*, and the entire Digital UNIX Version 4.0 documentation suite.

The printed version of the Digital UNIX documentation set is color coded to help specific audiences quickly find the books that meet their needs. (You can order the printed documentation from Digital.) This color coding is reinforced with the use of an icon on the spines of books. The following list describes this convention:

Audience	Icon	Color Code
General users	G	Blue
System and network administrators	S	Red
Programmers	P	Purple
Device driver writers	D	Orange
Reference page users	R	Green

Some books in the documentation set help meet the needs of several audiences. For example, the information in some system books is also used by programmers. Keep this in mind when searching for information on specific topics.

The *Documentation Overview, Glossary, and Master Index* provides information on all of the books in the Digital UNIX documentation set.

Reader's Comments

Digital welcomes any comments and suggestions you have on this and other Digital UNIX manuals.

You can send your comments in the following ways:

- Fax: 603-881-0120 Attn: UEG Publications, ZK03-3/Y32
- Internet electronic mail: `readers_comment@zk3.dec.com`

  A Reader's Comment form is located on your system in the following location:

  `/usr/doc/readers_comment.txt`

- Mail:

  Digital Equipment Corporation
  UEG Publications Manager

ZK03-3/Y32
110 Spit Brook Road
Nashua, NH 03062-9987

A Reader's Comment form is located in the back of each printed manual. The form is postage paid if you mail it in the United States.

Please include the following information along with your comments:

- The full title of the book and the order number. (The order number is printed on the title page of this book and on its back cover.)
- The section numbers and page numbers of the information on which you are commenting.
- The version of Digital UNIX that you are using.
- If known, the type of processor that is running the Digital UNIX software.

The Digital UNIX Publications group cannot respond to system problems or technical support inquiries. Please address technical questions to your local system vendor or to the appropriate Digital technical support office. Information provided with the software media explains how to send problem reports to Digital.

Conventions

The following conventions are used in this guide:

`%` `$`	A percent sign represents the C shell system prompt. A dollar sign represents the system prompt for the Bourne and Korn shells.
`#`	A number sign represents the superuser prompt.
`% cat`	Boldface type in interactive examples indicates typed user input.
`file`	Italic (slanted) type indicates variable values, placeholders, and function argument names.
`[ | ]` **`{ | }`**	In syntax definitions, brackets indicate items that are optional and braces indicate items that are required. Vertical bars separating items inside brackets or braces indicate that you choose one item from among those listed.
. . .	In syntax definitions, a horizontal ellipsis indicates that the preceding item can be repeated one or more times.
`cat`(1)	A cross-reference to a reference page includes the appropriate section number in parentheses. For example, `cat`(1) indicates that you can find information on the `cat` command in Section 1 of the reference pages.

Mb/s	This symbol indicates megabits per second.
MB/s	This symbol indicates megabytes per second.
Ctrl/*x*	This symbol indicates that you hold down the first named key while pressing the key or mouse button that follows the slash. In examples, this key combination is enclosed in a box (for example, Ctrl/C).

Introduction 1

1.1 Overview of Digital UNIX Version 4.0

Digital UNIX Version 4.0 is Digital Equipment Corporation's implementation of the Open Software Foundation Version 1.0 and Version 1.2 technology, and the Motif Version 1.2.3 graphical user interface and programming environment. Digital UNIX Version 4.0 also ships with Motif Version 1.1.3 to ensure backward compatibility with applications that link to those libraries. In addition, Digital UNIX Version 4.0 supports the full features of the X Window System, Version 11, Release 6 (X11R6) from MIT.

The Digital UNIX Version 4.0 operating system is a multiuser/multitasking, 64-bit, advanced kernel architecture based on Carnegie Mellon University's Mach Version 2.5 kernel design with components from Berkeley Software Distribution (BSD) Versions 4.3 and 4.4, UNIX System Laboratories System V Release 4.0, other software sources, the public domain, and from Digital Equipment Corporation.

Digital UNIX Version 4.0 incorporates several performance enhancements either developed or extended by Digital, including the Virtual Memory Unified Buffer Cache and eager swap policy; UFS file block clustering and cached writes over NFS; IP Multicasting, path MTU discovery, and optimized TCP/IP; and quickstarted shared libraries.

Digital UNIX Version 4.0 is also supported on selected workstations with 24 MB of memory and 535 MB of disk space.

In addition, Digital UNIX Version 4.0 provides a clear and concise system administration environment (both graphics and character-cell) to greatly simplify system administration tasks; supports an update installation that does not overwrite system files and a new and improved full installation that allows you to get up and running almost immediately while files are being copied from the CD-ROM onto your system disk; supports loadable drivers and other kernel subsystems, including loadable boot-path support for third-party disks and graphics cards; and provides support for dynamic system configuration and dynamic system recognition of disks and tapes.

Digital UNIX Version 4.0 also supports the Common Desktop Environment (CDE) as the default user interface. CDE provides a uniformed graphical user interface — portable across multiple platforms — to greatly facilitate

common end-user and system administration tasks. The CDE uniformed graphical interface makes Digital UNIX appear more like a PC or Macintosh environment, a feature that makes Digital UNIX more accessible to the many end-users familiar with those systems.

Digital UNIX Version 4.0 also provides realtime support and symmetrical multiprocessing (SMP), dataless servers and clients, and numerous features intended to assist application programmers in developing applications that use shared libraries, threads, and memory mapped files. It is fully compliant to the Single UNIX Specification, to the X/Open UNIX brand, to POSIX 1003.1B (Realtime) and to POSIX 1003.1C (with DECthreads).

To ensure a high level of compatibility with Digital's ULTRIX operating system, the Digital UNIX Version 4.0 operating system is compatible with the Berkeley 4.3 and System V programming interfaces and, by complying with the System V Interface Definition (SVID3 Base and Kernel Extensions), Digital UNIX Version 4.0 supports System V applications as well.

Since part of the charter of the Open Software Foundation is to provide an interface for developing portable applications that will run on a variety of hardware platforms, Digital UNIX Version 4.0 is compliant with the OSF Application Environment Specification (AES) that specifies the interface to support these portable applications. In addition, the Digital UNIX Version 4.0 operating system complies with standards and industry specifications, including FIPS, POSIX, X/Open, XTI, and AT&T System V Interface Definition (SVID).

For a complete list of the standards that Digital UNIX Version 4.0 supports, see the Software Product Description (SPD).

1.2 Packaging

Digital UNIX Version 4.0 is available as a base system kit, containing the operating system, windowing environment, and documentation all integrated on CD-ROM, as well as the following three extensions, also included on the CD-ROM, that provide additional functionality and that require separate licenses and Product Authorization Keys (PAK) to access:

- Server Extensions

  The Server Extensions kit contains the Remote Installation Service (RIS) software, which allows a server system using the `bootp` protocol to install Digital UNIX Version 4.0 to client systems over a Local Area Network (LAN). For more information on RIS, see the guide *Sharing Software on a Local Area Network*.

- Developers' Toolkit

  The Developers' Toolkit is designed for programmers using languages other than C, like Fortran, C++, Ada, or Pascal, who require the complete

software development environment but who do not require the C compiler, which is not available in this kit.

In addition, the Developers' Toolkit contains a fully operational `dbx` debugger that allows the debugging of source code. The `dbx` debugger that ships on the base system kit only supports debugging a kernel.

- C Developers' Extensions

  The C Developers' Extensions is designed for C programmers and includes the C compiler, assembler, and the complete software development environment for both the base and worksystem environment.

  In addition, the C Developers' Extensions kit contains a fully operational `dbx` debugger that allows the debugging of source code. The `dbx` debugger that ships on the base system kit only supports debugging a kernel.

Note

All Digital UNIX Version 4.0 documentation produced by Digital ships as Bookreader files on the Digital UNIX Version 4.0 CD-ROM and as ASCII reference pages accessible from the `man` command.

The following hardcopy documentation ships with the Digital UNIX Version 4.0 CD-ROM:

- *Release Notes*
- *Installation Guide*
- *Update Installation Quick Reference Card*
- *Technical Overview*
- *Quick Reference Card*
- *Documentation Map*

Complete hardcopy documentation and several third-party books are also available. For more information on the makeup of the documentation set, including optionally available documentation, see the *Documentation Overview, Glossary, and Master Index.*

The remaining chapters and appendices discuss the following components of Digital UNIX Version 4.0:

- Symmetrical Multiprocessing
- Networking
- The File System Subsystem
- The Virtual Memory Subsystem
- The I/O Subsystem
- The Development Environment
- The Windowing Environment
- System V Functionality
- Internationalization
- Security
- Installation and System Setup
- System Administration
- Maximum System Limits
- Conformance to Internet Host Requirements

Symmetrical Multiprocessing 2

2.1 Overview

Symmetrical multiprocessing (SMP) is the ability of two or more processes (or multiple threads of a threaded application) to execute simultaneously on two or more CPUs. This concurrency of execution greatly improves performance. Additionally, it affords customers the opportunity to extend the life and increase the cost-effectiveness of their multiprocessor systems by adding CPU cards (and their compute power) to their multiprocessors rather than buying more systems.

Digital UNIX supports an implementation of SMP that is designed to optimize the performance of **compute servers** (systems dedicated to compute-bound, multithreaded applications) and **data servers** (file servers, DBMS servers, TP systems, and mail routers that serve a large number of network clients). In addition, Digital UNIX supports multithreaded application development in an SMP environment. Note that SMP does not adversely affect using a multiprocessor as a timesharing system.

2.2 Implementation

The Digital UNIX SMP implementation makes use of **simple locks** (also called **spin locks**, since they "spin" for a specified period of time waiting for held locks to be freed before timing out), **complex locks** (read/write locks that can block waiting for a lock to be freed), and in very rare cases where locks would not be of benefit, **funneling**, whereby a process is forced to execute on a specified CPU.

The Digital UNIX SMP implementation also endeavors to achieve as much **concurrency** as possible by reducing the size of the system state that must be protected by locks, thereby reducing the necessity for locks and their attendant overhead.

Thus, both the kernel, and for the most part, the operating system as a whole, are fully parallelized so that multiple processes or multiple threads can run simultaneously on multiple CPUs. As these multiple processes or multiple threads execute, the operating system ensures—either through concurrency or through its locking strategy—that processes that access the same kernel data structures do so in a logical order so that the integrity of these data structures

is maintained and that processes do not hold and request each other's locks, thereby deadlocking the system. There are no architectural limits on the number of CPUs supported.

Digital UNIX SMP also supports **processor affinity**, the ability to bind a particular process to a specified CPU, and **load balancing**, whereby the scheduler attempts to distribute all runable processes across all available CPUs. (Note that load balancing will not override processor affinity).

To improve performance, the scheduler also attempts to execute each process on the last CPU where it ran to take advantage of any state that may be left in that CPU's cache.

SMP is configurable and any of the following five modes can be configured at system boot time:

- Uniprocessing
- Optimized realtime preemption
- Optimized SMP
- Optimized realtime preemption and SMP
- Lock debug mode

When uniprocessing is set, only those locks necessary to support multiple threads are compiled into the kernel at system boot time.

When lock debug mode is set, the system checks the lock hierarchy and minimum system priority level (SPL); stores debugging information by classes and maintains lock statistics; records the simple locks that are held by each CPU in CPU-specific arrays; and records all of the complex locks that a thread is holding in the thread structure. All of this debugging information can be accessed through the `dbx` debugger.

In addition, the development environment has been enhanced to support multithreaded application development. Specifically the `dbx`, `profile`, and `pixie` utilities have had been extended to include support for multiple threads, and more thread-safe libraries have been added to the system.

For information on the Digital UNIX development environment and the threads package that Digital UNIX supports, see the *Programmer's Guide* and the *Guide to DECthreads*. For information on configuring SMP, see the *System Administration* guide and the *System Tuning and Performance Management* guide.

Networking 3

3.1 Overview

The networking functionality in Digital UNIX Version 4.0 comes primarily from OSF Version 1.0, although certain modules, like System V Release 4.0 STREAMS which were not available in OSF Version 1.0, have been taken from the OSF Version 1.2 code base. Some functionality, like IP Multicasting and the packet filter applications, has been taken from the public domain, enhanced, and integrated into the operating system as a service to our customers. The Network File System (NFS) code, as well as the Remote Procedure Calling (RPC) code, Network Information Service (NIS), and remote daemons and their corresponding commands came from Sun Microsystem's Open Network Computing (ONC) Version 4.2. And finally, functionality that Digital has licensed and enhanced, like Yellow Pages/Network Information Service which was licensed from SUN, was ported to Digital UNIX Version 4.0 from ULTRIX, since, although conforming to the OSF Version 1.2 standards, it was determined to be more robust than the corresponding code from the OSF.

Like all subsystems in Digital UNIX Version 4.0, the networking subsystem is designed to provide a standardized programming interface to enable third-party vendors to develop and port their networking applications to OSF with a minimum of difficulty. To this end Digital UNIX Version 4.0 supports the following:

- Internet Protocol Suite
  - Application Protocols
    - Domain Name Protocol (DOMAIN)
    - Routing Protocols (RIP, OSPF, EGP, and BGP)
    - File Transfer Protocol (FTP)
    - Network File System Protocol (NFS)
    - Telnet Protocol (TELNET)
    - Trivial File Transfer Protocol (TFTP)
    - Finger Protocol (FINGER)
    - Simple Mail Transfer Protocol (SMTP)

        * Simple Network Manager Protocol (SNMP)
    - Transport Protocols
        * User Datagram Protocol (UDP)
        * Transmission Control Protocol (TCP)
    - Network Level Protocols
        * Internet Protocol (IP)
        * Address Resolution Protocol (ARP)
        * Internet Control Message Protocol (ICMP)
        * Internet Group Management Protocol (IGMP)
- Supported Networks
    - ATM
    - Ethernet
    - Fast Ethernet
    - FDDI
    - Token Ring
- Application Programming Interfaces
    - X/Open Transport Interface (XTI/TLI)
    - BSD 4.3 Sockets
    - System V Release 4.0 STREAMS
    - Data Link Interface (DLI)
    - Data Link Provider Interface (DLPI)
    - Extensible SNMP (eSNMP)
- Network Administration Software
    - The entire suite of network commands and utilities from OSF Version 1.2
    - Ethernet, FDDI and loopback, packet filtering
    - Several popular packet filter applications in the public domain (`rarpd`, `tcpdump`, `tcpslice`, `nfswatch`, `nfslogsum`)
    - The `screend` security policy daemon (developed at Digital)
    - UUCP from HoneyDanBer
    - Local Area Transport (LAT)
    - Dynamic Host Configuration Protocol (DHCP)

- Naming Services
  - Berkeley Internet Name Domain (BIND)
  - Yellow Pages/Network Information Services (YP/NIS)
- Time Services
  - Network Time Protocol (NTP)
  - Time Synchronization Protocol (TSP)

The following sections briefly discuss the networking functionality in Digital UNIX Version 4.0.

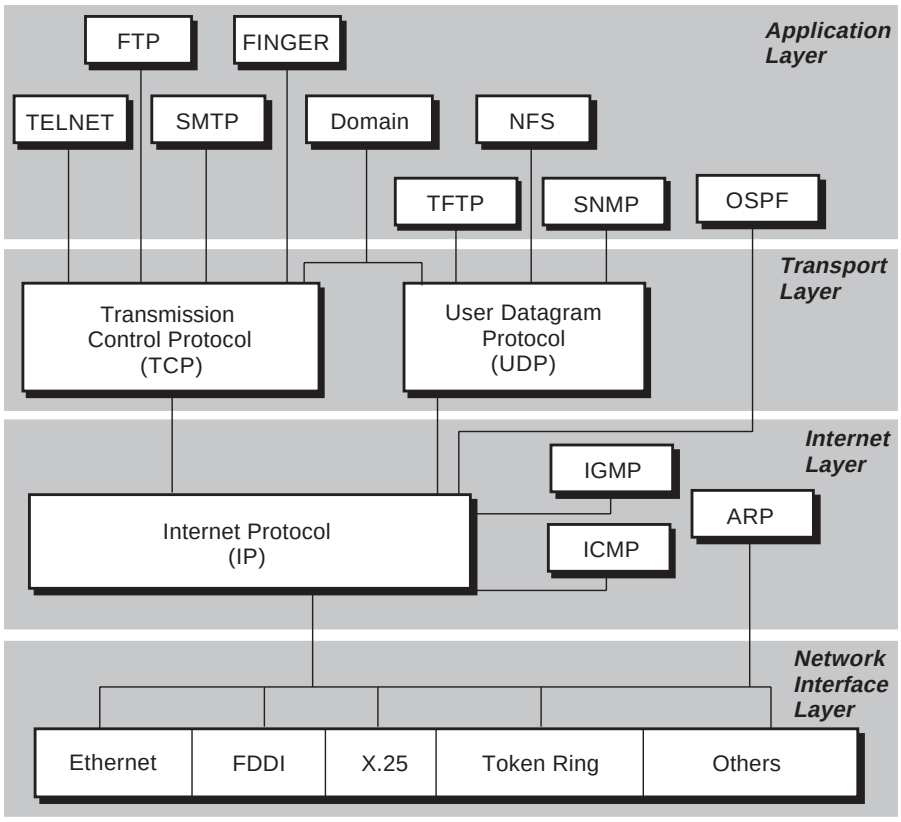
3.2 The Internet Protocol Suite

TCP/IP supports a suite of protocols, each of which provides a different service. These protocols allow networking communications to be independent of network hardware. The TCP/IP protocol suite is organized into the following groups:

- Application-Level Protocols, such as DOMAIN, Gateway Protocols (EGP, BGP, RIP, and OSPF), File Transfer Protocol (FTP), FINGER, TELNET, Trivial File Transfer Protocol (TFTP), Simple Mail Transfer Protocol (SMTP), and Simple Network Management Protocol (SNMP).
- Transport-Level Protocols, such as User Datagram Protocol (UDP) and Transmission Control Protocol (TCP)
- Network-Level Protocols, such as Address Resolution Protocol (ARP), Internet Control Message Protocol (ICMP), and Internet Protocol (IP)

Figure 3-1 illustrates the relationship of the major protocols in the TCP/IP suite.

Figure 3-1: TCP/IP Protocols



Applications programs send messages (streams of data) to the Internet Transport-Level Protocols, which are the UDP and the TCP. These protocols receive the data from the application, divide it into packets, add a transport header, and then pass the packets along to the next protocol layer, the Internet layer.

The Internet layer encloses the packet in an IP datagram, adds the datagram header, decides where to send the datagram (either directly to a destination or else to a gateway), and passes the datagram on to the Network Interface layer. The Network Interface layer accepts IP datagrams and transmits them as frames over a specific network hardware.

Frames received by a network go through the protocol layers in reverse. Each layer strips off the corresponding header information until the data is back at the application level. Frames are received by the Network Interface layer (for example, an Ethernet adapter), which strips off the physical layer header and sends the datagram to the Internet layer. In the Internet layer, the Internet Protocol strips off the IP header and sends the packet to the Transport layer. The Transport layer strips off the TCP or UDP header and sends the data up to the Application layer.

3.2.1 Application-Level Protocols

When an application needs to send data to an application on another host, the application sends the information down to the transport level protocols to prepare the information for transmission. These protocols include DOMAIN, EGP, BGP RIP, OSPF, FTP, NFS, TELNET, TFTP, FINGER, SMTP, and SNMP.

3.2.1.1 Domain Name Protocol

The Domain Name Protocol (DOMAIN) allows one or more hosts in a domain to act as a name server for other hosts within the domain. DOMAIN uses UDP or TCP as its underlying protocol and allows a local network to assign host names within its domain independently from other domains. UDP is the preferred protocol for use with DOMAIN; however, if the UDP response is truncated, TCP can be used.

In the Digital UNIX environment, the Berkeley Internet Name Domain (BIND) naming service uses the Domain Name Protocol. In this hierarchical naming system, local resolver routines may resolve Internet names and addresses using a local name resolution database maintained by the `named` daemon. If the name requested by the host is not in the local database, the resolver routine or the local `named` daemon queries the remote BIND name server.

3.2.1.2 Routing Protocols

Routing Protocols allow systems on either internal or external LANs to share routing information. In addition to the somewhat outmoded External Gateway Protocol (EGP), Digital UNIX supports the Border Gateway Protocol (BGP) and both the Routing Information Protocol (RIP) and Open Shortest Path First Protocol (OSPF) as part of the `GateD` daemon from Cornell University (for more information on the `GateD`, see Section 3.5.6).

3.2.1.2.1 Exterior Gateway Protocol (EGP) – The Exterior Gateway Protocol (EGP) allows the exterior gateway of an autonomous system to share routing information with exterior gateways on other autonomous systems.

An autonomous system is a group of networks and gateways for which one administrative authority has responsibility. Gateways are interior neighbors if they reside on the same autonomous system and exterior neighbors if they reside on different autonomous systems. Gateways that exchange routing information using EGP are said to be EGP peers (neighbors). Autonomous system gateways use EGP to provide reachability information to their EGP neighbors.

EGP allows an exterior gateway to provide remote communications among systems as follows:

- Ask another exterior gateway to agree to exchange reachability information
- Continually check to ensure that its EGP neighbors are responding
- Allow EGP neighbors to exchange reachability information by passing routing update messages

EGP restricts exterior gateways by allowing them to advertise only those destination networks reachable entirely within that gateway's autonomous system. Thus, an exterior gateway using EGP passes on information to its EGP neighbors, but does not advertise reachability information about its EGP neighbors.

EGP does not interpret the distance metrics that appear in routing update messages from other protocols. EGP uses the distance field to specify whether a path exists (a value of 255 means that the network is unreachable). The value cannot be used to compute the shorter of two routes, unless those routes are both contained within a single autonomous system. For this reason, EGP cannot be used as a routing algorithm. As a result, there is only one path from an exterior gateway to any network.

EGP routes are predetermined in the `/etc/gated.conf` file. This contrasts with the Routing Information Protocol (RIP), which can be used within (that is, interior to) an autonomous system of Internet networks that dynamically reconfigure routes. EGP assumes that IP is the underlying protocol. See the `gated`(8) reference page for further information.

3.2.1.2.2 Border Gateway Protocol – The Border Gateway Protocol (BGP) is an exterior routing protocol used for exchanging routing information between autonomous systems that are either multiple transit autonomous systems or transit and stub autonomous systems. BGP is related to EGP but operates with more capability, greater flexibility, and less required bandwidth. For example, BGP uses path attributes to provide more information about each

route and, unlike EGP, maintains an Autonomous System (AS) path, which provides enough information (such as the AS number of each autonomous system the route has traversed) to prevent routing loops in an arbitrary topology.

Like EGP, BGP supports both internal and external sessions. When sending routes to an external peer, BGP prepends the local AS number to the AS path so that routes received from an external peer are guaranteed to have the AS number of that peer at the start of the path. Routes received from an internal neighbor will not in general have the local AS number prepended to the AS path, and in general have the same AS path that the route had when the originating internal neighbor received the route from an external peer. Routes with no AS numbers in the path may be legitimately received from internal neighbors; these indicate that the received route should be considered internal to your own AS.

The Digital UNIX implementation of BGP supports three versions of the BGP protocol (versions 2, 3 and 4). BGP versions 2 and 3 are quite similar in capability and function. They will only propagate classed network routes, and the AS path is a simple array of AS numbers. BGP 4 will propagate fully general address-and-mask routes, and the AS path has some structure to represent the results of aggregating dissimilar routes.

3.2.1.2.3 Routing Information Protocol (RIP) – The Routing Information Protocol (RIP) is an implementation of a distance-vector, or Bellman-Ford routing protocol for local networks and in Digital UNIX is contained in the `GateD` daemon from Cornell University. RIP classifies routers as active and passive: active routers advertise their routes to other routers; passive routers listen and update their routes based on the advertisements they receive, but do not advertise themselves. Typically, routers run RIP in active mode, while hosts use passive mode.

A router running RIP in active mode broadcasts updates at set intervals. Each update contains paired values where each pair consists of an IP network address and an integer distance to that network. RIP uses a hop count metric to measure the distance to a destination. The number of hops along a path from a given source to a given destination refers to the number of gateways that a datagram would encounter along that path.

For example, a router advertises directly connected networks as having a hop count of one. Networks that are reachable through another gateway are two hops away, networks that are reachable through two gateways are three hops away, and so forth. Then RIP chooses the path with the shortest hop count.

Of course, using hop counts to calculate shortest paths between networks may not always produce optimal results. For example, a path with a hop count of three that crosses three Ethernets may be substantially faster than a path with a hop count of 2 that crosses two slow-speed serial lines. To

compensate for differences in network and serial line rates of transfer, administrators can configure RIP routers to advertise artificially high hop counts for slow links.

3.2.1.2.4 Open Shortest Path First (OSPF) – Open Shortest Path Routing (OSPF) is a shortest path first (SPF) or link-state interior gateway protocol that distributes routing information between routers in a single autonomous system. Suitable for complex networks with a large number of routers, OSPF provides equal cost multipath routing whereby packets to a single destination can be sent by more than one network interface simultaneously.

A link-state protocol dictates that each router maintains a database describing the entire AS topology, which it builds out of the collected link state advertisements of all routers. Each participating router distributes its local state (that is the router's usable interfaces and reachable neighbors) throughout the AS by **flooding**. Each multiaccess network that has at least two attached routers has a designated router and a backup designated router. The designated router floods a link state advertisement for the multiaccess network and has other special responsibilities. The designated router concept reduces the number of adjacencies required on a multiaccess network.

OSPF allows networks to be grouped into areas. Routing information passed between areas is abstracted, potentially allowing a significant reduction in routing traffic. OSPF uses four different types of routes, listed in order of preference: intra-area, inter-area, type 1 external and type 2 external. Intra-area paths have destinations within the same area, inter-area paths have destinations in other OSPF areas and Autonomous System External (ASE) routes are routes to destinations external to the AS. Routes imported into OSPF as type 1 routes are supposed to be from EGPs whose external metrics are directly comparable to OSPF metrics. When a routing decision is being made, OSPF will add the internal cost to the AS Border router to the external metric. Type 2 ASEs are used for EGPs whose metrics are not comparable to OSPF metrics. In this case, only the internal OSPF cost to the AS Border router is used in the routing decision.

From the topology database, each router constructs a tree of the shortest paths with itself as the root. This shortest-path tree gives the route to each destination in the AS. Externally derived routing information appears on the tree as leaves. The link-state advertisement format distinguishes between information acquired from external sources and information acquired from internal routers, so there is no ambiguity about the source or reliability of routes. Externally derived routing information (for example, routes learned from EGP or BGP) is passed transparently through the autonomous system and is kept separate from OSPF's internally derived data. Each external route can also be tagged by the advertising router, enabling a passing of additional information between routers on the borders of the autonomous system.

3.2.1.3 File Transfer Protocol

File Transfer Protocol (FTP) allows hosts to transfer files. FTP provides for such tasks as listing remote directories, changing the current remote directory, creating and removing remote directories, and transferring multiple files in a single request. FTP maintains a secure transport by passing user and account passwords to the foreign host. FTP allows interactive user-oriented sessions.

FTP uses reliable stream transport (TCP/IP) to send the files and uses a TELNET-like connection to transfer commands and replies. FTP also understands several basic file formats, including ASCII, IMAGE, and Local 8. TCP/IP implements FTP in the `ftp` user command and the `ftpd` server command.

3.2.1.4 Network File System Protocol over UDP transport

The Network File System (NFS) provides access to files via standard UNIX system calls. This allows any program to access files across the network. NFS uses the UDP transport layer; therefore, it has to deal with lost datagrams. NFS does this by retransmitting requests if a reply has not been received within a reasonable amount of time. Some requests can be re-executed on the server without problems, but others (such as file deletion) cause an error if the first request reaches the server but the reply is lost. If the second request is executed, the server finds that the file does not exist and returns an error. NFS servers hold on to such replies and retransmit them if they see a duplicate request.

On the other hand, the protocol is designed so that the servers need no other state information. This allows server performance to be improved by running multiple copies of the server daemon, and also means that server crashes are tolerated with no special code on either client or server.

For more information on NFS, see Chapter 4.

3.2.1.5 Network File System Protocol over TCP transport

Digital UNIX Version 4.0 contains Digital's first release of NFS support over the TCP transport. Previously, only the UDP transport was available. UDP may still be the preferred transport in local area networks, but for NFS access over wide area, congested, or lossy networks, TCP may offer better performance.

Separate threads are used to maintain some of the performance optimizations made to the UDP code paths. The `nfsiod` daemon has been changed to spawn kernel threads, instead of forking multiple processes as it did previously. Each `nfsiod` thread can handle UDP or TCP mounts, so the `nfsiod` command still accepts one argument.

For more information on NFS, see Chapter 4.

3.2.1.6 Telnet Protocol

The Telnet Protocol (TELNET) provides a standard method for terminal devices and terminal-oriented processes to interface. TELNET is commonly used by terminal emulation programs that allow you to log in to a remote host. However, TELNET can also be used for terminal-to-terminal communications and interprocess communications. TELNET is also used by other protocols (for example, FTP) for establishing a protocol control channel.

TCP/IP implements TELNET in the `telnet` user command and the `telnetd` server command.

3.2.1.7 Trivial File Transfer Protocol

The Trivial File Transfer Protocol (TFTP) can read and write files to and from a foreign host. Like FTP, TFTP can transfer files as either 8-bit NETASCII characters or as 8-bit binary data. Unlike FTP, TFTP cannot be used to list or change directories at a foreign host and it has no provisions for security, such as password protection. Data normally can be written or retrieved only in public directories.

TCP/IP implements TFTP in the `tftp` user command and in the `tftpd` server command.

3.2.1.8 Finger Protocol

The Finger Protocol (FINGER) is an application-level Internet protocol that provides an interface between the `finger` command and the `fingerd` daemon. The `fingerd` daemon returns information about the users currently logged in to a specified remote host. If you execute the `finger` command specifying a user at a particular host, you obtain specific information about that user. The Finger Protocol must be present at the remote host and at the requesting host. FINGER uses TCP as its underlying protocol.

3.2.1.9 Simple Mail Transfer Protocol

The Simple Mail Transfer Protocol (SMTP) is the standard for mail exchange between machines attached to the Internet. It specifies the format of control messages sent between two machines to exchange electronic mail.

As its name implies, SMTP is simple in design and purpose. Its objective is to provide a reliable and efficient mail delivery system across the links between machines. SMTP does not specify the mail interface.

3.2.1.10 Simple Network Management Protocol

The Simple Network Management Protocol (SNMP) is the Internet standard protocol for exchanging network management information. The SNMP agent provides a local or remote network manager with system information, network interface data, address resolution information (ARP), information about the routing layer (IP and ICMP), and information about the transport layer (TCP and UDP). Digital UNIX SNMP agent also permits management of basic host information, such as processes, file systems, memory, attached devices, and so forth.

3.2.2 Transport-Level Protocols

The TCP/IP transport-level protocols (UDP and TCP) allow application programs to communicate with other application programs. The User Datagram Protocol (UDP) and the Transmission Control Protocol (TCP) are the basic transport-level protocols for making connections between Internet hosts. When an application sends a message to the transport layer, UDP and TCP break the information into packets, add a packet header including the destination address, and send the information to the network layer for further processing.

Other protocols and applications use UDP to make datagram connections and TCP to make stream connections. The socket interface implements these protocols.

3.2.2.1 User Datagram Protocol

UDP provides a datagram means of communication between applications on Internet hosts. UDP uses destination protocol ports (abstract destination points within a machine), identified by positive integers, to send messages to one of multiple destinations on a host. The protocol ports receive and hold messages in queues until applications on the receiving host can retrieve them.

UDP relies on the underlying IP to send its datagrams and provides the same connectionless message delivery as IP. It offers no assurance of datagram delivery or duplication protection. However, UDP allows the sender to specify source and destination port numbers for the message and also calculates a checksum of both the data and header. These two features allow the sending and receiving applications to ensure the correct delivery of a message.

3.2.2.2 Transmission Control Protocol

TCP provides reliable stream delivery of data between Internet hosts. Like UDP, TCP uses IP, the underlying protocol, to transport datagrams, and supports the block transmission of a continuous stream of datagrams between process ports. Unlike UDP, TCP provides reliable message delivery and

ensures that data is not damaged, lost, duplicated, or delivered out of order to a receiving process. Because of this transport reliability, applications programmers are not required to build communications safeguards into their software.

Both TCP and UDP allow programs to send messages to and receive messages from applications on other hosts, and both use protocol ports on the host to identify the specific destination of the message. The TCP retransmission time-out value is dynamically determined for each connection, based on round-trip time.

TCP has the following operational characteristics:

- Basic Data Transfer

  TCP transfers a continuous stream of 8-bit octets in each direction between its users by packaging some number of bytes into segments for transmission through the Internet system. In general, TCP determines the best time to block and forward packets.

- Reliability

  TCP recovers data that is damaged, lost, duplicated, or delivered out of order by the Internet. To achieve this reliability, TCP assigns a sequence number to each octet it transmits, and requires a positive acknowledgment (ACK) from the receiving TCP. If the ACK is not received within the time-out interval, the data is retransmitted. At the receiver, the sequence numbers are used to correctly order segments that are received out of order and to eliminate duplicates. To detect damage, TCP adds a checksum to each segment transmitted, checks it at the receiver, and discards damaged segments.

- Flow Control

  To control how much data is sent, TCP returns the following information with every acknowledgment:

  – The sequence numbers it will accept next; these numbers are always greater than the number of the last segment that was successfully received.

  – The number of octets that the sender is allowed to transmit before receiving further permission.

- Multiplexing

  Many processes within a single host can use TCP communications facilities simultaneously. TCP maintains a set of addresses (ports) within each host. TCP combines the port number with the network address and the host address to uniquely identify each connection endpoint (socket). A pair of sockets uniquely identifies each connection.

- Connections

  TCP must initialize and maintain certain status information for each data stream. The combination of this information, including sockets, sequence numbers, and window sizes, is called a connection. Each connection is unique.

3.2.3 Network-Level Protocols

The Internet network-level protocols (IP, ARP, ICMP) handle machine-to-machine communications. These protocols provide for transmission and reception of transport requests, and handle network-level control.

3.2.3.1 Internet Protocol

The Internet Protocol (IP) is the primary network-level protocol and provides unreliable, connectionless packet delivery for the Internet. IP defines the format of all the data sent over the Internet. IP also specifies packet processing and error handling.

IP is connectionless because it treats each packet independently. It is unreliable because it does not guarantee delivery or the order of arrival of packets. However, underlying mechanisms guarantee data integrity, assuming it arrives.

IP provides the interface to the network interface level protocols. The physical connections of a network transfer information in a frame with a header and data. IP uses an Internet datagram, which contains a source host address, along with sequencing and control information.

IP automatically adds an IP header to outgoing packets and removes the IP header from incoming packets before sending them to higher level protocols. IP provides for the universal addressing of hosts in the Internet network.

IP is not a reliable communications facility because it does not require acknowledgments from the sending host, the receiving host, or intermediate hosts.

The total length of IP packets can be configured independently for each interface. Packets are broken up into smaller chunks at gateways and reassembled when they reach their destination.

3.2.3.1.1 IP Multicasting – Digital UNIX Version 4.0 supports IP Multicasting on a Local Area Network (LAN), as described in RFC 1112. Digital UNIX Version 4.0 also supports Version 3.5 of the IP Multicast Kernel support and Version 3.6 of the `mrouted` implementation of the Distance Vector Multicast Routing Protocol (DVMRP) which provides support for "tunnelling" and "pruning."

Unlike IP Broadcasting, IP Multicasting allows packets to be taken off the network only by those clients who have configured their systems to receive the packets. Packets are accepted or rejected at the hardware level, thereby greatly reducing processing overhead. In addition, IP Multicasting does not consume a lot of network bandwidth, since applications do not have to send separate packets with identical data to reach several destinations, as they do with point-to-point connections. With IP Multicasting, one packet is sent to all interested hosts.

As a result, IP Multicasting is very valuable to video conferencing applications and applications that provide constant updates to ever-changing information, like applications that provide stock market quotes.

The IP Multicasting code was taken from the public domain, integrated with DECnet and LAT, and is supported on all Ethernet and FDDI adapters.

3.2.3.1.2 Serial Line IP (SLIP) and Compressed Serial Line IP (CSLIP) – Digital UNIX Version 4.0 has complete IP support for a serial line, so that users can transfer files or NFS-mount file systems across phone lines. Using the CSLIP `slattach` option, headers can be compressed, thereby increasing performance.

The SLIP/CSLIP code is from OSF Version 1.0. However, since the OSF code did not provide a way to access the CSLIP feature, Digital modified the `slattach` command to provide the necessary access to CSLIP.

3.2.3.1.3 Point-to-Point Protocol (PPP) – Digital UNIX Version 4.0 supports the Point-to-Point (PPP) protocol (as defined in RFC 1144, 1171, 1172, 1331, 1332, 1334, 1548, 1549, 1661, and 1662) which provides a method for transmitting datagrams over serial point to point links. Unlike SLIP, PPP supports standard encapsulation, simultaneous multiplexing of different network layer protocols, an HDLC Frame Check Sequence for error detection, an HDLC escaping mechanism for use with miscellaneous non-8-bit-transparent telephone and switching equipment, and the dynamic negotiation of IP addresses.

In addition, while SLIP only supports `clist tty` drivers, PPP supports both `clist` and STREAMS-based `tty` drivers, as well as remote logins over LANs.

Note that the PPP code was taken from the public domain and includes contributions identified by the footnoted copyright notices [1]. Certain sections of the PPP code was derived from the RSA Data Security, Inc., MD5 Message-Digest Algorithm.

[1] Copyright (c) 1993 The Australian National University. Copyright (c) 1989 Carnegie Mellon University. Copyright (c) 1991 Gregory M. Christy. Copyright (c) 1989 Regents of the University of California. Copyright (c) 1990 RSA Data Security, Inc.

For more information on PPP, see the *System Administration* guide and the `pppd`(8), `pppstats`(8), and `chat`(8) reference pages.

3.2.3.2 Address Resolution Protocol

The Address Resolution Protocol (ARP) translates Internet addresses into hardware addresses. ARP does not translate addresses for the Serial Line Interface Protocol (SLIP) or Point-to-Point Protocol (PPP) because SLIP and PPP have no hardware address.

ARP dynamically traces Internet addresses to hardware addresses on local area networks. The result of this tracing is called a map. The mapped information is stored in mapping tables. TCP/IP uses ARP to collect and distribute the information for mapping tables.

The kernel maintains the mapping tables, and ARP is not directly available to users or applications. When an application sends an Internet packet to an interface driver, the driver requests the appropriate address mapping. If the mapping is not in the table, an ARP broadcast packet is sent through the requesting interface driver to the hosts on the local area network.

When a host that supports ARP receives an ARP request packet, the host notes the IP and hardware addresses of the requesting system and updates its mapping table, if necessary. If the receiving host's IP address does not match the requested address, the host ignores the request packet. If the IP address does match, the receiving host sends a reply packet to the requesting system. The requesting system stores the new mapping and uses it to transmit future Internet packets.

Unlike most protocols, ARP packets do not have fixed-format headers. Instead, the message is designed to be useful with a variety of network technologies.

3.2.3.3 Internet Control Message Protocol

The Internet Control Message Protocol (ICMP) is a required part of every IP implementation. ICMP handles error and control messages for IP.

ICMP does the following:

- Tests whether a destination is alive and reachable
- Reports parameter problems with a datagram header
- Performs clock synchronization and transit time estimations
- Obtains Internet addresses and subnet masks
- Provides transport-level reachability information
- Updates routing information

ICMP provides feedback about problems in the communications environment, but does not make IP reliable. That is, ICMP does not guarantee that an IP packet will be delivered reliably or that an ICMP message will be returned to the source host when an IP packet is not delivered or is incorrectly delivered.

ICMP messages are sent in varying situations, including the following:

- When a packet cannot reach its destination
- When a gateway host does not have the buffering capacity to forward a packet
- When a gateway can direct a host to send traffic on a better route

3.3 Supported Networks

Digital UNIX Version 4.0 supports interfacing to the following networks:

- ATM
- Ethernet
- Fast Ethernet
- FDDI
- Token Ring

3.3.1 ATM

Digital UNIX Version 4.0 supports PCI and TURBOchannel machines on 155.5 MB per second Asynchronous Transfer Mode (ATM) networks. ATM is a high-speed, connection-based, cell-switched network that—unlike traditional packet switched networks—can carry different kinds of traffic (voice, video, and data) simultaneously. In addition, ATM provides predictable services to those classes of traffic that require bounded latencies and dedicated bandwidths, and—because ATM separates the physical interface from the datalink layer—the same cell and packet formats can be used over a wide variety of physical interfaces from 1 MB per second to 10 GB per second.

Digital UNIX Version 4.0's implementation of ATM consists of permanent virtual circuit support; switched virtual circuit support through ATM Forum UNI 3.0 and 3.1 signalling for point-to-point connections; Classic IP (as defined by RFC 1577, RFC 1483, and RFC 1626) for a single Logical IP Subnet (LIS); and ATM Forum Interim Local Management Interface (ILMI) for dynamic network address registration. For more information on ATM, see the *Network Programmer's Guide*.

3.3.2 Ethernet

Digital UNIX Version 4.0 supports 10 MB per second Ethernet networks on all Alpha platforms.

At the physical and IP levels, Digital UNIX Version 4.0 supports an Ethernet network with a Maximum Transfer Unit (MTU) of 1500 bytes at a maximum of 10 MB per second.

In conformance with Ethernet standards, Digital UNIX Version 4.0 always ensures a minimum packet size of 60 bytes.

The default MTU at the IP level is 1500 bytes at a maximum of 10 MB per second, although this value can be decreased using the `ifconfig` command.

3.3.3 Fast Ethernet

Digital UNIX Version 4.0 supports 100 MB per second Fast Ethernet (IEEE 802.3 100Base-TX) networks on all PCI-based Alpha hardware platforms.

MTU sizes at the physical and IP levels are the same as those for regular 10 MB per second Ethernet.

3.3.4 FDDI

Digital UNIX Version 4.0 supports 100 MB per second FDDI networks in conformance with RFC 1042 and RFC 1188 on all Alpha hardware platforms.

At the physical level, Digital UNIX Version 4.0 supports an FDDI network with a Maximum Transfer Unit (MTU) of 4500 bytes at a maximum of 100 MB per second. At the IP level, the MTU is 4352 bytes at a maximum of 100 MB per second.

The default MTU at the IP level is always 4352 bytes at a maximum of 100 MB per second, although this value can be decreased using the `ifconfig` command.

3.3.5 Token Ring

Digital UNIX Version 4.0 supports 4 MB per second and 16 MB per second Token Ring networks and source routing in conformance with RFC 1042 on TURBOchannel, ISA, EISA, and PCI-based Alpha hardware platforms.

Support for Token Ring networks extends networking support to the PC community, since most PC networks are Token Ring and most PCs do not have Ethernet or FDDI adapters.

At the physical level, Digital UNIX Version 4.0 supports a Token Ring network with a Maximum Transfer Unit (MTU) of 4472 bytes at a maximum of 4 MB per second and 17800 bytes at a maximum of 16 MB per second. At the IP level, the MTU is 4092 bytes at a maximum of 4 MB per second and 8188 bytes at a maximum of 16 MB per second.

The default MTU at the IP level is always 4092 for both 4 and 16 MB per second, although this value can be increased or decreased using the `ifconfig` command.

3.4 Application Programming Interfaces

The network programming environment includes the programming interfaces for application, kernel, and driver developers writing network applications and implementing network protocols. Additionally, it includes the kernel level resources that an application requires to process and transmit data, some of which include libraries, data structures, header files, and transport protocols.

This section briefly discusses the following application programming interfaces that are supported in Digital UNIX Version 4.0:

- X/Open Transport Interface (XTI/TLI)
- BSD Sockets
- System V Release 4.0 STREAMS
- Data Link Interface (DLI)
- Data Link Provider Interface (DLPI)
- Extensible SNMP (eSNMP)

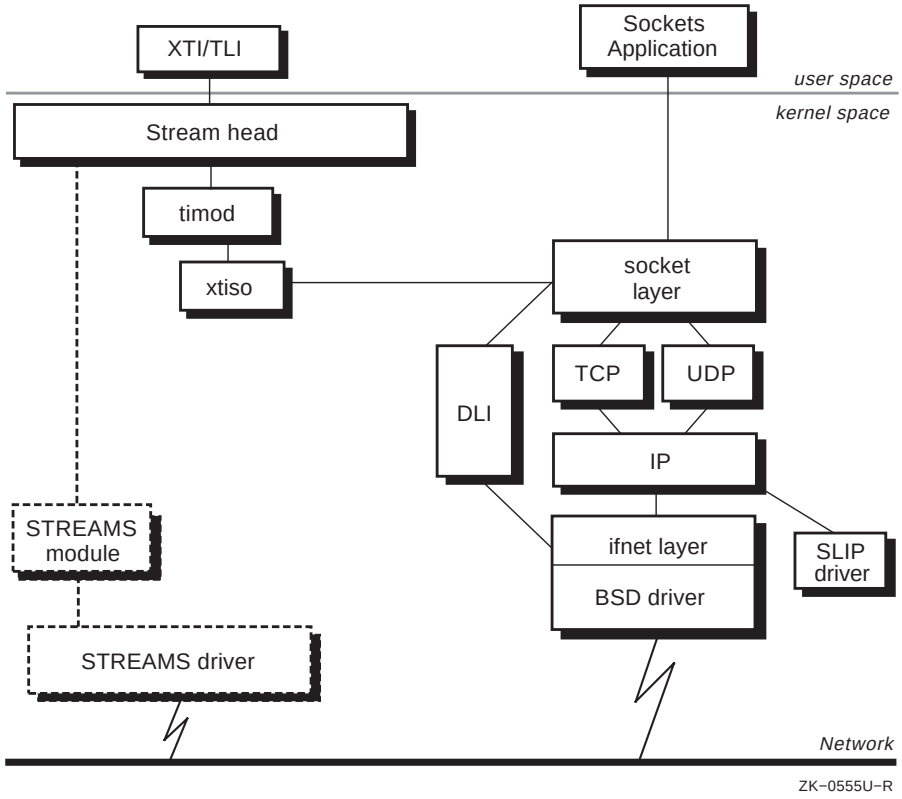
For more detailed information on the network programming environment, see the *Network Programmer's Guide*.

3.4.1 X/Open Transport Interface

The X/Open Transport Interface (XTI) defines a transport layer application interface that is independent of any transport provider. This means that programs written to XTI can be run over a variety of transport providers, such as the Transmission Control Protocol (TCP) or the User Datagram Protocol (UDP). The application specifies which transport provider to use.

Because XTI provides an interface that is independent of a transport provider, application developers are encouraged to write programs to XTI instead of STREAMS or sockets. Figure 3-2 illustrates the interaction between XTI and the STREAMS and sockets frameworks.

Figure 3-2: XTI, STREAMS and Sockets Interactions



Depending on the transport provider specified by the application, data can flow along one of two paths:

1. If a STREAMS-based transport provider is specified, data follows the same route that it did for an application written to run over STREAMS. It passes first through the Stream head, then to any modules that the application pushed onto the Stream, and finally to the STREAMS driver, which puts it on to the network. Digital UNIX Version 4.0 does not provide any STREAMS-based transport providers.
2. If a socket-based transport provider (TCP or UDP) is specified, data is passed through `timod` and `xtiso`. The appropriate socket layer routines are called and the data is passed through the Internet protocols and `ifnet` layer to the BSD-based driver, which puts it on to the network.

3.4.2 Sockets

Sockets are the industry standard programming interface. Digital UNIX Version 4.0 implements the socket interface for both 4.3BSD and *X/Open CAE Specification, Networking Services, Issue 4* interfaces. Using the `_SOCKADDR_LEN` option to the `connect` system call, however, you can access the 4.4BSD interface. For more information, see the `connect`(2) reference page.

The sockets framework consists of a series of system and library calls, header files, and data structures. Applications can access kernel-resident networking protocols, such as the Internet Protocol suite, through socket system calls. Applications can also use socket library calls to manipulate network information, for example, mapping service names to service numbers or translating the byte order of incoming data to that appropriate for the local system's architecture. The Internet Protocol suite, for example, which consists of TCP, UDP, IP, ARP, ICMP, and SLIP is implemented over sockets.

With sockets, the application in user space passes data to the appropriate socket system calls, which then pass it to the network layer. Finally, the network layer passes it, via the `ifnet` layer, to the BSD driver, which puts it on to the network. For more information, on sockets, see RFC 1200: *IAB Protocol Standards*, the *Network Programmer's Guide*, and the *X/Open CAE Specification, Networking Services, Issue 4*.

3.4.3 STREAMS

The STREAMS framework provides an alternative to sockets. The STREAMS interface was developed by AT&T and consists of system calls, kernel routines, and kernel utilities that are used to implement everything from networking protocol suites to device drivers. Applications in user space access the kernel portions of the STREAMS framework using system calls such as `open`, `close`, `putmsg`, `getmsg` and `ioctl`. Digital UNIX Version 4.0 supports System V Release 4.0 STREAMS from the OSF Version 1.2 code base, which provides support for the STREAMS `tty` interface (although Digital UNIX Version 4.0 continues to support the existing CLIST or Berkeley-based `tty` interface). For more information on STREAMS, see the *Network Programmer's Guide*.

3.4.4 Sockets and STREAMS Interaction

Digital UNIX Version 4.0 provides the `ifnet` STREAMS module to allow programs using Digital UNIX Version 4.0's BSD-based TCP/IP to access STREAMS-based drivers. It provides the Data Link Bridge (DLB) pseudodriver to allow programs using a STREAMS-based protocol stack to access BSD-based drivers provided on Digital UNIX Version 4.0.

3.4.5 Data Link Interface (DLI)

DLI is provided on Digital UNIX Version 4.0 as a backward compatibility feature to ULTRIX. DLI support on Digital UNIX Version 4.0 allows programs written to DLI on the ULTRIX operating system to access the data link layer. For more information on DLI, see the *Network Programmer's Guide*.

3.4.6 Data Link Provider Interface (DLPI)

DLPI is a kernel level interface that maps to the data link layer of the OSI reference model. DLPI frees its users from specific knowledge of the characteristics of the data link provider, allowing those characteristics to be implemented independently of a specific communications medium. It is primarily a kernel-level interface targeted for STREAMS protocol modules that either use or provide data link services.

Only a partial subset of the DLPI interface is supported in Digital UNIX Version 4.0. For more information, see the *Network Programmer's Guide*.

3.4.7 Extensible SNMP Interface (eSNMP)

Digital UNIX supports extensible SNMP (eSNMP), an application-layer Application Programming Interface (API) that permits user-written programs to function as part of a distributed SNMP agent on a Digital UNIX host system.

User programs can dynamically register SNMP MIB objects with the eSNMP master agent (`/usr/sbin/snmpd`), and subsequently handle the SNMP protocol operations for those objects.

The distribution of MIB objects between cooperating processes is transparent to SNMP applications, which can access all MIB objects using the standard transport endpoints specified in the SNMP RFCs.

For more information, see the *Network Programmer's Guide*.

3.5 Network Administration Software

Digital UNIX Version 4.0 supports a variety of network administration software which is briefly described in the following sections.

3.5.1 Networking Commands and Utilities

Digital UNIX Version 4.0 supports the entire suite of networking commands from OSF Version 1.2, including: `gated`, `finger`, `ftp`, `rdump`, `rdist`, `routed` and the complete suite of remote commands, `snmp`, `smtp`, `telnet`, and `tftp`. The `bootpd` command has been folded into the

Digital-specific `joind` command which provides configurations to clients using either the DHCP or BOOTP protocol. Additionally, Digital UNIX Version 4.0 supports the following Open Network Computing (ONC) Version 4.2 utility programs, which can be invoked by the `inetd`:

- `rwall/rwalld`
- `rusers/rusersd`
- `spray/rsprayd`
- `rup/rstatd`
- `rquotad`
- `pcnfsd`

3.5.2 Ethernet Packet Filter and Packet Filter Applications

The Ethernet packet filter is a software driver interface that provides demultiplexing of networking packet headers, as well as reception and transmission of packets containing user-defined network protocols. The packet filter can also function as an Ethernet monitor when used to filter specific network protocols.

Digital UNIX Version 4.0 supports the following packet filter applications:

- `/usr/sbin/rarpd` – Reverse ARP daemon

  The reverse ARP daemon responds to RARP requests on a network by sending an IP address to a host which only knows its Ethernet address. It uses the `/etc/ethers` file to map the Ethernet address to an IP address.

  The reverse ARP daemon can serve IP addresses to remote PC clients. Also, some customers are using ULTRIX on DECstations today and rely on the Reverse ARP protocol to supply remote stations with their IP address. If they want to serve these addresses using a Digital UNIX Version 4.0 server, they can do so with the `rarpd` daemon.

- `/usr/sbin/tcpdump` – TCP/IP tracing and monitoring tool

  Digital UNIX Version 4.0 supports Version 2.2.1 of the `tcpdump` utility. This version of `tcpdump` uses the Berkeley Packet Filter (BPF) language.

  The `tcpdump` utility is used to debug and analyze TCP/IP network activity, on both Ethernet and FDDI networks, and has some support for other protocol suites (including NFS). This product includes software developed by the University of California, Lawrence Berkeley Laboratory and its contributors.

- `/usr/sbin/tcpslice` – Log file tool

  The `tcpslice` utility manipulates `tcpdump` trace log files by either extracting pieces of or glueing together `tcpdump` log files. It can select portions of a large `tcpdump` log file and display selected traces without having to dump the entire log file.

- `/usr/sbin/nfswatch` – NFS monitoring tool

  Digital UNIX Version 4.0 supports Version 4.1 of `nfswatch` from Purdue University. The `nfswatch` utility is curses-based and displays the NFS traffic between any number of NFS servers and clients on a LAN.

- `/usr/sbin/nfslogsum` – NFS log file summary tool

  The `nfslogsum` utility condenses the log files produced by `nfswatch` into a traffic analysis summary and is very helpful in troubleshooting networks.

Note

Since the packet filter is an optional kernel subsystem, application programs that make calls to the packet filter kernel routines may fail if the packet filter is not configured in the currently running kernel. For more information, see the `packetfilter`(7) reference page.

3.5.3 Dynamic Host Configuration Protocol

Digital UNIX Version 4.0 supports the Dynamic Host Configuration Protocol (DHCP), a client/server framework in which the DHCP server can dynamically assign an IP address to a client as the client boots onto the network. Additionally, a DHCP server can provide configuration information to the client, such as the name of the BIND server or the name of the default router for that client.

For example, when a new system is booted for the first time, the DHCP server assigns that system a unique IP address; if that system is moved to another location on the same LAN (perhaps on a different subnet), the DHCP server ensures that a new IP address appropriate to that subnet is assigned to the system, if necessary, when it boots up for the first time.

As a result, with DHCP, customers with hundreds of clients no longer have to worry about the assignment of IP addresses; DHCP assigns IP addresses automatically and requires no intervention by a system administrator.

For more information on DHCP, see the *Network Administration* and the `dhcp`(7) reference page.

3.5.4 The Internet Boot Protocol Daemon (bootpd)

The `bootpd` daemon implements an Internet Boot Protocol server as defined in RFC 951, RFC 1532, and RFC 1533.

BOOTP is an extensible UDP/IP-based protocol that allows a booting host to configure itself dynamically without having to rely on user intervention. The BOOTP protocol assigns IP addresses to hosts, makes available a file containing a boot program that can be downloaded from a server, provides the address of that server, and if present the address of an Internet gateway.

Like DHCP, the BOOTP protocol supports the centralized management of network addresses.

3.5.5 SNMP Agent

The extensible SNMP agent in Digital UNIX Version 4.0 permits the dynamic addition of supported Management Information Bases (MIBs) on any Digital UNIX host. The MIB support that ships as part of the operating system allows management operations on the objects described in the following RFCs:

- Internet MIB (RFc 1213)
- FDDI MIB (RFC 1285)
- Token Ring MIB (RFC 1231)
- Host Resources MIB (RFC 1514)
- Various routing MIBs as described in Section 3.5.6.

The master agent, API, and base operating system MIB support are all contained in the standard networking subset (CLINET).

The extensible SNMP development tools are contained in the optional programming subset (PGMR).

3.5.6 The gated Daemon

The `gated` daemon allows any host with multiple network interfaces to function as an IP router by participating in various IP routing protocols (for example, RIP, OSPF, EGP, and BGP). Digital UNIX Version 4.0 supports the GateD Release 3.5 `gated` daemon from the Gatedaemon Project at Cornell University, which contains support for the following:

- RIP Version 1 (RFC 1058)

  Stipulates the proper subsuming of host routes, split horizon without poison reverse, and graceful shutdowns.

- RIP Version 2 (RFC 1388)

  Stipulates using IP Multicast where available; supports classless routing;

uses next hop (if different).

- OSFP Version 2 (RFC 1247)

  Uses local-wire IP Multicast support, MIB support (RFC 1253), and reconfiguration.

- Support for Routing Table MIB (RFC 1354)
- EGP 2 (RFC 904)

  A complete implementation of the specification, with optimizations for MILNET.

- BGP Versions 2 and 3 (RFC 1163 and RFC 1267)

  Complete implementations of specifications; BGP MIB (RFC 1269); AS path pattern matching RFC 1164); and OSPF/BGP Interaction (RFC 1403).

- BGP Version 4 (RFC 1654)
- DCN HELLO

  Proper subsuming of host routes; split horizon without poison reverse.

- Variable subnet masks through Routing Socket Support and improved synchronization of the kernel routing table
- Routing Table Enhancements

  Based on BSD 4.3 Reno radix tree, `gated` implements filtered routing based on policy. This allows network administrators to control the import and export of routing information by individual protocol, by source and destination autonomous system, source and destination interface, previous hop router, and specific destination address.

  Network administrators can also specify a preference level for each combination of routing information being imported by using a flexible masking capability. Once the preference levels are assigned, `gated` decides which route to use independent of the protocols involved.

- MIB Support for the Following Protocols ("Get Object" Support Only):
  - OSPF V2 MIB (RFC 1253)
  - EGP-MIB (RFC 1213)
  - BGP V3 MIB (RFC 1269)

For more information on `gated`, see the `gated`(8) and `gated.conf`(8) reference pages. Also, for a complete description of the `gated.conf` options, see the *EGate Daemon Configuration Guide*, which is accessible from the Digital UNIX Documentation Library page on the Digital UNIX Documentation CD-ROM.

3.5.7 The screend Daemon

The `screend` daemon is used in conjunction with the gateway screen facility to decide which IP packets should be forwarded when the system is acting as an IP gateway.

The gateway packet screening facility, on a Digital UNIX system acting as a gateway, allows the system manager to control which packets are forwarded or rejected. As a result, the gateway packet screening facility can be used as one part of a comprehensive network security policy. The facility consists of a kernel-resident mechanism and a user-level daemon, `/usr/sbin/screend`. When a packet is ready to be forwarded, the kernel mechanism submits the packet's headers to the daemon. The `screend` daemon then examines the headers and tells the kernel to forward or reject the packet, based on a set of rules defined in the configuration file, `/etc/screend.conf`. Optionally, some or all decisions can be logged allowing the network manager to detect improper configurations or potential security problems.

3.5.8 UNIX-to-UNIX Copy Program

The UNIX-to-UNIX Copy Program (UUCP) program is actually a group of programs that supports communications between two computers running UNIX operating systems.

DEC OSF/1 supports the HoneyDanBer version of UUCP. The UUCP system enables batched, error-free file transfer and remote command execution between two UNIX systems. The UUCP system is most frequently used to transfer electronic mail, network news, and public domain software over low-speed, low-cost communications links.

A worldwide network that functions through the informal cooperation of the user community has grown up around UUCP. The UUCP network is a series of point-to-point links, with the majority of sites located in Europe and North America.

The UUCP protocol supports only direct connections between two systems. However, electronic news and mail delivery depend on third-party forwarding. To facilitate mail and news delivery, most connected sites are willing to relay files for other sites. The UUCP network depends on direct distance dialing networks and off-peak long distance rates for its continued functioning. For more information on UUCP, see `uucp_setup`(8).

3.5.9 Local Area Transport

Local Area Transport (LAT) is a Digital protocol that supports communications between host computer systems and terminal servers with terminals, PCs, printers, modems and other devices over local area networks

(LANs). LAT software has the features required for a host to function as a service node, so requests for connections can be made by server users. The software also permits host applications to initiate connections to server ports, designated as application ports, to access remote devices. In Digital UNIX, the LAT driver is STREAMS-based and supports up to 4000 incoming connections, with a theoretical limit of 5000 users.

Note

In Digital UNIX, LAT supports both SVR4 and BSD-style `tty` devices. Integral serial `tty` devices and serial `tty` options share the same BSD `tty` namespace as LAT, which means that if users allocate special files for serial lines, those special files will reduce the number of BSD LAT devices that can be configured.

For more information on LAT, see the `lat_intro`(7) reference page and the *System Administration* guide.

3.6 Naming Services

Digital UNIX Version 4.0 supports the following distributed naming services:

- The Berkeley Internet Name Domain (BIND) service
- The Network Information Service (NIS), formerly named Yellow Pages

The library routines in `/usr/lib/libc.a` allow transparent access to BIND, NIS, and local `/etc` files. The name services configuration file, `/etc/svc.conf`, dictates which naming services are queried, and in what order, for a particular database.

The Digital UNIX Version 4.0 software allows you to convert from an NIS-distributed environment to a BIND-distributed environment, or to run both services in the same environment. Because the source files for both BIND and NIS can be `/etc`- style files, a distributed Berkeley Software Distribution (BSD) source area can be shared between the two services by means of symbolic links.

3.6.1 The BIND Service

The Berkeley Internet Name Domain (BIND) service is a host name and address lookup service for the Internet network. The BIND service is based on the client-server model. It allows client systems to obtain host names and addresses from BIND servers. Digital UNIX Version 4.0 only supports the `hosts` database.

Note

Depending on which naming services your LAN is running, the `hosts` file can be located in `/etc`, `/var/yp/src`, or `/etc/namedb/src`.

You can use the BIND service to replace or supplement the host table mapping provided by the local `/etc/hosts` file or NIS.

Digital UNIX Version 4.0 supports BIND 4.9.3.

3.6.2 Network Information Service

The Network Information Service (NIS) is a distributed name service that allows participating hosts to share access to a common set of system and network files. NIS allows the system administrator to manage these shared files on a single system.

NIS is intended for use in a secure environment only, where gateways do not allow outside access from the Internet to the NIS protocol.

3.7 Time Services

Digital UNIX Version 4.0 supports the following time services:

- Network Time Protocol (NTP)
- Time Synchronization Protocol (TSP)

Because it can be traced to clocks of high absolute accuracy, NTP provides a more accurate time service than TSP. By contrast, TSP synchronizes time to the average of the network host times. TSP is an acceptable time service if your system is not on the Internet and does not have access to a highly accurate time server; otherwise, NTP is recommended.

3.7.1 Network Time Protocol

The Network Time Protocol (NTP) provides accurate, dependable, and synchronized time for hosts on both wide area networks (like the Internet) and local area networks. In particular, NTP provides synchronization traceable to clocks of high absolute accuracy, and avoids synchronization to clocks keeping bad time.

Digital UNIX Version 4.0 supports NTP Version 3, based on RFC 1305, which contains the following enhancements to Version 2:

- New algorithms in several clock and peer routines to improve accuracy and stability and reduce errors.
- An authentication mechanism that uses MD5 algorithms for encryption.

- The `ntptrace` utility, which traces a chain of NTP hosts back to their master time source.

Hosts running NTP periodically exchange datagrams querying each other about their current estimate of the time. Using the round-trip time of the packet, a host can estimate the one-way delay to the other. (The assumption is that the delay is roughly equal in both directions.) By measuring the one-way delay and examining the timestamps that are returned with the NTP packet, a host computes the difference between its clock time and that of the host it queried.

A host queries a remote host several times over a period and feeds the results from the multiple samples to a digital-filtering algorithm. The algorithm provides a more accurate estimate of the delay, clock offset, and clock stability than could be obtained with a single sample.

NTP messages also contain information about the accuracy and reliability of the time sources. An NTP host connected directly to a highly accurate time source, such as a radio receiver tuned to a time code signal broadcast by a government agency, is called a stratum 1 server. Every other NTP host adopts a stratum number that is one higher than the host from which it sets its own time. For example, a host synchronized to a stratum 1 server becomes a stratum 2 host. Stratum determination is done automatically, and the stratum of a host can vary as its connectivity changes.

A host running NTP combines various information to decide which of the hosts it queried provides the time it believes to be the most accurate. This information includes the output of the digital-filtering algorithm and the stratum numbers of the hosts it queried. By communicating with several other hosts, an NTP host can usually detect those hosts that are keeping bad time, and is able to stay synchronized even if some of the other hosts become unavailable for long periods.

In practice, NTP is able to synchronize clocks to within a few tens of milliseconds even over wide area networks spanning thousands of miles.

For detailed information on NTP, see RFC 1305: *Internet Time Synchronization: the Network Time Protocol*.

3.7.2 Time Synchronization Protocol

The Time Synchronization Protocol (TSP) is the protocol used by the `/usr/sbin/timed` daemon. In its simplest application, the TSP servers on a broadcast network (for example, an Ethernet) periodically broadcast TSP packets. The hosts on the network elect one of the hosts on the network running TSP as a master. The master then controls the further operation of the protocol until it fails and a new master is elected. The master collects time values from the other hosts and computes the average of all the times reported. It then sets its own clock to this average, and tells the other hosts

to synchronize their clocks with it.

TSP quickly synchronizes all participating hosts. However, because TSP does not trace time back to sources of known accuracy, it is unable to correct for systematic errors. If a clock drifts significantly, or if a mistake is made in setting the time on a participating host, the average time calculated and distributed by the master can be affected significantly.

File System 4

4.1 Overview

Digital UNIX Version 4.0 supports the following file systems which are accessed through the OSF/1 Version 1.0 Virtual File System (VFS):

- UNIX File System (UFS)
- Network File System (NFS)
- CD-ROM File System (CDFS)
- Memory File System (MFS)
- `/proc` File system (PROCFS)
- File-on-File Mounting File System (FFM)
- File Descriptor File System (FDFS)
- POLYCENTER Advanced File System (AdvFS)

Note that all of the file systems are integrated with the Virtual Memory Unified Buffer Cache (UBC).

In addition, Digital UNIX Version 4.0 supports the Logical Storage Manager (LSM) and the Prestoserve file system accelerator.

Note that the Logical Volume Manager is being retired in this release.

The following sections briefly discuss VFS, the file systems supported in Digital UNIX Version 4.0, the Logical Storage Manager, and the Prestoserve file system accelerator.

4.2 Virtual File System

The Virtual File System (VFS), which is based on the Berkeley 4.3 Reno Virtual File System, provides a uniform interface abstracted from the file system layer which allows common access to files, regardless of the file system on which the files reside. A structure known as a `vnode` (analogous to an inode) contains information about each file in a mounted file system and is more or less a wrapper around file system-specific nodes. If, for example, a read or write is requested on a file, the `vnode` points the system call to the system call appropriate for that file system (a `read` request is

pointed to a ufs_read if the request is made on a file in a UFS file system or to an nfs_read if the request is made on a file in an NFS-mounted file system). As a result, file access across different file systems is transparent to the user.

Digital's VFS implementation also supports Extended File Attributes (XFAs). Although originally intended to provide support for system security (Access Control Lists) and the Pathworks PC server (so that a Pathworks PC server could assign PC-specific attributes to a file, such as icon color, the startup size of the application, its backup date, and so forth), the XFA implementation was expanded to provide support for any application that wants to assign an XFA to a file. Currently, both UFS and AdvFS support XFAs, as well as the pax backup utility which has a tar and cpio front-end. XFAs are also supported for remote UFS file systems, to a server which supports a special protocol which currently only Digital supports. For more information on XFAs, see setproplist(2). For more information on pax, see pax(1).

Information for File System Developers

In Digital UNIX Version 4.0, the VOP_READDIR kernel vnode operation interface has been changed to accommodate a new structure, kdirent, in addition to the existing dirent structure.

The new kdirent structure was developed to make file systems other than UFS work properly over NFS.

Note, however, that if you implement a file system under Digital UNIX, you do not need to make any changes to your VOP_READDIR interface routine for Digital UNIX Version 4.0, and applications see the same interface as before the addition of the new kdirent structure.

Unlike the dirent structure, the kdirent structure has a kd_off field that subordinate file systems can set to point to the on-disk offset of the next directory entry. Arrays of struct kdirent must be padded to 8-byte boundaries, using the KDIRSIZE macro, so that the off_t is properly aligned; arrays of struct dirent are only padded to 4 bytes.

Each mounted file system has the option of setting the M_NEWRDDIR flag in the mount structure m_flag field. If the M_NEWRDDIR flag is set, then the routine calling VOP_READDER expects the readdir on that vnode to return an array of struct kdirent; if the M_NEWRDDIR flag is clear (the default), then the the readdir on that vnode returns an array of struct dirent.

In terms of NFS, if the M_NEWRDDIR flag is not set, then the NFS server uses the dirent structures and then calculates the necessary offset to pass back to the server. Thus, to ensure proper operation over NFS, any file system that does not have the M_NEWRDDIR flag set must be prepared to

have VOP_READDIR called with offsets based on a packed array of struct dirent, which may be in conflict with the offsets on the on-disk directory structure. However, if the M_NEWRDDIR flag is set, then the NFS server uses the kd_off fields of the kdirent structures to generate the necessary offsets to pass back to the server.

A new vnode operation VOP_PATHCONF was added to the kernel in order to return filesystem-specific information for the fpathconf() and pathconf() system calls. This vnode operation takes as arguments the pointer to struct vnode, the pathconf name int, return value pointer to long and error int. It also sets the return value and ERRNO. Note that each filesystem must implement the vnode operation by providing a function in the vnodeops structure after the vn_delproplist component (at the end of the structure). This function takes as arguments the pointer to vnode, the pathconf name, and the return value pointer to long. The function sets the return value and returns zero for succes or an error number.

4.3 UNIX File System

The UNIX File System (UFS) is compatible with the Berkeley 4.3 Tahoe release. UFS allows a pathname component to be 255 bytes, with the fully qualified pathname length restriction of 1023 bytes. The Digital UNIX Version 4.0 implementation of UFS supports file sizes which exceed 2 GBs.

Digital added support for file block clustering which provides sequential read and write access that is equivalent to the raw device speed of the disk and up to a 300% performance increase over previous releases of the operating system; file-on-file mounting (FFM) for STREAMS; and integrated UFS with the Unified Buffer Cache. UFS also supports Extended File Attributes (XFAs). For more information on XFAs, see Section 4.2.

4.4 Network File System

The Network File System (NFS) is a facility for sharing files in a heterogeneous environment of processors, operating systems, and networks, by mounting a remote file system or directory on a local system and then reading or writing the files as though they were local.

Digital UNIX Version 4.0 supports NFS Version 3, in addition to NFS Version 2. NFS Version 2 code is based on ONC Version 4.2, which Digital licensed from Sun Microsystems. The NFS Version 3 code supersedes ONC Version 4.2, although at the time that NFS Version 3 was ported to Digital UNIX, Sun Microsystems had not yet released a newer, public version of ONC with NFS Version 3 support.

4.4.1 NFS Version 3 Functionality

NFS Version 3 supports all the features of NFS Version 2 as well as the following:

- 64-bit remote access

  Allows users to access files larger than 2 GBs over NFS

- Improved performance
  - Support for reliable asynchronous writes which improves write performance over NFS Version 2 by a factor of seven, thereby reducing client response latency and server I/O loading
  - Support for a `READDIRPLUS` procedure that returns file handles and attributes with directory names to eliminate `LOOKUP` calls when scanning a directory
  - Support for servers to return metadata on all operations to reduce the number of subsequent `GETATTR` procedure calls
  - Support for weak cache consistency data to allow a client to manage its caches more effectively
- Improved security
  - Provides an `ACCESS` procedure that fixes the problems in NFS Version 2 with superuser permission mapping, and allows access checks at file-open time, so that the server can better support Access Control Lists (ACLs)
  - File names and pathnames specified as strings of variable length, with the maximum length negotiated between the client and server using the `PATHCONF` procedure
- Guaranteed exclusive creation of files

Since Digital UNIX supports both NFS Version 3 and Version 2, the NFS client and server bind at mount time using the highest NFS version number they both support. For example, a Digital UNIX Version 4.0 client will use NFS Version 3 when it is served by a Digital UNIX Version 4.0 NFS server; however, when it is served by an NFS server running an earlier version of Digital UNIX, the Digital UNIX Version 4.0 NFS client will use NFS Version 2.

For more detailed information on NFS Version 3, see the paper *NFS Version 3: Design and Implementation* (USENIX, 1994).

4.4.2 Digital Enhancements to NFS

In addition to the NFS Version 3.0 functionality, Digital UNIX supports the following Digital enhancements to NFS:

- NFS over TCP

  NFS has been traditionally run over the UDP protocol. Digital Unix V4.0 now supports NFS over the TCP protocol. See `mount`(8) for additional details.

- Write-gathering

  On an NFS server, multiple write requests to the same file are combined to reduce the number of actual writes as much as possible. The data portions of successive writes are cached and a single metadata update is done that applies to all the writes. Replies are not sent to the client until all data and associated metadata are written to disk to ensure that write-gathering does not violate the NFS crash recovery design.

  As a result, write-gathering increases write throughput by up to 100 % and the CPU overhead associated with writes is substantially reduced, thereby further increasing server capacity.

- NFS-locking

  Using the `fcntl` system call to control access to file regions, NFS-locking allows you to place locks on file records over NFS, thereby protecting, among other things, segments of a shared, NFS-served database. The status daemon, `rpc.statd`, monitors the NFS-servers and maintains the NFS lock if the server goes down. When the NFS server comes back up, a reclaiming process allows the lock to be reattached.

- Automounting

  The `automount` daemon automatically and transparently mounts and unmounts NFS file systems on an as-needed basis. It provides an alternative to using the `/etc/fstab` file for NFS mounting file systems on client machines.

  The `automount` daemon can be started from the `/etc/rc.config` file or from the command line. Once started, it sleeps until a user attempts to access a directory that is associated with an automount map or any directory or file in the directory structure. The daemon awakes and consults the appropriate map and mounts the NFS file system. After a specified period of inactivity on a file system, 5 minutes by default, the automount daemon unmounts that file system.

  The maps indicate where to find the file system to be mounted and the mount options to use. An individual automount map is either local or served by NIS. A system, however, can use both local and NIS automount maps.

Automounting NFS-mounted file systems provides the following advantages over static mounts:

- If NIS maps are used and file systems are moved to other servers, users do not need to do anything to access the moved files. Every time the file systems need to be mounted, the daemon will mount them from the correct locations.
- In the case of read-only files, if more than one NFS-server is serving a given file system, `automount` will connect you to the fastest server that responds. If at least one of the servers is available, the mount will not hang.
- By unmounting NFS-mounted file systems that have not been accessed for more than a certain interval (5 minutes by default), the `automount` daemon conserves system resources, particularly memory.

- PC-NFS

  PC-NFS, a product for PC clients available from Sun Microsystems, allows personal computers running DOS to access NFS servers as well as providing a variety of other functionality.

  Digital supports the PC-NFS server daemon, `pcnfsd`, which allows PC clients with PC-NFS configured to do the following:

  - Mount NFS file systems

    The PC-NFS `pcnfsd` daemon, in compliance with Versions 1.0 and 2.0 of the `pcnfsd` protocol, assigns UIDs and GIDs to PC clients so that they can talk to NFS.

    The `pcnfsd` daemon performs UNIX login-like password and username verification on the server for the PC client. If the authentication succeeds, the `pcnfsd` daemon then grants the PC client the same permissions accorded to that username. The PC client can mount NFS file systems by talking to the `mountd` daemon as long as the NFS file systems are exported to the PC client in the `/etc/exports` file on the server. Since there is no mechanism in DOS to perform file permission checking, the PC client calls the authentication server to perform checking of the user's credentials against the file's attributes. This happens when the PC client makes NFS requests to the server for file-access that requires permission checking, such as opening of a file.

  - Access network printers

    The `pcnfsd` daemon authenticates the PC client and then spools and prints the file on behalf of the client.

4.5 CD-ROM File System

Digital UNIX Version 4.0 supports the ISO-9660 CDFS standard for data interchange between multiple vendors; High Sierra Group standard for backward compatibility with earlier CD-ROM formats; and an implementation of the Rock Ridge Interchange Protocol (RRIP), Version 1.0, Revision 1.09. The RRIP extends ISO-9660 using the system use areas defined by ISO-9660 to provide mixed-case and long filenames; symbolic links; device nodes; deep directory structures (deeper than ISO-9660 allows); UIDs, GIDs, and permissions on files; and POSIX time stamps.

This code was taken from the public domain and enhanced by Digital.

In addition, Digital UNIX Version 4.0 also supports X/Open Preliminary Specification (1991) CD-ROM Support Component (XCDR). XCDR allows users to examine selected ISO-9660 attributes through defined utilities and shared libraries, and allows system administrators to substitute different file protections, owners, and file names for the default CD-ROM files.

4.6 Memory File System

Digital UNIX Version 4.0 supports a Memory File System (MFS) which is essentially a UNIX File System that resides in memory. No permanent file structures or data are written to disk, so the contents of an MFS file system are lost on reboots, unmounts, or power failures. Since it does not write data to disk, the MFS is a very fast file system and is quite useful for storing temporary files or read-only files that are loaded into it after it is created.

For example, if you are performing a software build which would have to be restarted if it failed, the MFS is a very appropriate choice to use for storing the temporary files that are created during the build, since by virtue of its speed it would reduce the build time. For more information, see the `newfs`(8) reference page.

4.7 /proc File System

The `/proc` file system enables running processes to be accessed and manipulated as files by the system calls `open`, `close`, `read`, `write`, `lseek`, and `ioctl`. While the `/proc` file system is most useful for debuggers, it enables any process with the correct permissions to control another running process. Thus, a parent/child relationship does not have to exist between a debugger and the process being debugged. The `dbx` debugger that ships in Digital UNIX Version 4.0 supports attaching to running processes through `/proc`. For more information, see the `proc`(4) and `dbx`(1) reference pages.

4.8 File-on-File Mounting File System

The File-on-File Mounting (FFM) file system allows regular, character, or block-special files to be mounted over regular files, and, for the most part, is only used by the SVR4-compatible system calls `fattach` and `fdetach` of a STREAMS-based pipe (or FIFO). With FFM, a FIFO, which normally has no file system object associated with it, is given a name in the file system space. As a result, a process that is unrelated to the process that created the FIFO can then access the FIFO.

In addition to programs using FFM through the `fattach` system call, users can mount one regular file on top of another using the `mount` command. Mounting a file on top of another file does not destroy the contents of the covered file; it simply associates the name of the covered file with the mounted file, making the contents of the covered file temporarily unavailable. The covered file can be accessed after the file mounted on top of it is unmounted, either by a reboot or by a call to `fdetach`, or by entering the `umount` command. Note that the contents of the covered file are still available to any process which had the file open at the time of the call to `fattach` or when a user issued a `mount` command that covered the file.

4.9 File Descriptor File System

The File Descriptor File System (FDFS) allows applications to reference a process's open file descriptors (0, 1, 2, 3, and so forth) as if they were files in the UNIX File System (for example, `/dev/fd/0`, `/dev/fd/1`, `/dev/fd/2`) by aliasing a process's open file descriptors to file objects. When the FDFS is mounted, opening or creating a file descriptor file has the same effect as calling the `dup`(2) system call.

The FDFS allows applications that were not written with support for UNIX I/O to avail themselves of pipes, named pipes, and I/O redirection.

The FDFS is not mounted by default and must either be mounted by hand or by an entry placed in the `/etc/fstab` file.

For more information on the FDFS, see the `fd`(4) reference page.

4.10 POLYCENTER Advanced File System

The POLYCENTER Advanced File System (AdvFS), which consists of a file system that ships with the base system and a set of file system utilities that are available as a separate, layered product, is a log-based (journaled) file system that is especially valuable on systems with large amounts of storage. Because it maintains a log of active file-system transactions, AdvFS avoids lengthy file system checks on reboot and can therefore recover from a system failure in seconds. AdvFS ensures that log records are written to disk before data records, ensuring that file domains (file systems) are recovered to a

consistent state. AdvFS uses extent-based allocation for optimal performance.

To users and applications, AdvFS looks like any other UNIX file system. It is compliant with POSIX and SPEC 1170 file-system specifications. AdvFS file domains and other Digital UNIX file systems, like UFS, can exist on the same system and are integrated with the Virtual File System (VFS) and the Unified Buffer Cache (UBC). AdvFS file domains can also be remote-mounted with NFS and support extended file attributes (XFAs). For more information on XFAs, see Section 4.2.

In addition to providing rapid restart and increased file-system integrity, AdvFS supports files and file systems much larger than 2 GBs and, by separating the file system directory layer from the logical storage layer, provides increased file-system flexibility and manageability.

In addition to the Advanced File System that ships as part of the base operating system, the POLYCENTER Advanced File System Utilities are available as a layered product. The AdvFS Utilities enable a system administrator to create multivolume file domains, add and remove volumes online, clone filesets for online backup, unfragment and balance file domains online, stripe individual files, and establish trashcans so that users can restore their deleted files. The AdvFS Utilities also provide a Graphical User Interface for configuring and managing AdvFS file domains. The AdvFS Utilities require a separate license Product Authorization Key (PAK). Contact your Digital representative for additional information on the AdvFS Utilities product. For more information on AdvFS, see the *System Administration* guide and the *POLYCENTER Advanced File System Utilities Technical Summary*.

4.11 Logical Storage Manager

Digital UNIX Version 4.0 supports the Logical Storage Manager (LSM), a more robust logical storage manager than Logical Volume Manager (LVM), which it has replaced. LSM supports all of the following:

- Disk spanning

  Disk spanning allows you to concatenate entire disks or parts (regions) of multiple disks together to use as one, logical volume. So, for example, you could "combine" two RZ26s and have them contain the `/usr` file system.

- Mirroring

  Mirroring allows you to write simultaneously to two or more disk drives to protect against data loss in the event of disk failure.

- Striping

  Striping improves performance by breaking data into segments that are

written to several different physical disks in a "stripe set."

- Comprehensive disk management capabilities

  LSM supports disk management utilities that, among other things, change the disk configuration without disrupting users while the system is up and running.

Mirroring, striping and the graphical interface require a separate license PAK. The LSM code came from VERITAS (the VERITAS Volume Manager) and was enhanced by Digital.

For each logical volume defined in the system, the LSM volume device driver maps logical volume I/O to physical disk I/O. In addition, LSM uses a user-level volume configuration daemon (`vold`) that controls changes to the configuration of logical volumes. Users can administer LSM either through a series of command-line utilities or by availing themselves of an intuitive Motif-based graphical interface.

To ensure a smooth migration from LVM to LSM, Digital has developed a migration utility that maps existing LVM volumes into nonstriped, nonmirrored LSM volumes that preserves all of the LVM data. After the migration is complete, administrators can mirror the volumes if they so desire.

Similarly, to help users transform their existing UFS or AdvFS partitions into LSM logical volumes, Digital has developed a utility that will transform each partition in use by UFS or AdvFS into a nonstriped, nonmirrored LSM volume. After the transformation is complete, administrators can mirror the volumes if they so desire.

Note that LSM volumes can be used in conjunction with AdvFS, as part of an AdvFS domain; with RAID disks; and with the Available Server Environment (ASE), since LSM supports logical volume failover. For more information on LSM, see the *Logical Storage Manager*.

4.12 Overlap Partition Checking

The enhancements related to Overlap Partition Checking are described next.

4.12.1 Partition Overlap Checks Added to Utilities

Partion overlap checks were added to a number of commands in Digital UNIX Version 4.0. Some of the commands which use these checks are: `newfs`, `fsck`, `mount`, `mkfdnm`, `swapon`, `voldisksetup`, and `voldisk`. The enhanced checks require a disk label to be installed on the disk. Refer to the `disklabel`(8) reference page for further information.

The checks ensure that if a partition or an overlapping partition is already in use (for example, mounted or used as a swap device), the partition will not

be overwritten. Additionally, the checks ensure that partitions will not be overwritten if the specific partition or an overlapping partition is marked in use in the `fstype` field on the disk label.

If a partition or an overlapping partition has an in-use `fstype` field in the disklabel, some commands inquire interactively if a partition can be overwritten.

4.12.2 Library Functions for Partition Overlap Checking

Two new functions, `check_usage`(3) and `set_usage`(3) are available for use by applications. These functions check whether a disk partition is marked for use and set the `fstype` of the partition in the disk label. See the appropriate reference pages for these functions for more information.

4.13 Prestoserve File System Accelerator

The Prestoserve file system accelerator is a hardware option that speeds up synchronous disk writes, including NFS server access, by reducing the amount of disk I/O. Frequently-written data blocks are cached in nonvolatile memory and then written to disk asynchronously.

The software required to drive the board ships as an optional subset in Digital UNIX Version 4.0 and once it is installed can be accessed with a PAK that comes with the board.

Prestoserve uses a write cache for synchronous disk I/O. Prestoserve works in a way that is similar to the way the system buffer cache speeds up asynchronous disk I/O requests. Prestoserve is interposed between the operating system and the device drivers for the disks on a server. Mounted file systems and unmounted block devices selected by the administrator are accelerated.

When a synchronous write request is issued to a disk with accelerated file systems or block devices, it is intercepted by the Prestoserve pseudodevice driver, which stores the data in nonvolatile memory instead of on the disk. Thus, synchronous writes occur at memory speeds, not at disk speeds.

As the nonvolatile memory in the Prestoserve cache fills up, it asynchronously flushes the cached data to the disk in portions that are large enough to allow the disk drivers to optimize the order of the writes. A modified form of Least Recently Used (LRU) replacement is used to determine the order. Reads that hit (match blocks) in the Prestoserve cache also benefit.

Nonvolatile memory is required because data must not be lost if the power fails or if the system crashes. As a result, the hardware board contains a battery that protects data in case the system crashes. From the point of view of the operating system, Prestoserve appears to be a very fast disk.

Note that there is a substantial performance gain when Prestoserve is used on an NFSV2 server.

The `dxpresto` command allows you to monitor Prestoserve activity and to enable or disable Prestoserve on machines that allow that operation. For more information on Prestoserve see the *Guide to Prestoserve* and the `dxpresto`(8X) reference page.

Virtual Memory 5

5.1 Overview

The virtual memory (VM) subsystem was completely rewritten by Digital in V2.0 in order to improve upon the Mach design adopted by the OSF. Specifically, Digital added the following functionality to improve performance and maintainability (V4.0 added several enhancements to these areas):

- Improved upon the lazy allocation policy
- Improved upon the eager reservation policy
- Unified Buffer Cache
- Round-robin swapping algorithm
- Page in and page out clustering
- Memory-mapped device interface
- Mach `mmap` MAP_PRIVATE semantics to make them compatible with Sun Microsystems and System V Release 4.0
- Ensured that processes cannot read or write other processes' shared memory segments
- Support for shared text segments
- Support for page coloring
- New kernel memory allocator
- Improved memory reclamation policy
- Restructured swap allocation mechanism

The following sections discuss these improvements to VM.

5.2 Lazy Allocation Policy

In lazy allocation, swap space is reserved dynamically as the system needs to reclaim physical memory instead of having to allocate it in advance for every page of anonymous memory (that is, memory devoted to the stack and heap of a process, and to data that is not file-backed).

However, when the list of active pages falls below the preconfigured limit and the OSF memory manager attempts to reclaim pages for the free list by paging out virtual pages, if the available swap space has already been exhausted, the OSF memory manager does not back off of the page-out and instead simply discards the page. Thus, when the process whose page has been discarded takes a page fault and attempts to reactivate the missing page, unpredictable behavior results, including system hangs, panics, and at times data corruption.

To correct this problem, Digital reworked the page-out algorithm to ensure that pages are not lost if the memory manager is unable to allocate swap space for a virtual page.

As swap space decreases, the Digital UNIX Version 4.0 memory manager logs warning messages at the console until finally, if the memory manager is unable to allocate swap space for a page-out, it selects the oldest idle process and kills it, thereby freeing up swap space and returning virtual pages to the free list.

5.3 Eager Reservation Policy

Unlike lazy allocation, the eager reservation policy reserves a page of swap space for every page of anonymous memory that is allocated. Although this policy is expensive in terms of reserved disk space, it eliminates the chance that the memory manager will have to kill a process to reclaim virtual pages and free up swap space.

The eager reservation policy is set by default, although either the lazy or eager policy can be configured. For more information, see the *System Tuning and Performance Management* guide

5.4 Unified Buffer Cache

To increase file system performance, Digital implemented a Unified Buffer Cache (UBC) fully integrated with the file system that caches file system data and can grow or shrink upon demand. Unlike the conventional Buffer Cache which is configured and allocated at boot-time and which relies on `bcopy` routines to move data in and out of memory, the UBC references the same physical pages as virtual memory and can use map operations rather than `bcopy` routines to access data, thereby increasing system performance. In addition, since the UBC contains only file system data, the Buffer Cache only needs to cache metadata, requiring only 3% of physical memory rather than the 25% required by previous versions of the operating system.

By default, the UBC can grow to consume all of physical memory so that the system can determine dynamically the percent of memory that should be allocated to the UBC. However, the maximum percent of memory that the

UBC can grow to is configurable and can be set in the system configuration file by defining the `ubc-maxpercent` variable.

5.5 Round-Robin Swapping

In an effort to improve performance, Digital changed the OSF paging algorithm to support simultaneous paging to multiple swap partitions. By contrast, OSF paging is to one swap partition at a time, waiting until the first swap partition is filled before moving to the next. As a result, since disk transfer rates are several thousand times slower than the speed of memory and the Alpha CPU, system administrators can greatly reduce this disparity in speed by spreading swap partitions among different disks and different controllers. In fact, by supporting simultaneous paging to multiple swap partitions, Digital UNIX Version 4.0 allows multiple tasks to take simultaneous page faults, thereby further increasing performance.

5.6 Page In and Page Out Clustering

Whereas OSF/1 supports a paging algorithm that moves pages in and out one at a time, to improve performance, Digital UNIX adds support for page in and page out clustering. Although in most cases, it is more efficient to do multiple multipage DMA operations rather than multiple single page DMA operations, this is a configurable option.

5.7 Memory-Mapped Device Interface

Rather than mapping the I/O space into memory, the memory-mapped device interface points to a data structure that defines the I/O space. As a result, large quantities of kernel virtual address space are saved as well as physical memory in general.

5.8 Mach mmap MAP_PRIVATE Semantics and System V Release 4.0

Since the OSF departed from the Sun Microsystems and System V Release 4.0 `mmap` semantics, Digital rewrote `mmap` so that Sun and System V applications could compile and run on Digital UNIX Version 4.0.

5.9 Secure Shared Memory Segments

Using the same kind of permission bits that the file system employs, shared memory segments can be read and write protected to prevent unwanted access.

5.10 Shared Text Segments

Shared text segments allow multiple processes to share the same page tables that map shared text. All processes that share the same text segment benefit from one process taking a page fault, because less memory is needed and the performance of `fork` is improved.

5.11 Page Coloring

The Alpha EV4 CPU contains a direct mapped physical OFF chip secondary cache, which is organized so that if the secondary cache size is N pages, then every Nth page of the physical pages of memory hashes into the same page. Digital UNIX VM manages the physical pages of memory in such a way that, if an entire resident working set of a process can fit into the secondary cache, VM places it there. As a result, because VM strives to ensure that a process's entire working set is always in the secondary cache, the number of physical memory accesses is greatly reduced as a process executes.

5.12 Caches

The Alpha EV5 CPU optionally contains 3 levels of caches: internal, secondary, and tertiary caches. Digital UNIX manages physical memory in such a way that the most active subset of a process working set will remain in the fastest cache.

5.13 Kernel Memory Allocator

A new kernel memory allocator (kernel `malloc`) was added to Digital UNIX to use kernel-wired memory more efficiently. All calls to the `mbuf` allocator are now mapped to the new kernel memory allocator. In addition, several components of the I/O subsystems can use the kernel memory allocator directly, rather than having to manage memory on their own. As a result, we save the amount of memory these allocators were reserving. In addition, the new allocator handles allocation under interrupt context better than the `kalloc` allocator and has a garbage collection thread to free memory.

5.14 External Pager

Although the OSF provides guidelines for writing an external pager, these guidelines are (at best) provisional. Digital believes that the complexity and efficiency of the Digital UNIX Version 4.0 memory management system makes it impractical at this time to provide a sensible interface for an external pager.

5.15 Improved Memory Reclamation Policy

The memory reclamation policy was enhanced in Digital UNIX Version 4.0 to improve system performance. In previous releases of the operating system, once the system began running out of physical memory, global paging would begin to reclaim memory, attempting to select pages fairly between the Unified Buffer Cache (UBC) and VM. However, as the system runs and files are opened and closed, a large percentage of memory in the UBC is always referencing old, closed files—owing to its file caching algorithms. As a result, attempting to select some pages from VM and some pages from the UBC is actually unnecessary initially, since theoretically only 10% of the pages in the UBC are dirty, whereas almost all the pages in VM are dirty (in actual practice, however, the amount of dirty pages in the UBC is much smaller than 10% since most of the pages in the UBC are not in use). Because the UBC, unlike VM, contains so few dirty pages, it is much more efficient to reclaim pages from the UBC first, down to some configurable level, than it is to begin global paging immediately.

In Digital UNIX Version 4.0, the page reclamation policy was further enhanced to take advantage of the many unreferenced pages that are in both VM and UBC. When a system begins to exhaust its physical memory, memory is first reclaimed from the UBC down to a threshold defined by the parameter `ubc-borrowpercent`. This parameter is set by default to be 10% of the memory in the UBC and is configurable. If, after all unused memory is reclaimed from the UBC and the system still requires more physical memory, global paging is then invoked.

In effect, this policy can double the load that can be placed on a system before demands for memory begin to noticeably degrade performance.

5.16 Rewrote Swap Allocation Mechanism

In Digital UNIX Version 4.0, the swap allocation mechanism was restructured to further reduce the fragmentation of swap space which could occur over time. Swap space is now allocated and deallocated in contiguous units, so that seek time is greatly reduced, thereby greatly improving performance.

For information on how to tune and configure the virtual memory subsystem, see the *System Tuning and Performance Management* guide and the *System Administration* guide.

I/O Subsystem 6

6.1 Overview

The Digital UNIX Version 4.0 I/O subsystem supports a variety of buses and devices depending upon the particular machine and its configuration. Table 6-1 lists the most common configurations.

Table 6-1: Supported Processors and Buses

Processors	Buses
AlphaStation 200, 250, and 255 series processors	PCI Bus ISA Bus SCSI Bus
AlphaStation/Server 400 series processors	PCI Bus ISA Bus SCSI Bus
AlphaStation 600 series processors	PCI Bus EISA Bus SCSI Bus
AlphaServer 1000 series processors	PCI Bus EISA Bus SCSI Bus
Alpha VME 2100 series processors	VME Bus EISA Bus
AlphaServer 2000 series processors	PCI Bus EISA Bus SCSI Bus
AlphaServer 2100 series processors	PCI Bus EISA Bus SCSI Bus

Table 6-1: (continued)

Processors	Buses
AlphaServer 8000 series processors	PCI Bus Futurebus+ XMI Bus EISA Bus SCSI Bus
DEC 2000 series processors	EISA Bus SCSI Bus
DEC 2100 series processors	PCI Bus EISA Bus SCSI Bus
DEC 3000 series processors	TURBOchannel Bus SCSI Bus
DEC 4000 series processors	Futurebus+ SCSI Bus
DEC 7000/10000 series processors	Futurebus+ XMI Bus SCSI Bus CI to HSC to RA/TA devices KDM to RA/TA devices
Single Board Computer (SBC) EB66+	PCI Bus EISA Bus
Single Board Computer (SBC) EB64+	PCI Bus EISA Bus
Single Board Computer (SBC) EB164	PCI Bus EISA Bus
Single Board Computer (SBC) AlphaPC64	PCI Bus EISA Bus

All of the device driver code is written by Digital and is written to published, industry standards.

In addition, Digital's UNIX Publications Group publishes a device driver tutorial and a series of bus books that are designed to assist third-party vendors in writing device drivers that are compatible with Digital UNIX Version 4.0.

For more information on how to write device drivers for Digital UNIX Version 4.0, see the following books:

Writing Device Drivers: Tutorial
Writing Device Drivers: Reference
Writing EISA and ISA Bus Device Drivers
Writing PCI Bus Device Drivers

Writing Device Drivers for the SCSI/CAM Architecture Interfaces
Writing TURBOchannel Device Drivers
Writing VMEbus Device Drivers

For information on the various peripheral devices that Digital UNIX Version 4.0 supports, refer to the *Alpha Systems Handbook*, the *Software Product Description* (SPD), and the *Systems and Options Catalog*.

The following sections briefly discuss the buses supported in Digital UNIX Version 4.0 as well as I/O enhancements such as RAID and tagged queueing, and the XMI CI and KDM controllers.

6.2 Supported Buses

Depending on the particular processor, Digital UNIX Version 4.0 supports the following buses:

- PCI
- ISA
- EISA
- Futurebus+
- SCSI
- TURBOchannel
- XMI
- VME

6.2.1 PCI Bus

The Peripheral Component Interconnect Local Bus (PCI) is an open, high-performance 32-bit or 64-bit synchronous bus with multiplexed address and data lines, and numerous compatible hardware implementations. Table 6-1 lists the Digital processors that support the PCI Bus.

Most Digital processors that support the PCI bus support a PCI frequency of 33 MHz and a transfer rate of 132 MB per second. However, the AlphaStation 600 has 2 different types of PCI slots:

- 32-bit slots with a PCI frequency of 33Mhz and a transfer rate of 132 MB per second.
- 64-bit slots with a PCI frequency of 33Mhz and a transfer rate of 264 MB per second. [64-bit slots can also hold 32-bit option cards, but will only run at 132 MB per second.]

Digital does not yet supply any 64-bit option cards, but as PCI is an industry open bus, other vendors may offer 64 bit options.

Table 6-2 provides a list of some of the PCI bus adapters and interconnects that are available both from Digital and third-party vendors. For more specific information on supported PCI bus adapters and interconnects in Digital UNIX Version 4.0, see the SPD. Note that because of the open nature of the PCI bus, Digital does not control the development and availability of adapters.

Table 6-2: PCI Bus Adapters and Interconnects

Interconnect	Adapters
FDDI	DEFPA-AA (single) DEFPA-DA (dual) DEFPA-UA (UTP)
NI	DE435-AA, DE434, DE436 (Quad), DE450, DE500 (Fast Ethernet)
SCSI	KZPSM, KZPAA, KZPBA-BB (FWD), KZPDA KZPSA (FWD),
RAID	SWXCR-Px
Graphics	PBXGA (TGA), PB2GA-FA (ATI MACH 64 CX), PB2GA-BA (Compaq Qvision 1280/p), PB2GA-JA (S3TR1064)
NVRAM	Digital Option

For more information on the PCI bus, see the *PCI Local Bus Specification Revision 2.0* and the *PCI to PCI Bridge Architecture Specification.*

6.2.1.1 Redundant Array of Independent Disks (RAID)

See Section 6.2.3.1 for information on PCI support for RAID, which is supported by PCI and EISA buses.

6.2.2 ISA Bus

The Industry Standard Architecture (ISA) bus is an open, 8-bit (PC and XT) or 16-bit (AT) asymmetrical I/O channel with numerous compatible hardware implementations and, as Table 6-1 indicates, is supported on the AlphaStation 200 and 400 series processors and on the AlphaServer 400 series processors. Digital UNIX Version 4.0 supports the high-speed ISA bus implementation that is completely separate from the system bus and allows data transfer rates at a bandwidth of up to 33 MB per second, supports a 16 MB address space and 8 DMA channels.

Table 6-3 illustrates the frequency and transfer rate for various Digital processors that support the ISA bus.

Table 6-3: ISA Bus Frequency and Transfer Rates

Processor	ISA Frequency	ISA Transfer Rate
AlphaStation 200 series	8.33 MHz	33 MB/s
AlphaStation/Server 400 series	@8.33 MHz	33 MB/s

Table 6-4 provides a list of the ISA bus adapters and interconnects that are available both from Digital and third-party vendors. For more specific information on supported ISA bus adapters and interconnects in Digital UNIX Version 4.0, see the SPD.

Table 6-4: ISA Bus Adapters and Interconnects

Interconnect	Adapters
NI/Ethernet	DE205, DE204, DE203
ISA, Dual serial line	PC4XD-AB
ISA, Serial line and Parallel Line	PC4XD-AA
ISA, 2400 baud modem	PCXBF-AA
ISA, 9600-baud modem	PCXCF-AA
ISA 14400	PCXDF-AA
Token Ring	Digital DW110
Graphics	PB2GA-FB (ATI MACH 64)

For more specific information on supported ISA adapters and interconnects in Digital UNIX Version 4.0, see the Software Product Description.

6.2.3 EISA Bus

The Extended Industry Standard Architecture (EISA) bus is an open, 32-bit, asymmetrical I/O channel with numerous compatible hardware implementations and, as Table 6-1 indicates, is supported on variety of Digital processors. Digital UNIX Version 4.0 supports the high-speed EISA bus implementation that is completely separate from the system bus and

allows data transfer rates at a bandwidth of up to 33 MB per second, supports a 4 GB address space, 8 DMA channels, and is backward compatible with the Industry Standard Architecture (ISA) bus.

The Digital processors that support the EISA bus support an EISA frequency of 8.33 MHz and a transfer rate of 33 MB per second.

Table 6-5 provides a list of the EISA bus adapters and interconnects that are available both from Digital and third-party vendors. For more specific information on supported EISA bus adapters and interconnects in Digital UNIX Version 4.0, see the SPD.

Table 6-5: EISA Bus Adapters and Interconnects

Interconnect	Adapters
FDDI	Digital DEFEA-AA
Ethernet	Digital Equipment Corporation DE422 and DE425
SCSI-2 SCSI-2 with FDI controller	Adaptec AHA-1740A (High Performance) Adaptec AHA-1742A (High Performance)
ISA, Dual serial line ISA, Serial line and Parallel Line ISA, 2400 baud modem ISA, 9600 baud modem ISA 14400	PC4XD-AB PC4XD-AA PCXBF-AA PCXCF-AA PCXDF-AA
Token Ring	DW300
Prestoserve-NVRAM	Digital Equipment Corporation PB2SX-AA
RAID	SWXCR-Ex
Graphics	PB2GA-AA (Compaq Qvision 1024/E), PB2GA-FA (ATI Mach 64 ISA)

Note that in general, all ISA options can be used on an EISA bus system.

For more specific information on supported EISA adapters and interconnects in Digital UNIX Version 4.0, see the SPD. See Section 6.2.3.1 for information on the redundant array of independent disks (RAID) feature, which is supported by both PCI and EISA buses.

6.2.3.1 Redundant Array of Independent Disks

The Redundant Array of Independent Disks (RAID) enhances I/O performance and reliability by supporting such features as disk shadowing and the breaking up of data between several disks (called striping). Digital UNIX Version 4.0 supports an EISA RAID controller, SWXCR-E, and a PCI RAID controller, SWXCR-P. On these controllers, devices represent themselves to the operating system as standard `re` disks. For more information on `re` devices, see the `re`(7) reference page.

All RAID controllers support various levels of shadowing and striping, ranging from 0 to 7. The EISA and PCI RAID controllers support RAID levels 0, 1, 5, and 6, with level 6 being vendor-specific.

Support consists of the following:

- Level 0

  Striping without redundancy

- Level 1

  Shadowing

- Level 5

  Striping and parity

- Level 6

  Striping without redundancy and shadowing (level 0 + level 1)

- JBOD – Just a Bunch of Disks

  The SWXCR series controller works like a regular disk controller; no RAID functionality is enabled.

6.2.4 Futurebus+

Futurebus+ is an open bus, designed by the IEEE 896 committee, whose architecture and interfaces are publicly documented, and that is independent of any underlying architecture. It has broad-base, cross-industry support; very high throughput (the maximum rate for 64-bit bandwidth is 160 MB per second; for the 128-bit bandwidth, 180 MB per second); and, as Table 6-1 indicates, it is supported on the DEC 4000, 7000, and 10000 series processors. In addition, Futurebus+ supports a 64-bit address space and a set of control and status registers (CSRs) that provides all the necessary ability to enable or disable features; thus supporting multivendor interoperablity.

Table 6-6 provides a list of Futurebus+ adapters and interconnects that are available from both Digital and third-party vendors. For more specific information on supported Futurebus+ adapters and interconnects in Digital UNIX Version 4.0, see the SPD.

Table 6-6: Futurebus+ Adapters and Interconnects

Interconnect	Adapters
FDDI	Digital Equipment Corporation (DEFZA-AA)
HiPPI	From Aeon System, Inc and Myriad Logic
IPI	From GENROCO, Inc.

For more information on the Futurebus+ adapters and interconnects in Table 6-6, see the *Alpha Systems Handbook* and the SPD.

6.2.5 SCSI Bus

The Small Computer Systems Interface (SCSI) bus is an ANSI standard for the interconnection of computers with each other and with disks, floppies, tapes, printers, optical disks, and scanners. The SCSI standard includes all the mechanical, electrical, and functional requirements needed for these devices to interconnect.

Digital UNIX Version 4.0 supports the SCSI CAM (Common Access Method) architecture, which defines a software model that provides a standard, hardware-independent interface for SCSI devices. The hardware independence is achieved by using the Transport (XPT) and SCSI Interface Module (SIM) components of CAM. Thus, because the XPT/SIM interface is defined and standardized, users can write SCSI/CAM peripheral device drivers for a variety of devices and use the existing Digital UNIX Version 4.0 support for SCSI. For more information on SCSI/CAM, see *Writing Device Drivers for the SCSI/CAM Architecture Interfaces.*

Digital UNIX Version 4.0 supports fast SCSI buses (maximum transfer rate of 10 megatransfers per second) and slow SCSI buses (maximum transfer rate of 5 megatransfers per second) which can be either wide (16 bits per data unit) or narrow (8 bits per data unit).

Data transfer rates are individually negotiated with each device attached to a given SCSI bus. For example, a 4 MB per second device and a 10 MB per second device may share a fast narrow bus. When the 4 MB per second device is using the bus, the transfer rate is 4 MB per second. When the 10 MB per second device is using the bus, the transfer rate is 10 MB per second. However, when faster devices are placed on a slower bus, their transfer rate is reduced to allow for proper operation in that slower environment.

Note that the speed of the SCSI bus is a function of cable length, with slow, single-ended SCSI buses supporting a maximum cable length of 6 meters, and fast, single-ended SCSI buses supporting a maximum cable length of 3

meters. In addition, there are differential adapters (such as the DWZZA, KZTSA, or KZPSA) to increase the maximum cable length to 25 meters.

Table 6-7 illustrates the frequency and transfer rate for baseboard SCSI buses:

Table 6-7: Baseboard SCSI Frequency and Transfer Rates

Processor	Bus Size	SCSI Transfer Rate
AlphaStation 200 series	Slow/Narrow	5 MB/s
	Fast/Narrow	10 MB/s
AlphaStation/Server 400 series	Slow/Narrow	5 MB/s
	Fast/Narrow	10 MB/s
AlphaStation 600 series	Fast/Wide	20 MB/s
AlphaServer 1000 series	Fast/Narrow	10 MB/s
AlphaServer 8000 series	Fast/Wide	10 MB/s 20 MB/s
DEC 2000 series	Fast/Narrow	10 MB/s
DEC 3000 Model 300 DEC 3000 Model 400 DEC 3000 Model 500 DEC 3000 Model 700	Slow/Narrow	5 MB/s
DEC 3000 Model 600 DEC 3000 Model 800 DEC 3000 Model 900	Fast/Narrow	10 MB/s
DEC 4000 series	Slow/Narrow	3.5 MB/s
	Fast/Narrow	5 MB/s (with an external connector) 10 MB/s (with no external connector)
DEC 2100 Model 500	Fast/Narrow	10 MB/s

Table 6-8 illustrates the frequency and transfer rate for SCSI adapters:

Table 6-8: SCSI Adapter Frequency and Transfer Rates

Adapter	Host Bus	SCSI BUS Size	SCSI Transfer Rate
KZPAA	PCI	Fast/Narrow	10 MB/s
KZPBA-BB (Differential)	PCI	Fast/Wide	10/20 MB/s
KZPSA (Differential)	PCI	Fast/Wide	10/20MB/s
Adaptec AHA1740	EISA	Fast/Narrow	10 MB/s
KZPSM	PCI	Fast/Wide	10/20 MB/s
KZPDA	PCI	Fast/Wide	10/20 MB/s
KZTSA(Differential)	TC	Fast/Wide	10/20 MB/s
KZMSA	XMI	Fast/Narrow	10 MB/s
PMAZB	TC	Slow/Narrow	5 MB/s
PMAZC	TC	Fast/Narrow	10 MB/s

Note that all adapters are single-ended, except for KZPSA and KZPBA-BB.

For more specific information on supported SCSI buses in Digital UNIX Version 4.0, see the SPD.

Digital UNIX Version 4.0 also supports the following functionality on SCSI buses:

- Command Tagged Queueing
- Redundant Array of Independent Disks (RAID)

6.2.5.1 Command Tagged Queueing

Command Tagged Queueing, is supported on these processors and adapters:

- DEC 3000 series processors
- DEC 4000 series processors
- KZPAA, KZPBA, KZPSA, KZPSM, KZPDA, KZTSA
- Adaptec 174X
- PMAZB, PMAZC

This feature allows a device to accept multiple concurrent commands. Since multiple commands can be accepted by the device before earlier commands are completed, the device can optimize its operation for improved performance. This allows for improved "pipelining" of requests into the device.

6.2.5.2 Redundant Array of Independent Disks

The Redundant Array of Independent Disks (RAID) enhances I/O performance and reliability by supporting such things as disk shadowing and the breaking up of data between several disks (called striping). Digital UNIX Version 4.0 supports two SCSI RAID controllers (HSZ10 and HSZ40) whose devices present themselves to the operating system as standard SCSI disks.

All RAID controllers support various levels of shadowing and striping, ranging from 0 to 5. The SCSI RAID controllers support RAID levels 0, 1, and 5, with level 3 supported on controllers that can support disk access to logical block sectors of 512 bytes.

Support consists of the following:

- Level 0

  Striping without redundancy
- Level 1

  Shadowing
- Level 3

  Striping with dedicated parity drive
- Level 5

  Striping and parity

6.2.6 TURBOchannel Bus

The TURBOchannel bus is a synchronous, 32-bit, asymmetrical I/O channel that can be operated at any fixed frequency in the range 12.5 MHz to 25 MHz and, as Table 6-1 indicates, is supported on DEC 3000 series processors. It is also an open bus, developed by Digital, whose architecture and interfaces are publicly documented.

At 12.5 MHz, the peak data rate is 50 MB per second. At 25 MHz, the peak data rate is 100 MB per second.

Table 6-9 illustrates the frequency and transfer rate for various DEC 3000 series processors that support the TURBOchannel bus.

Table 6-9: TURBOchannel Frequency and Transfer Rates

Processor	TURBOchannel Frequency	TURBOchannel Transfer Rate
DEC 3000 Model 900 DEC 3000 Model 800 DEC 3000 Model 600 DEC 3000 Model 500	25.0 MHz	100 MB/s
DEC 3000 Model 400	22.2 MHz	88.8 MB/s
DEC 3000 Model 300 DEC 3000 Model 700	12.5 MHz	50 MB/s

The TURBOchannel is asymmetrical in that the base system processor and system memory are defined separately from the TURBOchannel architecture. The I/O operations do not directly address each other. All data is entered into system memory before being transferred to another I/O option. The design facilitates a concise and compact protocol with very high performance.

Table 6-10 provides a list of some of the TURBOchannel adapters and interconnects that are available from both Digital and third-party vendors.

Table 6-10: TURBOchannel Adapters and Interconnects

Interconnect	Adapter
SCSI	PMAZB-AA Slow Narrow Single-ended (SNS) PMAZC-AA Fast Narrow Single-ended (FNS) KZTSA Fast Wide Differential (FWD)
FDDI	DEFTA-AA DEFZA-AA
Token Ring	DETRA
ATM	Digital DGLTA
IPI	IPI-240T
Ethernet	PMAD-AA

Table 6-10: (continued)

Interconnect	Adapter
Graphics	PMAGB-BE PMAGB-FE PMAGB-JA PMAG-FA
Prestoserve NVRAM	PMTNV-AA

For more specific information on supported TURBOchannel adapters and interconnects in Digital UNIX Version 4.0, see the SPD.

6.2.7 XMI Bus

The XMI bus is a 64-bit wide parallel bus that can sustain a 100 MB per second bandwidth in a single processor configuration, and, as Table 6-1 indicates, is supported on the DEC 7000, 8000, and 10000 series processors. The bandwidth is exclusive of addressing overhead; the XMI bus can transmit 100 MB per second of data.

The XMI bus implements a "pended protocol" design so that the bus does not stall between requests and transmissions of data. Several transactions can be in progress at a given time. Bus cycles not used by the requesting device are available to other devices on the bus. Arbitration and data transfers occur simultaneously, with multiplexed data and address lines. These design features are particularly significant when a combination of multiple devices has a wider bandwidth than the bus itself.

Table 6-11 provides a list of XMI adapters and interconnects. For more specific information on supported XMI adapters and interconnects in Digital UNIX Version 4.0, see the SPD.

Table 6-11: XMI Adapters and Interconnects

Interconnect	Adapter
CI	CIXCD-AX
FDDI	DEMFA
SCSI	KZMSA
SDI/STI UQPORT DSA (Digital Storage Architecture)	KDM70
NI	DEMNA

6.2.7.1 CI and KDM Controllers

The Computer Interconnect (CI) and KDM controllers, supported on the XMI bus of the DEC 7000, 8000, and 10000 series processors, interconnect multiple CPUs with various RA disks and TA tapes. The CI, through the CIXCD-AX adapter, allows DEC 7000, 8000, and 10000 series processors to connect to Hierarchical Storage Controllers (HSCs), which in turn are attached to various RA disks and TA tapes.

The maximum transfer rate for the CI is 70 MB per second per channel and Digital UNIX supports two channels. The CI driver will automatically detect the presence of multiple channels and alternate between them to improve maximum throughput.

The KDM controller allows DEC 7000, 8000, and 10000 series processors to connect directly to RA or TA devices without having to use a CI or HSCs.

In both CI and KDM environments, the RA devices are capable of operating on a large number of requests at the same time. This allows for improved performance due to increased "pipelining," and the overlapping of operations.

6.2.8 VME Bus

Digital UNIX includes a generic VME interface layer that provides customers with a consistent interface to VME devices across Alpha AXP workstation and server platforms. Currently, VME adapters are only supported on the TURBOchannel bus. To use the VME interface layer to write VMEbus device drivers, you must have the Digital UNIX TURBOchannel/VME Adapter Driver Version 2.0 software (Software Product Description 48.50.00) and its required processor and/or hardware configurations (Software Support Addendum 48.50.00-A).

For more information on VME Bus characteristics, consult the release notes.

Development Environment 7

7.1 Overview

The development environment in Digital UNIX Version 4.0 is fully ANSI C/ISO C compliant; offers the programming features of both BSD and System V UNIX and is compliant with most current standards, including POSIX, XPG4, and XPG4-UNIX; features debuggers that support C, Assembler, FORTRAN (F77 and F90), C++, Ada, and connecting to `/proc`; supports shared libraries, threads, versioning; and has a fully optimized C compiler that produces extremely efficient code to exploit fully the 64-bit address space of the Alpha architecture.

In addition, Digital UNIX Version 4.0 supports internationalization, standard UNIX development tools such as `awk`, `lint, make`, and `prof`, and provides various run-time libraries such as C++ and FORTRAN.

The following sections highlight the major functionality in the development environment. For more detailed information on the development environment, see the *Programmer's Guide*, the guide *Programming Support Tools*, *Assembly Language Programmer's Guide*, and *Writing Software for the International Market.*

7.2 Compiler

The Digital UNIX Version 4.0 C compiler was designed to support 64-bit data types and is NIST-validated for compliance with the ANSI Standard for C. The C front end supports both 64-bit addressing and the interfaces to the System V shared libraries.

The GEM-based DEC C compiler, accessed optionally in previous releases through the `-migrate` switch, is now the default compiler; access to the older MIPS-based compiler is still available through the `-oldc` switch on the cc and c89 command lines.

DEC C uses Digital's backend compiler technology (GEM), which has been specifically developed and optimized for use with Alpha systems. Both compilers have full binary compatibility with each other.

In addition, the compiler:

- Compiles C dialects of user choice including:
  - K&R C (`-std0 mode`)
  - Strict ANSI C (`-std1 mode`)
  - ANSI C with extensions (`-std mode`)
- Supports the XPG4-UNIX standard
  - By default, under the `c89` command
  - With the `-D_XOPEN_SOURCE_EXTENDED` option to `cc`.

    For more information on the various standards supported by Digital UNIX, see the `standards`(5) manpage.
- Supports floating point/double precision operations in the following two modes:
  - IEEE support (including proper handling for exceptional conditions like NaN, INF, and so forth)
  - Fast Math mode (INF, NaN, and so forth, translated to avoid exception handling)
- Supports the following language extensions:
  - C++ style structured exception handling by using `try...except` and termination handling using `try...finally`
  - User-defined assembly language sequences using `asm` sequences
  - 32-bit pointers to help reduce the amount of memory used by dynamically allocated pointers, and to facilitate the porting of 64-bit hostile programs
  - Linking programs in 32-bit address space to facilitate the porting of 64-bit hostile programs
  - Pragmas for controlling alignment of structures

For more information on the Digital UNIX C compiler, see the `cc`(1) reference page.

7.3 Debuggers

Digital UNIX Version 4.0 supports the following two source code debuggers:

- `dbx`
- `ladebug`

7.3.1 The dbx Debugger

The `dbx` debugger supports debugging programs written in C, FORTRAN, Assembler, Cobol, and Pascal. It supports debugging active kernels, either locally or remotely; analyzing kernel crash dumps; debugging program core dumps; shared libraries; and, through `/proc`, attachment to running processes and programs using multiple threads. It can also patch the on-disk copy of either user programs or the kernel. The `dbx` debugger also supports multiprocess debugging and allows debugging through `fork` and `exec` calls.

7.3.2 The ladebug Debugger

The `ladebug` debugger is a source level, object-oriented symbolic debugger that has both a graphical user interface (GUI) and a command-line interface similar to the `dbx` command-line interface, Note that the GUI is also integrated with FUSE and can be accessed from the Common Desktop Environment (CDE).

The `ladebug` debugger supports the following functionality:

- Attaching to and detaching from running processes.
- Loading programs to the debugger.
- Detecting and debugging across `fork` and `exec`.
- Debugging multiple processes
- Debugging multithreaded programs; either DECthreads applications or kernel modules that make use of kernel threads.
- Debugging programs written in C++, C, Fortran 77, Fortran 90, Ada, Cobol, and Assembler

  Note that `ladebug` is a full C++ debugger which demangles C++ names, understands C++ expressions, provides support for inline functions, templates, and C++ exceptions.

  Also note that the support for F77/F90 includes case insensitivity, common blocks, alternate entry points, language-dependent type printing, and assume shape arrays.
- Debugging machine level code
- Debugging running programs or core dumps
- Debugging Shared Objects
- Catching unaligned access problems
- Debugging active kernels, either locally or remotely, and analyzing kernel crash dumps

- Evaluating expressions using the syntax of the source programming language
- Remote debugging of programs running on different target machines (such as EB64, EB64+, and EB66 evaluation boards from Digital's Semiconductor Engineering Group) by way of the remote debugging server.

  Note that the `ladebug` remote debugging protocol is also available along with the C source code for a sample remote debugging server that adheres to the protocol.
- Internationalization

  Note that internationalization support is available in a separate kit. The internationalized `ladebug` debugger accepts multibyte characters as input, and outputs local language characters according to the current global locale set in the debugger. It also supports the `wchar_t` datatype in C/C++.

7.4 Profiling Tools

Digital UNIX Version 4.0 supports the following profiling toolkit:

- ATOM

  Provides a flexible code instrumentation interface that is capable of building a wide variety of user-defined program analysis tools and comprises an instrumentation control tool and a library whose procedural interface enables programmers to easily develop special-purpose instrumentation/analysis tools.

  ATOM provides the following built-in instrumentation/analysis tools:

  – `hiprof`

    A call-graph profiling tool with output that can be post-processed by `gprof`.

  – `third` (Third degree)

    Finds memory leaks and checks for incorrect memory accesses.

  – `pixie`

    A superset of the existing `pixie` basic block profiler which can profile a program's executables and its shared libraries. The output of `pixie` can be analyzed by `prof`.

Digital Unix V4.0 supports the following profiling tools:

- `gprof`

  For programs compiled with the `-pg` option, displays how many calls

named procedures made to each other and how much CPU time each procedure consumed, using PC-sampling statistics. Also analyzes the output of programs instrumented with `hiprof`.

- `prof`

  For programs compiled with the `-p` option, displays how much CPU time was consumed by each procedure in a program and its shared libraries, using PC-sampling statistics. Also analyzes the output of programs instrumented with `pixie` or monitored with `uprofile/kprofile`.

- `uprofile/kprofile`

  Sample a variety of events in the CPU using the Alpha chip's built-in performance counters during the execution of an application program or the kernel itself. Can report on CPU cycles, memory/cache effects, and so forth, which `prof` can then analyze.

For more information on profiling tools, see the *Programmer's Guide* and the appropriate reference pages.

7.5 Shared Libraries

Digital UNIX Version 4.0 provides a full complement of dynamic shared libraries, compatible with System V semantics for shared library loading and symbol resolution as well as the System V API for dynamic loading (`dlopen`, `dlclose`, `dlsym`, and `dlerror`). Because they allow programs to include only information about how to load and access routines rather than the routines themselves, shared libraries increase system performance, reduce disk and memory requirements, and simplify system management.

Digital UNIX Version 4.0 supports the shared libraries described in the following two tables.

Table 7-1: Digital UNIX Version 4.0 Shared Libraries

Library /usr/shlib	Description
`libDXm.so`	Digital Motif Extensions library
`libDXterm.so`	DECterm widget library, used by `dxterm`
`libDtHelp.so`	CDE online help routines
`libDtMail.so`	Shared library support for the `dtmail` CDE mail utility
`libDtSvc.so`	CDE service routines for desktop management

Table 7-1: (continued)

Library /usr/shlib	Description
libDtTerm.so	Shared library support for the CDE ddterm terminal emulator utility
libDtWidget.so	shared library of CDE widgets to supplement Motif widget
libICE.so	Inter-Client Exchange library, which enables the building of protocols
libMrm.so	Motif Resource Manager library
libSM.so	The X Session Management Protocol (XSMP) provides a uniform mechanism for users to save and restore their sessions using the services of a network- based session manager. It is built on ICE and is the C interface to the protocol.
libUil.so	The callable Motif UIL (User Interface Language) compiler used by applications that want to compile UIL at run time.
libX11.so	Xlib library
libXETrap.so	X Extension Library
libXaw.so	X Athena Widgets run-time library
libXext.so	X Client-side Extension library
libXi.so	X Input Extension client-side library
libXIE.so	X Imaging Extension client-side run-time library (V5)
libXie.so	X Imaging Extension client-side run-time library (V3)
libXm.so	Motif Widgets library
libXmu.so	X Miscellaneous utilities run-time library
libXt.so	X Intrinsics library
libXtst.so	A library of routines for X clients to make use of the XTEST Extension.
libXv.so	X video Extension client-side run-time library
libaio.so	POSIX realtime asynchronous I/O functions
libaio_raw.so	POSIX realtime asynchronous I/O functions (raw disk and tape only)
libaud.so	C2 security auditing library
libbkr.so	Motif Help System library
libc.so	C library
libc_r.so	Threadsafe libc (link to libc.so)
libcda.so	CDA run-time library
libcdrom.so	Rock Ridge Extensions to CDFS library

Table 7-1: (continued)

Library /usr/shlib	Description
libchf.so	CDA/Imaging signal handling routines
libcmalib.so	CMA threads library
libcsa.so	Shared library portion of the CDE dtcm calendar manager utility
libcurses.so	Curses screen control library
libcxx.so	NEW
libdb.so	NEW
libdnet_stub.so	DECnet library
libdps.so	Adobe Display PostScript client-side run-time libraries
libdpstk.so	Adobe Display PostScript toolkit
libdvr.so	CDA run-time viewer library
libdvs.so	CDA run-time layout library
libesnmp.so	NEW
libexc.so	Library that provides support for exception handling.
libiconv.so	Internationalization codeset conversion routines
libids.so	Image display services library
libids_nox.so	Image display services not dependent on X
libimg.so	Image processing routines
libips.so	Image processing routines
libm.so	Digital Portable Mathmatics Library (DPML)
libmach.so	Mach library
libmxr.so	Library used by mxr, the ULTRIX binary interpreter for OSF/1
libndb.so	NEW
libots.so	Compiler run-time support
libpacl.so	NEW
libproplist.so	VFS Extended File Attributes library
libpset.so	NEW
libpsres.so	Adobe Display PostScript resource utilities
libpthread.so	Application Programming Interface for Digital UNIX's threads
libpthreads.so	DECthreads library
libsecurity.so	C2 security library

Table 7-1: (continued)

Library /usr/shlib	Description
libsm_x.so	Systems Management Graphical support library; no user-level interfaces available.
libtcl.so	Base Tool Command Language (TCL) support library
libtclx.so	Extended TCL support library
libtk.so	Graphical TCL (TK) Extensions library
libtkx.so	Graphical Extended TCL support library
libtli.so	XTI library
libtt.so	SunSoft Tooltalk routines
libvxvm.so	LSM utility library
libmsfs.so	AdvFS system call interface library
libfilsys.so	File system utility library
libxnet.so	NEW
libxti.so	XTI library

Table 7-2: Digital UNIX Version 4.0 Shared /usr/shlibi/X11 Libraries

Library /usr/shlib/X11	Description
libXau.so	X Authorization library
libXdmDecGreet.so	Motif loadable greeter library
libXdmGreet.so	Athena-style loadable greeter library
libXdmcp.so	X Display Manager control program library
lib_adobe_dps.so	Adobe Display PostScript Extension library
lib_dec_cirrus.so	Device support for the Cirrus VGA graphics card
lib_dec_ffb.so	Supports the sfb+ graphics accelerator for 2D and 3D drawing operations
lib_dec_sfb.so	Device support for the smart frame buffer (HX)
lib_dec_smt.so	Shared memory transport library
lib_dec_tx.so	Device support for the TX graphic adapter
lib_dec_ws.so	Low-layer operating system interface for the X server

Table 7-2: (continued)

Library /usr/shlib/X11	Description
lib_dec_xi_pcm.so	Dynamically-loadable X Input Extension library that supports the dial and box
lib_dec_xi _serial_mouse.so	Support library for the serial mouse
lib_dec_xv_tx.so	X Video Extension support for the TX graphic option
libcfb.so	Color frame buffer library
libcfb16.so	16-bit visual support for the color frame buffer
libcfb32.so	32-bit visual support for the color frame buffer
libdbe.so	DOUBLE-BUFFER Extension library
libdix.so	Device-independent portion of the X Server
libdixie.so	With libmixie.so, supports the X Image Extensions (XIE) Extension library
libextMITMisc.so	MIT-SUNDRY-NONSTANDARD Extension library
libextMultibuf.so	Multi-Buffering Extension library
libextScrnSvr.so	MIT-SCREEN-SAVER Extension library
libextSync.so	SYNC Extension library
libextXCMisc.so	XC-MISC Extension library
libextbigreq.so	BIG-REQUESTS Extension library
libextkme.so	Keyboard-Management-Extension
libextshape.so	SHAPE Extension library
libextshm.so	MIT-SHM Extension library
libextxtest.so	XTEST Extension library
libextxtrap.so	DEC-XTRAP Extension library
libfont.so	Font access library
libfr_Speedo.so	Loadable font renderer library
libfr_Type1.so	Loadable font renderer library
libfr_fs.so	Loadable X Server font renderer for using a font server
libmfb.so	Monochrome frame buffer support
libmi.so	Machine-independent portion of the X Server
libmixie.so	With libdixie.so, supports the X Image Extensions (XIE)
libos.so	Operating-system dependent portion of the X Server
libxinput.so	X Input Extension server-side library

Table 7-2: (continued)

Library /usr/shlib/X11	Description
libxkb.so	XKEYBOARD Extension library

Note

Digital UNIX Version 4.0 also provides static versions of these libraries.

7.5.1 Quickstart

Digital UNIX Version 4.0 supports **quickstart** which allows shared libraries with unique addresses to start faster than if their addresses were in conflict. Essentially, each shared library must have a unique address placed in the `/usr/shlib/so_locations` file which allows applications that link against these shared libraries to start execution faster since the shared objects do not have to be relocated at run time. The `ld` utility can read and write an `so_locations` file when it creates a shared library.

7.5.2 Dynamic Loader

Digital UNIX Version 4.0 uses a System V Release 4.0 compatible loader to load shared libraries dynamically. This loader provides the following enhanced features:

- Calling into dynamically loaded shared libraries
- System V Release 4.0 symbol resolution semantics, including symbol preemption
- Prelinking of libraries for fast program loading

7.5.3 Versioning

Digital UNIX Version 4.0 supports full and partial duplication of shared libraries. The loader looks for backward compatible versions of shared libraries using a path constructed by appending the version string as a subdirectory of the normal search path. As a result, any changes to kernel interfaces or to global data definitions that would ordinarily break binary compatibility will not affect your applications, since you can maintain multiple versions of any shared library and link your application against the appropriate version of that shared library.

In Motif Version 1.2, for example, the OSF changed several of the interfaces, thereby breaking binary compatibility with applications built against Motif 1.1.3 libraries. To preserve binary compatibility, Digital UNIX Version 4.0

supports both Motif 1.1.3 and Motif 1.2.3 shared libraries in Digital UNIX Version 4.0 with our versioning functionality, so that applications that need to can access the Motif 1.1.3 shared libraries. For more information on versioning, see the *Programmer's Guide.*

7.6 Run-Time Libraries

Digital UNIX Version 4.0 supports the following run-time libraries

- DEC Ada Run-Time Libraries (RTL)

  The DEC Ada run-time library `libada` enables users to run previously compiled DEC Ada programs without having to install DEC Ada on their system. These libraries support such DEC Ada run-time functionality as tasking, exceptions, timer services, and miscellaneous computations.

- DEC C++ Run-Time Libraries (RTL)

  The DEC C++ run-time libraries (`libcxx`, `libcomplex`, and `libtask`) enable users to run previously compiled DEC C++ programs without having to install DEC C++ on their system. These libraries support such DEC C++ run-time functionality as I/O, complex arithmetic, and multitasking.

- DEC COBOL Run-Time Libraries (RTL)

  The DEC COBOL run-time libraries (`libcob`, `libots2`, and `libisamstub`) enable users to run previously compiled DEC COBOL programs without having to install DEC COBOL on their system. These libraries support such COBOL run-time functionality as I/O, decimal arithmetic, COBOL ACCEPT/DISPLAY statements, STRING/UNSTRING operations, and CALL and CANCEL.

- DEC FORTRAN Run-Time Libraries (RTL)

  The DEC FORTRAN run-time libraries (`libfor`, `libfutil`, and `libUfor`) enable users to run previously compiled DEC FORTRAN programs without having to install DEC FORTRAN on their system. These libraries support such FORTRAN run-time functionality as I/O, intrinsic functions, data formatting, data conversion, miscellaneous math functions, and FORTRAN bindings to common operating system services.

- DEC Pascal Run-Time Libraries (RTL)

  The DEC Pascal run-time library `libpas` enables users to run previously compiled DEC Pascal programs without having to install DEC Pascal on their system. These libraries support such Pascal run-time functionality as I/O, miscellaneous math functions, time and date services, and miscellaneous file services.

7.7 Development Commands

Digital UNIX Version 4.0 supports the full array of development tools, including `ar`, `as`, `btou`, `cb`, `cc`, `cflow`, `cpp`, `ctags`, `cxref`, `dbx`, `dis error`, `file`, `indent`, `ld`, `lex`, `lint`, `loader`, `m4`, `make`, `mig`, `mkstr`, `nm`, `odump`, `pixie`, `ppu`, `prof`, `ranlib`, `size`, `stdump`, `strings`, `strip`, `tsort`, `uopt`, `uld`, `utob`, `xstr`, and `yacc`, as well as the source code control systems `rcs` and `sccs`.

Note that many of the development commands are specified by the System V, POSIX, XPG4 and XPG4-UNIX standards to which Digital UNIX is fully complaint. Also note that Digital UNIX supports both the OSF `make` command and the ULTRIX version of `make`, since the ULTRIX `make` command is POSIX 1003.2 compliant and more robust.

7.8 DECthreads

DECthreads is a library of routines built on the basic Mach threads capabilities in the OSF code that support the development of multithreaded applications on Digital UNIX. DECthreads is an implementation the POSIX 1003.1c-1995 standard API and also provides a proprietary API to aid in porting applications from other Digital platforms such as OpenVMS. Note that DECthreads also provides an implementation of draft 4 of the POSIX 1003.1c (formerly known as 1003.4a) specification which will be retired in the next release. This implementation is being provided only to allow applications extra time to convert their draft standard implementation to the finalized POSIX standard interface.

DECthreads is compatible with DCE requirements for threads and is the threads library used by Digital's DCE product. In addition, DECthreads is integrated with the Digital UNIX kernel, providing SMP capabilities for multithreaded applications and realtime scheduling policies and priorities for multithreaded realtime applications.

7.9 Thread Independent Services

Digital UNIX supports Thread Independent Services (TIS) routines, which are provided to enable application writers to write thread-safe code for non-threaded libraries and applications. In the presence of threads, these routines provide the indicated thread-safe functionality. In the absence of threads, these routines impose the minimum possible overhead on their caller. Note that the TIS routines are used by the C runtime library to provide support for both single and multithreaded applications.

7.10 Memory-Mapped File Support (mmap)

Digital UNIX Version 4.0 supports the Berkeley `mmap` function and therefore allows an application to access data files with memory operations rather than file I/O operations.

7.11 Realtime

Digital UNIX Version 4.0 supports a realtime user and programming environment, developed by Digital and shipped as an optional realtime subset. The Digital UNIX Version 4.0 realtime programming environment conforms to the POSIX 1003.1b-1993 standard for realtime which allows you to develop and run portable realtime applications in a POSIX environment. The realtime interfaces are collected in the static and shared libraries `/usr/ccs/lib/librt.a` and `/usr/shlib/librt.so`, respectively.

If you enable kernel preemption, a higher-priority process can preempt a lower-priority process regardless of whether it is running in kernel mode or user mode. With this fully preemptive kernel, the Process Preemption Latency (the amount of time it takes to preempt a lower-priority process) is minimized.

In addition to a preemptive kernel, the Digital UNIX Version 4.0 realtime programming environment supports the following POSIX 1003.1b features:

- Realtime clocks and timers
- Realtime Queued Signals
- Fixed priority scheduling policies
- Realtime scheduler priority levels
- Counting Semaphores
- Shared memory
- Process memory locking
- Asynchronous I/O
- Synchronized I/O
- Message-passing interfaces
- Thread-safe implementation of realtime libraries

For more information on the realtime programming environment, see the *Guide to Realtime Programming*. For information on configuring the realtime kernel, see the *System Administration* guide.

Windowing Environment 8

8.1 Overview

Digital UNIX Version 4.0 supports a full-featured implementation of the X Consortium's X Window System, Version 11, Release 6 (X11R6) up to and including public patch 12, as well as the complete Motif Toolkit from CDE/Motif Version 1.0. Aside from Digital extensions to the X server to add graphics support, and the addition of several Motif-based Digital X clients, the windowing code is essentially passed through untouched from the X and CDE consortiums.

The Digital UNIX Version 4.0 implementation of X11R6 and Motif makes use of both static and shared libraries, allowing client programs that link shared to make use of the latest library code without recompiling, as well as saving memory and disk space.

For more information on shared libraries, see The following sections briefly discuss the Common Desktop Environment (CDE), which is the default graphical user-interface now supported in Digital UNIX, and the X11R6 and Motif components of the Digital UNIX Version 4.0 windowing environment.

8.2 Common Desktop Environment

The Common Desktop Environment is the new default user interface under Digital UNIX Version 4.0.

The Common Desktop Environment (CDE) provides an easy method of interacting with the Digital UNIX operating system. It is a jointly developed graphical user interface based on industry standards which include the X Consortium's X Window System and the Open Software Foundation's Motif user interface. CDE is an X/Open standard which provides a consistent look and feel as well as common APIs across multivendor platforms.

CDE presents a visual desktop that you can customize. Using the CDE interface, you can use the mouse or keyboard to navigate and interact with applications. The desktop itself offers a Front Panel, which is a graphical display at the bottom of the screen area that provides access to applications, printers, and frequently used objects, including online help.

In addition to user services, CDE provides everything needed to implement fully integrated applications. Because CDE is standards based, such integration work is transportable to other platforms that comply with this standard. For example, the help files and the means to access them apply across all compliant platforms. For more information, see the *Common Desktop Environment: Programmer's Overview.*

The CDE Front Panel displays the tools that you use to start applications, manage tasks in a desktop session, or change workspaces. Each tool is represented by an icon that indicates its purpose. A workspace is the screen itself, which includes the Front Panel. A tool on the Front Panel is provided to switch between different workspaces.

The tools available on the Front Panel are described in Table 8-1. For detailed information on the use of each tool, see the *Common Desktop Environment: User's Guide.*

Table 8-1: Front Panel Tools

Application	Use of Application
Clock	Displays the time of day in analog format. Clicking on this tool does not perform an action.
Calendar	Displays the current month and day. Use this application to schedule appointments and To Do Items, set reminders, browse other calendars, and schedule group appointments. Dropping an appointment file on the Calendar tool adds the appointment to your calendar database.
File Manager	Provides a view of directories (folders) and files. Dropping a directory on the File Manager tool opens a view of that directory.
Text Editor	Opens a Text Editor window where you can create letters or notes. Dropping a file on the Text Editor icon opens that file in a Text Editor window.
Mailer	Starts the desktop Mailer application. Use this application to send, receive, save, and forward mail messages. Dropping a file on this tool displays the contents of the file in a New Message window.
Lock	Pauses a session indefinitely. Pausing a session locks the workstation display, but applications continue to run. To resume a session, enter your password.
Workspace Switches	Changes workspaces. Use this switch to move to different work areas. There are four workspaces by default.

Table 8-1: (continued)

Application	Use of Application
Busy Light	Indicates that an action is being performed. For example, when you start an application, the light blinks. Once the call to the application is complete, the busy light stops blinking. Clicking on this icon does not produce an action.
Printer	Displays the status of the default printer. Dropping a file on the Printer icon prints that file on the default printer.
Exit	Starts the logout process for a session.
Style Manager	Opens the Style Manager application. Use this application to change the characteristics of your environment.
Application Manager	Starts the Application Manager, which is a container for applications and other tools available on your system.
Help Manager	Displays the top level of available online help information. Dropping a master help volume file (`*.sdl`) onto the Help Manager opens a help viewer window and displays the contents of that volume.
Trash Can	Opens the Trash Can application. Use this application to delete files. Dropping a file on the Trash Can tool moves the file to a discard directory.

For more information, see the CDE documentation set, which consists of:

- *CDE Companion Guide*
- *CDE User's Guide*
- *CDE Advanced User's and System Administrator's Guide*
- *CDE Help System Author's and Programmer's Guide*
- *CDE Programmer's Overview*
- *CDE Programmer's Guide*
- *CDE Style Guide and Certification Checklist*
- *CDE Desktop Korn Shell User's Guide*
- *CDE Internationalization Programmer's Guide*
- *CDE Application Builder User's Guide*

- *CDE ToolTalk Messaging Overview*
- *CDE Glossary*

8.3 X Window System

The X11R6 windowing software consists of the following components:

- X Client Libraries
- X Server
- Display Manager
- X Protocol Extensions
- Font Server
- X Clients

8.3.1 X Client Libraries

Digital UNIX Version 4.0 supports the complete set of X11R6 X client libraries:

- Athena Widget Set (`libXaw`)

  A high-level library of user-interface components (scroll bars, labels, buttons)

- X Intrinsics Library (`Xt`)

  Middle-level routines that call into `Xlib`

- X library (`Xlib`)

  Low-level routines that interface with the X server

For more information on individual X client libraries, see the guides *X Window System* and *X Window System Toolkit*.

8.3.2 X Server

Through the extensive use of shared libraries, Digital UNIX Version 4.0 supports a single X11R6 X server image for all graphic options. The Digital UNIX Version 4.0 X server dynamically configures itself at init-time, loading only those server components required by a specific system configuration, and rarely requires any intervention by a system administrator.

For a list of the shared libraries that make up the X server, see Chapter 7.

8.3.2.1 Multihead Graphic Support

Multihead graphic support is transparent in Digital UNIX Version 4.0, provided the proper option cards are installed and the additional graphic adapters are built into the kernel.

For more information on the graphic options supported in Digital UNIX Version 4.0, see the *Systems and Options Catalog*.

8.3.2.2 X Server Extensions

Digital UNIX Version 4.0 supports the following X server extensions. Note that to conserve memory, the X server, by default, defers loading most server extensions until it receives a request from a client for that specific extension.

- XKB

  The X Keyboard Management extension. Provides support for the ISO 9995 standard and includes the AccessX keyboard extension. Included with the XKB extension are XKB keymaps for all the keyboards/locales we support as well as many supporting applications.

  The XKB is turned on by default, and the X server automatically compiles and loads a keymap based upon the console language and keyboard.

- The Keyboard Extension for X11R6 (XKB)

  The XKB server extension is new for X11R6 and for Digital Unix V4.0. XKB enhances control and customization of the keyboard under the X Window system by providing:

  - Support for the ISO 9996 standard for keyboard layouts

  - Compatibility with the core X keyboard handling; no client modifications needed

  - Standard methods for handling keyboard LEDs and locking modifiers such as CapsLock and NumLock

  - Support for keyboard geometry

  Additionally, the X11R5 (for versions of Digital Unix earlier than V4.0) AccessX server extension for people with physical impairments has been incorporated into the XKB server extension. These accessibility features include StickyKeys, SlowKeys, BounceKeys, MouseKeys, and ToggleKeys, as well as complete control over the autorepeat delay and rate.

- DPS (Adobe Display PostScript Extension — DPS Level II)

  Supports realtime PostScript display, including color, motion, and advanced text display to the screen.

- MIT-SHM (MIT Shared Memory)

  Enhances performance for local image-intensive applications.

- MIT-SUNDRY-NONSTANDARD

  Miscellaneous extension from the X Consortium, which currently controls bug-compatibility modes for the X Server.

- Multibuffering

  Supports smooth animations by drawing to multiple buffers.

- SHAPE

  Supports nonrectangular windows used for round, oval, and nonregular shaped windows.

- SMT (Shared Memory Transport)

  Allows for the use of shared memory as an X transport for local clients, giving a significant performance boost. Transport specified by `local:0.0`.

- XIE (X Imaging Extension, Version 3 and 5)

  Provides advanced control over imaging, as well as device-independent image display.

  Digital UNIX Version 4.0 ships both Version 3 (`/usr/lib/Xie.a` and `/usr/shlib/libXie.so`) and the de facto industry standard, Version 5 (`/usr/lib/libXIE.a` and `/usr/shlib/libXIE.so`).

- X Input

  Allows users to write their own drivers for third-party input devices, and then load them dynamically into the X server by making entries in the X server configuration file (`/usr/var/X11/Xserver.conf`). The new input devices are then recognized the next time the X server is reset.

  In traditional, statically-linked X Servers, each time a new extension device is added the X Server must be rebuilt. Digital UNIX's loadable X server implementation has overcome this limitation by permitting system administrators to add new new input device support as external shareables that are loaded by the X server at init-time.

  Sample code showing how such a driver should be written is included in the `/usr/examples` directory.

- XKME (X Keyboard Management Extension)

  Internal extension for better support of international X clients. Note that the XKME functionality has been made obsolete by the XKB extension, but has been provided for backwards compatibility.

- X Screen Saver

Enables a client to receive notification when the screen has been inactive for a specified amount of time or whenever it cycles. This extension is useful to developers writing screensaver applications.

- XSync

  The `XSync` function, in conjunction with the `XFlush`, `XEventsQueued`, and `XPending` functions, allows synchronization between X clients to take place entirely within the X server, thereby eliminating any errors introduced by the network and enabling different hosts running different operating systems to synchronize X clients. This extension is particularly useful for multimedia applications that require the synchronization of audio, video, and graphics; and for animation applications, which can have their requests synchronized to internal, X server timers.

- XTest

  Allows applications to simulate X events for testing purposes.

- XTrap

  Supports the recording and playback of X events for the purpose of X client testing.

- XV (X Video)

  Allows clients to control video options, such as the live video PIP option for the TX graphic device.

For more information on the X server, see the *X Window System Environment* and the `X`(1X) and the `Xdec`(1X) reference pages.

8.3.3 Display Manager

Digital UNIX Version 4.0 supports the standard `Xdm` terminal manager software. The `Xdm` terminal manager starts up the X server locally and allows for network-transparent login prompting, so that users can log in to any system on their network that is supported by `xdm` as if the remote system's graphic console were in front of them. This functionality provides for the seamless integration of X terminals into the Digital UNIX Version 4.0 environment. For more information on using `Xdm`, see the *System Administration* guide and the `xdm`(1X) reference page.

8.3.3.1 xmodmap Keymap Format

The keymaps supplied with Digital UNIX Version 4.0 use the `xmodmap` keymap format, the de facto industry standard. Unlike the format of the keymaps supplied in earlier versions of Digital UNIX, which was difficult to read and edit because it was written using hexadecimal numbers, the `xmodmap` format is written using symbolic key names and can be easily

customized.

Also, the xmodmap format supports the ability to specify modifier keys (Compose, Alt, Shift, and so forth), which the old format did not support.

Now, instead of the X server itself loading the keymap when it starts or resets, xdm (the X Display Manager) causes the appropriate xmodmap-format keymap to be loaded by using the xmodmap command.

The xmodmap keymap format is compatible whether the X server is running the XKB extension or not. The xmodmap keymaps, however, are being shipped for backward compatibility reasons. Digital suggests using the newer XKB standard keymap format instead of the xmodmap keymap format.

8.3.3.2 XDM-AUTHORIZATION-1

Whenever an X client application establishes a connection to the X server, it passes an authorization code, called a key, to the X server. If the X server recognizes this key, the connection is allowed. When the user's X session is started, xdm (the X Display Manager) writes one or more keys into the .Xauthority file in a user's home directory. The X Display Manager (xdm) also writes these keys into a file readable by the X server.

In previous releases of the Digital UNIX, the keys were in the MIT-MAGIC-COOKIE-1 format and were not encrypted.

Now, however, to improve security, Digital UNIX supports both the MIT-MAGIC-COOKIE-1 key format as well as the XDM-AUTHORIZATION-1 encrypted key format, which is the default.

8.3.4 Font Server

Digital UNIX Version 4.0 supports a standard scalable font server that supplies a network of systems with access to fonts resident on any Digital UNIX system. The font server maintains a repository of fonts and responds to requests from other X servers on the network for fonts that they may not have locally. In addition to providing network-transparent access to fonts, the font server unloads the compute burden of font scaling from local X servers, since it scales fonts appropriately before supplying them to the requesting X server.

8.3.4.1 Loadable Font Renderers

Before a font can be displayed by an X server, its glyphs must be converted from their on-disk formats into bitmaps. This conversion is done by font renderer code in the X server or in a font server which may be supplying fonts to the X server.

Digital UNIX Version 4.0 supports loadable font renderers, so that users who adhere to the X11R6 standard can write their own font renderer for their own set of fonts and install them on a Digital UNIX system. After the fonts and the font renderer are installed, the necessary entries for them are placed in the X server configuration file (`/usr/var/X11/Xserver.conf`), the font server configuration file (`/usr/var/X11/fs/config`), or in both configuration files. The new font renderer is then recognized the next time the X server or font server (whichever has the font renderer configured) is reset.

8.3.5 X Clients

Digital UNIX Version 4.0 supports the entire suite of X clients that ships with X11R6, including `appres`, `atobm`, `bdftopcf`, `bitmap`, `bmtoa`, `editres`, `fs`, `fsinfo`, `fslsfonts`, `fstobdf`, `getcons`, `ico`, `imake`, `listres`, `lndir`, `mkfontdir`, `oclock`, `optacon`, `pswrap`, `puff`, `puzzle`, `resize`, `showfont`, `showrgb`, `twm`, `uil`, `viewres`, `x11perf`, `x11perfcomp`, `xauth`, `xbiff`, `xcalc`, `xcd`, `xclipboard`, `xclock`, `xcmsdb`, `xcmstest`, `xconsole`, `xcutsel`, `xdm`, `xdpr`, `xdpyinfo`, `xedit`, `xemacs`, `xev`, `xeyes`, `xfd`, `xfontsel`, `xgc`, `xhost`, `xkbcomp`, `xkbprint`, `xkbdfltmap`, `xkill`, `xload`, `xlogo`, `xlsatoms`, `xlsclients`, `xlsfonts`, `xmag`, `xman`, `xmbind`, `xmh`, `xmkmf`, `xmodmap`, `xon`, `xpr`, `xprop`, `xrdb`, `xrefresh`, `xset`, `xsetroot`, `xsoundsentry`, `xstdcmap`, `xterm`, `xwd`, `xwininfo`, and `xwud`.

For more information on individual X clients, see the appropriate reference page.

8.4 Motif

Digital UNIX Version 4.0 supports the entire suite of CDE Motif Version 1.0 components, including the widget library (`Xm`), the resource manager (`Mrm`), the widget metalanguage (`wml`), the User Interface Language (UIL), the Motif window manager (`mwm`), the key binding utility (`xmbind`), and the Motif Demonstration programs (`examples`).

In Motif Version 1.2 the OSF has added support for ANSI C, Internationalization, Drag and Drop, and TearOff Menus. Unfortunately, however, much of this support required breaking binary compatibility with Motif Version 1.1.3.

To mitigate this problem, Digital UNIX Version 4.0 provides the Motif Version 1.1.3 libraries through its versioning functionality to allow applications that are linked against Motif 1.1.3 to continue to run. These libraries are available in an optional subset. For more information on versioning, see the *Programmer's Guide* and Chapter 7.

The following sections briefly discuss these Digital extensions to Motif:

- Digital Extended Widget Set
- Digital X Clients

For more information on Motif, see the *OSF/Motif Programmer's Guide*, *DECwindows User's Guide*, and the appropriate reference pages. For information on Motif support for internationalization, see Chapter 10.

8.4.1 Digital Extended Widget Set

In addition to the `Xm` widget library, Digital UNIX Version 4.0 supports the Digital Extended Widget Set (`DXm`), which contains the following widgets:

- `DXmColorMix`

  Supports editing and the selection of colors
- `DXmPrintWidget`

  Presents graphical print options
- `DXmCSText`

  Supports the editing of compound strings in a user interface similar to `XmText`
- `DXmHelpWidget`

  Displays help topics
- `DXmSvn`

  Supports structured navigation through lists of data

8.4.2 Digital X Clients

In addition to the entire suite of X clients that ships with X11R6, Digital UNIX Version 4.0 supports a variety of Digital X clients including `accessx`, `dxbook`, `dxcalc`, `dxcalendar`, `dxcardfiler`, `dxclock`, `dxconsole`, `dxdiff`, `dxkbledpabel`, `dxkeyboard`, `dxkeycaps`, `dxmail`, `dxnotepad`, `dxpaint`, `dxpause`, `dxpresto`, `dxprint`, `dxsession`, `dxterm`, and `dxvdoc`.

For more information on individual Digital X clients, see the *DECwindows User's Guide* and the appropriate reference pages.

System V Functionality 9

9.1 Overview

Digital UNIX Version 4.0 supports the following two System V packages in an effort to provide users, programmers, and administrators with complete System V Release 4 functionality:

- The System V Compatibility habitat
- The System V Environment

9.1.1 System V Compatibility Habitat

The System V Compatibility habitat ships with the base system and, in conjunction with the work done to extend the Digital UNIX Version 4.0 libraries to contain System V functionality, allows Digital UNIX Version 4.0 to conform to the following two volumes from the four volumes listed in the System V Interface Definition 3 (SVID 3):

- Volume 1: Base System and Kernel
- Volume 4: X11 Windows

  Note that NeWS Windows is not supported.

The System V Compatibility habitat consists of several commands in the `/usr/opt/s5` directory (commands that are commonly used in shell scripts and that format their data differently from the corresponding Digital UNIX commands) as well as a separate System V shared and static library (`libsys5.a` and `libsys5.so`) that contains functions that are different from those already in the standard `libc`. For the most part, however, Digital has attempted to extend the functions and system calls in `libc` to include the necessary System V functionality and behavior so that many programs written for System V can compile and run on Digital UNIX Version 4.0 without the need to link against `libsys5`.

Also, Digital has added the `swapctl` and `memcntl` System V system calls to `libc` as well as support for the System V pseudodevice, `/dev/zero`, the `/proc` file system, the FDFS file system, SVR4 signals, and SVR4 STREAMS.

Users and programmers can access the Digital UNIX Version 4.0 System V Compatibility habitat either by placing the `/usr/opt/s5` path in their `.profile` file or by using the absolute pathname for this directory. The `cc` and `ld` commands in the `/usr/opt/s5` directory for example, are in fact shell wrappers which, when called, search the `libsys5.a` and `libsys5.so` libraries before looking in `libc`, so that programmers can access these libraries transparently.

For more information on the System V Compatibility habitat, see the *System Administration* guide, *Programmer's Guide*, and the *Command and Shell User's Guide*.

9.1.2 The System V Environment

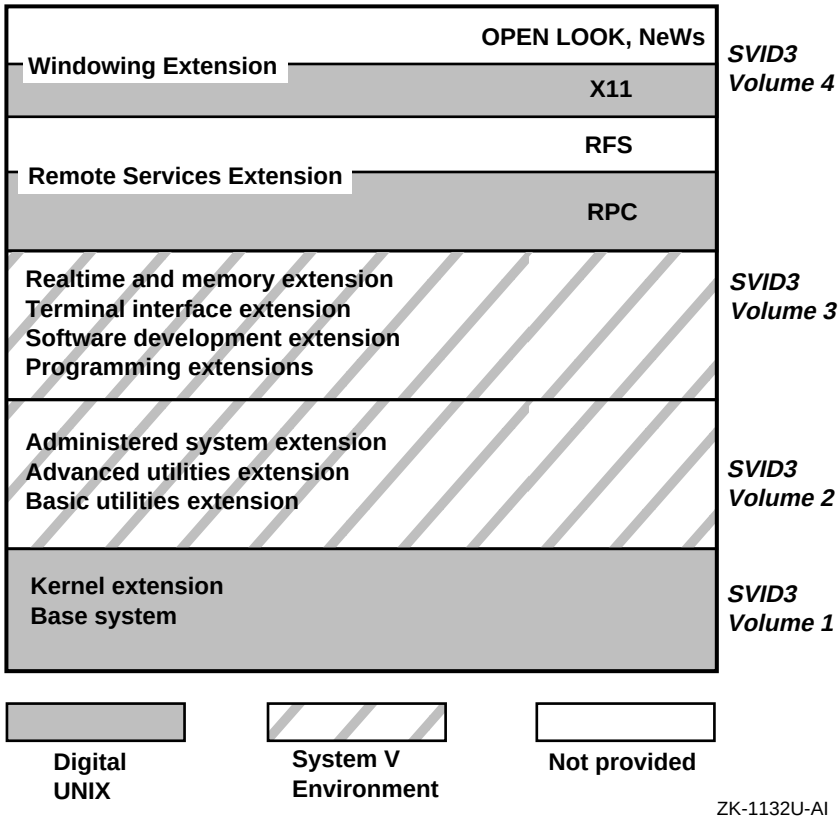
The System V Environment extends the functionality provided by the System V Compatibility habitat by supporting a more complete System V Release 4 (SVR4) environment for general users, application programmers, and system administrators. The System V Environment is an extension to the operating system that contains a separate System V Release 4.0 binary license from UNIX Software Laboratories and requires a special license and a Product Authorization Key (PAK) to access.

The System V Environment extends the base system SVID 3 compliance to provide complete compliance to the SVID 3 standard, by supporting the following two additional SVID 3 volumes:

- Volume 2: Utilities and Administration
- Volume 3: Software Development, Terminal Interface, Realtime and Memory Management, Remote Services

See Figure 9-1 for a summary of SVID support in both Digital UNIX and the System V Environment.

Figure 9-1: SVID Compliance in the System V Environment



In addition, the System V Environment meets operating system requirements critical to the telecommunications industry, as defined in the Bellcore Standard Operating Environment (SOE), Issue 2.

The System V Environment supports the following functionality:

- Development tools and libraries
- Extended Terminal Interface
- Software management utilities (`pkg*` commands)
- System administration commands and utilities, including backup and restore services
- User account management
- System activity reporting (sar)

- SVR4 Bourne Shell
- SVR4 printing subsystem
- Service access facility
- Realtime extensions

The System V Environment delivers the complete suite of SVR4 commands as well as the `libsvr4` library, which contains all the SVR4 routines for the base system. Essentially, the System V Environment extends the functionality of the System V Compatibility habitat, allowing users, programmers, and administrators to work in a completely native System V environment, without the Digital UNIX "look and feel." However, through manipulating each user's `.profile` file, each environment, System V and Digital UNIX, can be accessed without difficulty and, with the exception of the Printing Subsystem, without reconfiguring the system. Note that programmers can develop and run both Digital UNIX and SVR4 applications simultaneously, no matter which environment they choose to adopt. For more information on the System V Environment, see the *System V Environment for Digital UNIX Version 4.0 User's Guide* and the *SVR4 to SVE for Digital UNIX Version 4.0 Migration Guide.*

Internationalization 10

10.1 Overview

The term "internationalization" is formally defined by X/Open as a "provision within a computer program of the capability of making itself adaptable to the requirements of different native languages, local customs, and coded character sets," which means, essentially, that internationalized programs can run in any supported **locale** without having to be modified. A locale is a software environment that correctly handles the cultural conventions of a particular geographic area, such as China or France, and a language as it is used in that area. So by selecting a Chinese locale, for example, all commands, system messages, and keystrokes can be in Chinese characters and displayed in a way idiomatic to Chinese dialects.

Digital UNIX Version 4.0 is an internationalized operating system that not only allows users to interact with Digital UNIX Version 4.0 in their native language, but also supports a full set of application interfaces, referred to as the X/Open Worldwide Portability Interfaces (WPI), to enable software developers to write internationalized applications. The code came from the OSF and was enhanced by Digital.

The internationalization subsystem in Digital UNIX Version 4.0 is based on POSIX 1003.2 and Single UNIX specifications. Commands, utilities, and libraries (including the curses library) have been internationalized, and a set of enhanced US English message catalogs and message catalogs that supports Asian languages have been included in the base system. In addition, Digital UNIX Version 4.0 supports the X Input Method (XIM) and X Output Method(XOM) to facilitate input of local language characters, text drawing, measurement, and inter-client communication which is implemented according to the X11R6 specification.

Note that Digital UNIX Version 4.0 also supports a 32 bit `wchar_t` datatype which in turn enables support for a wide array of codesets, including the full ISO 10646 standard.

10.2 Supported Languages

Table 10-1 lists the languages supported in Digital UNIX Version 4.0 and their corresponding locales. The locales are built using new internationalization utilities and are more robust than those offered on previous versions of the operating system. Note that in Digital UNIX Version 4.0, the content of the locale definitions has changed to align with new national profiles and registered locale definitions. Those marked with an asterisk are available as part of language-variant subsets which can be optionally installed.

Table 10-1: Languages and Locales

Language	Locale Name
Catalan *	ca_ES.ISO8859-1
Simplified Chinese/PRC *	zh_CN.dechanzi zh_CN.dechanzi@pinyin zh_CN.dechanzi@radical zh_CN.dechanzi@stroke
Chinese/Hong Kong *	zh_HK.big5 zh_HK.dechanyu zh_HK.dechanzi zh_HK.eucTW
Traditional Chinese/Taiwan *	zh_TW.big5 zh_TW.big5@chuyin zh_TW.big5@radical zh_TW.big5@stroke zh_TW.dechanyu zh_TW.dechanyu@chuyin zh_TW.dechanyu@radical zh_TW.dechanyu@stroke zh_TW.eucTW zh_TW.eucTW@chuyin zh_TW.eucTW@radical zh_TW.eucTW@stroke
Czech *	cs_CZ.ISO8859-2
Danish	da_DK.ISO8859-1
Dutch	nl_NL.ISO8859-1
Belgian Dutch	nl_BE.ISO8859-1
US English/ASCII	en_US.8859-1 (C/POSIX)
US English/ISO8859-1	en_US.ISO8859-1
GB English	en_GB.ISO8859-1

Table 10-1: (continued)

Language	Locale Name
Finnish	fi_FI.ISO8859-1
German	de_DE.ISO8859-1
Swiss German	de_CH.ISO8859-1
Greek	el_GR.ISO8859-7
French	fr_FR.ISO8859-1
Belgian French	fr_BE.ISO8859-1
Canadian French	fr_CA.ISO8859-1
Swiss French	fr_CH.ISO8859-1
Hebrew *	iw_IL.ISO8859-8
Hungarian	hu_HU.ISO8859-2
Icelandic	is_IS.ISO8859-1
Italian	it_IT.ISO8859-1
Japanese *	ja_JP.eucJP ja_JP.SJIS ja_JP.deckanji ja_JP.sdeckanji
Korean *	ko_KR.deckorean ko_KR.eucKR
Lithuanian *	ko_KR.eucKR
Norwegian	no_NO.ISO8859-1
Polish	pl_PL.ISO8859-2
Portuguese	pt_PT.ISO8859-1
Russian	ru_RU.ISO8859-5
Slovak	sk_SK.ISO8859-2
Slovene *	sl_SIISO8859-2
Spanish	es_ES.ISO8859-1
Swedish	sv_SE.ISO8859-1
Thai *	th_TH.TACTIS
Turkish	tr_TR.ISO8859-9

Note that you can switch languages or character sets as necessary and can even operate multiple processes in different languages or codesets in the same system at the same time.

For information on supported character sets, see the guide *Writing Software for the International Market* and reference pages for individual languages and codesets.

10.3 Code Conversion and the iconv Utility

Digital UNIX Version 4.0 extends the base `tty` terminal driver subsystem to include additional BSD line disciplines and STREAMS `tty` modules for processing data in all languages. The line discipline or STREAMS modules sed to process Japanese, Chinese, and Korean, for example, provides the following support:

- Japanese Kana-Kanji conversion input method
- Character-based line processing in cooked mode
- Input line history and editing (BSD line discipline only)
- Input line history and editing
- Software on-demand-loading for user-defined characters
- Code conversion between terminal codeset and application codeset

Digital UNIX Version 4.0 supports the `iconv` utility, which converts text from one locale's codeset to another, thereby assisting programmers in the writing of international applications.

Code conversion is also implemented in the terminal driver and printing subsystem to allow the use of terminals and printers with different codesets. Additionally, code conversion is implemented in mail utilities for mail interchange with systems using different codesets (see the `man` command for reference page displays) and in the X Window Toolkit for text input, drawing, and interclient communication. For more information on the `iconv` utility, see `iconv_intro`(5).

For information on all the languages supported by the international terminal subsystem, see the guide, *Writing Software for the International Market.*

The following sections briefly discuss additional internationalization functionality.

10.4 Unicode Support

Digital UNIX provides a set of locales and and codeset convertors that supports the Unicode and ISO 10646 standards.The codeset convertor modules enable an application to convert between other supported codesets and UCS-4. The following UCS-4 locales are supported:

Table 10-2: Languages and Locales

Language	Locale Name
Simplified Chinese/PRC *	zh_CN.dechanzi@pinyin@ucs4 zh_CN.dechanzi@radical@ucs4 zh_CN.dechanzi@stroke@ucs4
Chinese/Hong Kong *	zh_HK.dechanyu@ucs4 zh_HK.dechanzi@ucs4 zh_HK.eucTW@ucs4
Traditional Chinese *	zh_TW.dechanyu@ucs4 zh_TW.dechanyu@chuyin@ucs4 zh_TW.dechanyu@radical@ucs4 zh_TW.dechanyu@stroke@ucs4 zh_TW.eucTW@ucs4 zh_TW.eucTW@chuyin@ucs4 zh_TW.eucTW@radical@ucs4 zh_TW.eucTW@stroke@ucs4
Czech *	cs_CZ.ISO8859-2@ucs4
Danish	da_DK.ISO8859-1@ucs4
Dutch	nl_NL.ISO8859-1@ucs4
Belgian Dutch	nl_BE.ISO8859-1@ucs4
US English/ASCII	en_US.8859-1@ucs4i@ucs4
US English/ISO8859-1	en_US.ISO8859-1@ucs4
GB English	en_GB.ISO8859-1@ucs4
Finnish	fi_FI.ISO8859-1@ucs4
German	de_DE.ISO8859-1@ucs4
Swiss German	de_CH.ISO8859-1@ucs4
Greek	el_GR.ISO8859-7@ucs4
French	fr_FR.ISO8859-1@ucs4
Belgian French	fr_BE.ISO8859-1@ucs4
Canadian French	fr_CA.ISO8859-1@ucs4
Swiss French	fr_CH.ISO8859-1@ucs4
Hebrew *	iw_IL.ISO8859-8@ucs4
Hungarian	hu_HU.ISO8859-2@ucs4
Icelandic	is_IS.ISO8859-1@ucs4
Italian	it_IT.ISO8859-1@ucs4
Japanese *	ja_JP.SJIS@ucs4 ja_JP.deckanji@ucs4

Table 10-2: (continued)

Language	Locale Name
Korean *	ko_KR.deckorean@ucs4
Norwegian	no_NO.ISO8859-1@ucs4
Polish	pl_PL.ISO8859-2@ucs4
Portuguese	pt_PT.ISO8859-1@ucs4
Russian	ru_RU.ISO8859-5@ucs4
Slovak	sk_SK.ISO8859-2@ucs4
Slovene *	sl_SIISO8859-2@ucs4
Spanish	es_ES.ISO8859-1@ucs4
Swedish	sv_SE.ISO8859-1@ucs4
Turkish	tr_TR.ISO8859-9a@ucs4
Universal	universal.utf8@ucs4

Digital UNIX also provides a function called `fold_string_w()`, which maps one Unicode string to another performing the specified Unicode transformations. For more information on the `fold_string_w()` function, see `fold_string_w`(3). For more information on Unicode support, see `Unicode`(5).

10.5 ISO-C

Digital UNIX provides support for the new ISO-C 1944 standard. This includes support for several new interfaces within libc as well as support within the new DEC C compiler.

The addition of these new routines provides a more complete coverage of routines that are `wchar_t` aware, which in turn allows Unicode to be more easily supported on the platform.

10.6 Internationalized Curses

Digital UNIX supplies an internationalized Curses library in conformance with X/Open Curses, Issue 4. This provides functions for processing single-byte and multibyte characters. Multibyte characters may be in either wide-character (`wchar_t`) or complex-character (`cchar_t`) formats. The complex-character format provides for a single logical character made up of multiple wide characters. Some of the components of the complex character may be nonspacing characters.

For information on the syntax and effect of all Curses interfaces, see `curses`(3). For a description of the enhancements provided by the internationalized Curses routines, and their relationship to previous Curses routines, see the guide, *Writing Software for the International Market.*

10.7 Printing

Digital UNIX Version 4.0 supports the printing of plain text and PostScript files for a variety of languages and provides outline fonts for high quality printing on PostScript printers. For the printing of Asian languages whose font files are typically too large to fit in printer memory, Digital UNIX Version 4.0 makes use of a unique font-faulting technology which substantially reduces memory requirements on the supported PostScript printers. For more information, on printing, see `i18n_printing`(5) and *Writing Software for the International Market.*

10.8 Creating Locales and the localedef Utility

The `localedef` utility allows programmers to create their own locales, compile their source, and generate a unique name for their new locale.

For more information on `localedef`, see the `localedef`(1) reference page.

10.9 I18N Configuration Tool

The I18N Configuration Tool, available via the CDE Application Manager, is one of the System Administration Configuration tools. It provides a graphical interface for the system administrator to configure I18N-specific settings. It also provides a convenient way to see what countries, locales, fonts, and keymaps are supported on the host. I18nconfig can be used to remove unused fonts and country support installed on the system.

10.10 Special Support for Ideogrammatic Languages

The following sections discuss special support in Digital UNIX Version 4.0 for ideogrammatic languages, like Chinese and Japanese.

10.10.1 Sorting and the asort Utility

Digital UNIX Version 4.0 supports the `asort` utility, an extension of the `sort` command, which allows characters of ideogrammatic languages, like Chinese and Japanese, to be sorted according to multiple collation sequences. For more information on the `asort` utility, see `asort`(1).

10.10.2 Multilingual EMACS

Digital UNIX Version 4.0 supports the Multilingual EMACS editor (MULE) for Asian languages. See `MULE`(1) for more information.

10.10.3 Mail and 8-Bit Support

Digital UNIX Version 4.0 provides support for ideogrammatic languages in `mailx`, `dtmail`, `MH`, and `comsat`.

For more information on these mail utilities, see the corresponding reference pages.

10.10.4 User-Defined Characters

Digital UNIX Version 4.0 provides support for creating user-defined characters (UDCs) for ideogrammatic languages, so that users can create and define character fonts and their attributes, including DECwindows fonts, with the `cgen` and `cedit` utilities. For more information on these utilities, see the appropriate reference pages.

Digital UNIX Version 4.0 also provides font rendering facilities so that X clients can use UDC databases through the X Server or font server to obtain bitmap fonts for user-defined characters.

For more information on user-defined characters, see *Writing Software for the International Market*.

10.11 Internationalization and Motif

Motif Version 1.2.3 takes advantage of many of the internationalization features of X11R6 and the C library to support locales. Motif Version 1.2.3 also supports the use of alternate input methods, which allows input of non-ISO Latin-1 keystrokes, and delivers an extensively rewritten `XmText` widget which supports multibyte and wide characters and on-the-spot input style.

Motif supports multibyte and wide characters through the use of the X multibyte functions, and the localized C run-time functions (such as `strlen`). In addition, the compound string routines have been modified to include the X11R6 `XFontSet` functionality to allow for the creation of localized strings.

The User Interface Language (UIL) supports the creation of localized UID files through the `-s` compile-time switch on the UIL compiler, which causes the compiler to construct localized strings.

Alternate input methods can be specified by a resource on the `VendorShell` widget. Widgets that are parented by a `Shell` class widget

can take advantage of this resource and register themselves as using a specific method for input.

The following sections discuss additional Motif internationalization functionality.

10.11.1 Internationalized Motif Widgets

The following lists contain the widgets in the Motif Toolkit and in the DECwindows Extensions to the Motif Toolkit that support local language characters I/O capabilities and local language message displays.

Note that the Motif UIL compiler has been extended to support local language characters in UIL files.

- Motif Toolkit
  - Command
  - FileSelectionBox
  - Label
  - MessageBox
  - SelectionBox
  - Text
  - TextField
- DECwindows Extensions
  - ColorMix
  - CSText
  - Help
  - Print
  - Structured Visual Navigation (SVN)

10.11.2 Internationalized Common Desktop Environment (CDE)

CDE becomes the default desktop for Digital UNIX V4.0. Digital UNIX provides internationalization support for the following CDE clients:

- Application Manager
- Calculator
- Calendar
- Create Action
- File Manager

- Front Panel
- Help Viewer
- Icon Editor
- Login Screen
- Message
- Mailer
- Print Manager
- Style Manager
- Terminal Emulator
- Trash Can

10.11.3 Internationalized DECwindows X Clients

Digital UNIX Version 4.0 provides internationalization support for the following DECwindows X clients:

- Console Log
- Bookreader
- Calendar
- Cardfiler
- CDA Viewer
- Differences
- Keycap
- LinkWorks Manager
- Mail
- Motif Window Manager
- Notepad
- Paint
- Pause Screen
- Print Screen
- Session Manager
- X Display Manager

Security 11

11.1 Overview

Digital UNIX Version 4.0, running enhanced security, is designed to meet or exceed the requirements of the C2 evaluation class of DoD 5200.28-STD *Trusted Computer System Evaluation Criteria* (TCSEC), also known as the *Orange Book*. The enhanced security features ship as optional subsets. After the security subsets are installed, you can configure an enhanced security kernel and access secure commands and utilities.

11.2 C2 Functionality and TCSEC

The following C2 requirements specified in the *Orange Book* are supported by Digital UNIX Version 4.0 running enhanced security:

- Audit
- Identification and authentication
- Object reuse
- Discretionary access controls
- System architecture
- Integrity
- Security testing
- *Security* guide

11.2.1 Audit

The following audit features are provided in Digital UNIX Version 4.0:

- A new `dxaudit` GUI (graphical user interface)
- Command line interfaces compatible with those provided in ULTRIX Version 4.0 and higher releases
- The ability to send audit logs to a remote host
- Fine-grained preselection of system events, application events, and site-definable events

- Fine-grained post-analysis of system events, application events, and site-definable events
- Link-time configurability of audit subsystem
- Per-user audit characteristics profile with enhanced Identification and Authorization (I&A)

The audit system is set up from the command line. Maintenance for the audit subsystem is done from the command line or with the `dxaudit` GUI.

Digital UNIX Version 4.0 intends to support the POSIX 1003.6 standard for audit when it is approved. The Digital implementation will also provide backward compatibility with the current audit interfaces. For more information, see the guide *Security*.

11.2.2 Identification and Authentication

Digital's Security Interface Architecture (SIA) allows a single set of identification and authentication (I&A) utilities to work in either the nontrusted system or the trusted (enhanced security) system. By using the `secsetup` command, you can configure your system to use either nontrusted or enhanced security commands.

The following I&A features are provided in Digital UNIX Version 4.0 running enhanced security:

- Password control
  - Configurable maximum password length is up to 80 characters.
  - Configurable password lifetimes. This includes an optional minimum interval between password changes.
  - A floating value of the minimum password length, based directly on the *Department of Defense Password Management Guideline (Green Book)* guidelines and the password lifetime.
  - Per user password generation flags, which include the ability to require a user to have a generated password.
  - Recording of who (besides the user) last changed the user's password.
- Login control
  - Recording of last terminal and time of the last successful login, and of the last unsuccessful login attempt.
  - Automatic account lockout after a specified number of consecutive bad access attempts. This feature can be overridden by root in case of system database corruption.

  - A per-terminal setting for delay between consecutive login attempts, and the maximum amount of time each attempt is allowed before being declared a failed attempt.
  - A per-terminal setting for maximum consecutive failed login attempts before locking any new accesses from that terminal.
- A notion of ownership for pseudoaccounts.
- A notion of whether the account is "retired" or "locked."
- Code for handling a remote host like a terminal, without confusing the issue of a pty versus a host. This is only set up to handle Internet hosts, and has no support for similar concepts that would be useful for Local Area Transport (LAT) and DECnet.
- A notion of system default values for the various I&A fields.
- A CDE-based GUI (`dxaccounts`) to perform many of the I&A administration tasks.
- New `edauth`, `convauth`, and `convuser` utilities to make the migration of accounts to the enhanced security level easier.

For more information, see the guide *Security*.

11.2.3 Object Reuse

Object reuse is a standard feature of Digital UNIX Version 4.0. Object reuse ensures that the physical storage (memory or disk space) assigned to shared objects or physical storage that is released prior to reassignment to another user, is cleared or scrubbed. Examples of object reuse are disk space that is released after a file is truncated or physical memory that is released prior to reassignment to another user to read.

11.2.4 Discretionary Access Controls

Discretionary access controls (DACs) are a standard feature of Digital UNIX Version 4.0. Discretionary access control provides the capability for users to define how the resources they create can be shared. The traditional UNIX permission bits provide this capability.

The Digital UNIX Version 4.0 system also provides optional access control lists (ACLs) to provide object protection at the individual user level.

Setting permissions, including ACLs, is discussed in the *Security* manual.

11.2.5 System Architecture

Digital UNIX Version 4.0 maintains a separate execution domain for the trusted computing base (TCB) components using hardware memory management to protect the TCB while it is executing. It maintains a kernel address space for the operating system, and maintains separate address spaces for each instance of an executing trusted (or untrusted) application process. Writable address space sharing between processes is controlled by discretionary access controls (DAC), with the default being to disallow sharing. Sharing of read-only address space sections (for example, shared libraries) can be disabled.

Digital UNIX Version 4.0 also protects the on-disk TCB components using discretionary access control. Attempted violations of the DAC protections can be audited so that remedial action can be taken by the system security officer.

In addition, the TCB is structured into well defined, largely independent modules.

Digital UNIX Version 4.0 is designed, developed, and maintained under a configuration management system that controls changes to the specifications, documentation, source code, object code, hardware, firmware, and test suites. Tools, which are also maintained under configuration control, are provided to control and automate the generation of new versions of the TCB from source code and to verify that the correct versions of the source have been incorporated into the new TCB version. The master copies of all material used to generate the TCB are protected from unauthorized modification or destruction.

11.2.6 Integrity

Digital UNIX Version 4.0 provides the capability to validate the correct operation of hardware, firmware, and software components of the TCB. The firmware includes power-on diagnostics and more extensive diagnostics that can optionally be enabled. The firmware itself resides in EEPROM and can be physically write-protected. It can also be compared against, or reloaded from, an off-line master copy. Digital's service engineers can run additional hardware diagnostics as well.

The firmware can require authorization to load any operating software other than the default or to execute privileged console monitor commands that examine or modify memory.

Once the operating system is loaded, system diagnostics can be run to validate the correct operation of the hardware and software. In addition, test suites are available to ensure the correct operation of the operating system software.

The following two tools can be run automatically to detect inconsistencies in the TCB software and databases:

- `fverify`

  The `fverify` command reads subset inventory records from standard input and verifies that the attributes for the files on the system match the attributes listed in the corresponding records. Missing files and inconsistencies in file size, checksum, user ID, group ID, permissions, and file type are reported.

- `authck`

  The `authck` program checks both the overall structure and internal field consistency of all components of the authentication database and reports all problems that it finds.

11.2.7 Enhanced Security Administration

The Digital UNIX Version 4.0 operating system provides system administrators with tools to improve the ease of use of administering system security.

11.2.7.1 Configuring System Security

System administrators can select the security level associated with their system. The default security level consists of object reuse and DAC; by running the `secsetup` command, system administrators can select enhanced security features. The audit subsystem and ACL subsystem are configurable at kernel link time, regardless of the security level of the system.

11.2.7.2 Windows-Based Administration Utilities

Three GUIs are provided to deal with the day-to-day security administration on the local machine. Based on OSF/Motif, the enhanced security version `dxaccounts` (Account Manager under the CDE-based system administration utilities) utility is used to create and enhanced user accounts, modify of system defaults, and the audit mask for users.

The `dxaudit` GUI controls the administration of the audit system and the generation of audit reports. Administrators have the flexibility to configure the audit subsystem without the requirement of installing additional enhanced security features.

The `dxdevices` GUI is used to configure secure devices.

The old `XSysAdmin` and `XIsso` interfaces are provided for compatibility and will be retired in a future release.

For more information, see the `dxaccounts`(8X), `dxaudit`(8X), and `dxdevices`(8X) reference pages.

11.3 Other Security Features

Digital UNIX Version 4.0 supports the some features not available in other OSF-based UNIX operating systems.

11.3.1 Security Integration Architecture

All security mechanisms that run on the Digital UNIX Version 4.0 operating system run under the Security Integration Architecture (SIA) layer. The SIA allows you to layer various local and distributed security authentication mechanisms onto Digital UNIX Version 4.0 with no modification to the security-sensitive Digital UNIX Version 4.0 commands, such as `login`, `su`, and `passwd`. The SIA isolates the security-sensitive commands from the specific security mechanisms, thus eliminating the need to modify them for each new security mechanism.

See the *Security* manual for further details.

11.3.2 Toggling Between Security Mechanisms

Through the use of a middle-layer interface, the Security Integration Architecture (SIA), Digital UNIX Version 4.0 allows use of the `secsetup` command to toggle back and forth between the secure and the nonsecure commands and utilities.

11.3.3 Network Information Service (NIS) Compatibility

Digital provides support for accessing NIS distributed databases while running enhanced security.

Users on a Digital UNIX Version 4.0 enhanced security system can, for example, use the `ypcat passwd` command to gather information about users on the network; however, the user's encrypted password in the NIS distributed password database is not the same as the encrypted password on the secure system which cannot be viewed by unprivileged users.

In addition, on a Digital UNIX Version 4.0 system running enhanced security, NIS can be used to distribute the enhanced security protected password database as well.

11.3.4 DECnet Interoperability

The SIA interface provides support for Digital's networking software, DECnet.

11.3.5 Distributed Computing Environment (DCE) Interoperability

Through the SIA, Digital UNIX Version 4.0, when configured for enhanced security, allows you to enter both your system password and your DCE password at login time. You do not have to log in to the Digital UNIX Version 4.0 secure system and then log in again to DCE.

11.3.6 Configuration and Setup Scripts

Digital UNIX Version 4.0 supports the `secsetup` configuration and setup script which allows you to select the security level you wish to run, permits you to toggle back and forth between secure and nonsecure commands and utilities, and configures security at boot time depending upon the value of the `SECURITY` variable in the `/etc/rc.config` file.

11.3.7 Graphical User Interfaces

Digital UNIX Version 4.0 provides the `dxaccounts`, `dxaudit`, and `dxdevices` utilities that permit the creation and modification of user accounts, modification of system defaults, and all of the audit interfaces and devices.

11.4 Performance

With all security options configured and running (including auditing), Digital UNIX Version 4.0 shows a performance degradation of only 3%. With auditing turned off, there is no measurable performance degradation. With enhanced security configured but not turned on, there is no performance degradation whatsoever.

Under normal usage, ACLs do not significantly degrade performance.

For more information on security, see the *Security* manual.

Installation and System Setup 12

12.1 Overview

Digital UNIX Version 4.0 supports full, custom, and cloned installations for new systems and an update installation that allows users who already have Version 3.2C, 3.2D-1, or 3.2D-2 of the operating system installed to update to Digital UNIX Version 4.0 without overwriting system files. Also, Digital UNIX Version 4.0 supports a variety of setup utilities that allow users to configure their systems quickly and with relative ease.

Both the installation and setup utilities use a graphical user interface (GUI) on systems that support graphics and a text-based interface on systems that do not.

Additionally, the CD-ROM used to install Digital UNIX Version 4.0 contains file systems that are laid out just as the software would be installed on the system. It has directly accessible `root`, `/usr`, and `/var` areas. This format makes almost every operating system command and utility available to the installation process because your system actually mounts the installation media and runs off of it during the installation. You therefore have access to a complete, albeit generic, Digital UNIX operating system during the installation itself. This means that the UNIX commands required for recovery procedures, such as restoring corrupt file systems, are readily available even if your operating system is not yet fully functional. Also, if you inadvertently delete a system file, you have easy access to the file on the CD-ROM. The RIS area from which you invoke an installation is laid out in the same format as the CD-ROM and provides the same advantages.

12.2 Installation

Digital UNIX Version 4.0 supports full, cloned, and update installations either from a CD-ROM or across the network from a Remote Installation Services (RIS) server. For more information on RIS, see the guide *Sharing Software on a Local Area Network*.

Note

The RIS software is available in the Server Extensions kit and requires a separate license and PAK to access. For more information, see Chapter 1 and the *Software Product Description* for Digital UNIX Version 4.0.

- Full Installation

  A full installation allows users to install Digital UNIX Version 4.0 on new systems and is divided into the following three procedures:

  – Default Installation

    A default installation installs all mandatory software subsets onto a single disk chosen by the user. Additional optional subsets can be installed later (if there is room on the disk) by using the `setld` command.

    The default installation is intended for those users who do not want to customize their disk partitions, install across multiple disks, and who want to get the operating system up and running quickly and easily.

  – Custom Installation

    A custom installation is thoroughly configurable, allowing users to select disks and partitions for the `root`, `usr`, and `var` file systems and for primary and secondary swap. In addition, the custom installation allows users to select from a list of optional software subsets, rather than automatically installing only the mandatory subsets.

    The Custom Installation is intended for users who install across different disks and who know which optional subsets they need to install.

  – Cloned Installation

    A cloned installation lets you duplicate the file system layout, file system type, and software subset selections from a similar type system that has already been installed with Digital UNIX Version 4.0. A cloned installation can only be performed using RIS. If your system is registered to a RIS environment and a configuration description file (CDF) is specified, the installation procedure retrieves the CDF and uses the system configuration information stored in the CDF to configure and install your system.

  For more information on full installations, see the *Installation Guide.*

- Update Installation

  The Update Installation allows users to update their systems to a new version of the operating system without overwriting customized system

files, user files, altering file systems, or destroying existing disk partitions.

Note

Your system must be running Digital UNIX Version 3.2C, 3.2D-1, or 3.2D-2 in order to update install to Digital UNIX Version 4.0.

The `installupdate` utility invokes the update installation and does the following:

- Updates existing software subsets to Digital UNIX Version 4.0 and installs new mandatory software subsets.
- Does not overwrite customized system files, like `/etc/passwd` and `/etc/fstab`.
- Merges files that have changed with the old files on your system

  If something new is delivered in a system file that you have customized, or if a system file has changed, the update installation attempts to merge your file with the new file. Although the update installation attempts to merge files automatically, it writes a log of those files that must be merged manually.
- Provides an array of log files documenting what it has done.

Once the update installation is complete, you can load additional subsets using the `setld` command.

For more information on the update installation, see the *Installation Guide* and the *Update Installation Quick Reference Card*.

12.3 System Setup

Once the Digital UNIX Version 4.0 software is installed, if you have graphics capabilities, you can use the SysMan `Configuration Checklist` to set up your system. The first time you log in as `root` after a system installation or the first time you log in to a factory installed software (FIS) system, the SysMan `Configuration Checklist` displays the SysMan applications that are available to set up your system for general use.

Note

When you are logged in as superuser or `root`, you can invoke the SysMan `Configuration Checklist` at any time by clicking on the `Configuration Checklist` icon in the `System_Administration` folder, or entering the following command on the UNIX command line:

```
# /usr/sbin/checklist
```

The following utilities are available from the SysMan `Configuration Checklist`:

- Network Configuration Application
- BIND Configuration Application
- NIS - Network Information Service
- NFS Configuration Application
- License Manager
- Account Manager
- Mail Configuration Application
- Disk Configuration Application
- LAT - Local Area Transport
- UUCP - UNIX-to-UNIX Copy System
- NTP - Network Time Protocol
- Printer Configuration Application
- Security (BSD/2)
- Security Auditing
- Prestoserver I/O Acceleration
- Update Administration Utility
- Graphical UI Selection Utility

Many of the SysMan `Configuration Checklist` utilities are also available in text-based interfaces that can be displayed on systems that only have character-cell displays. See Section 13.2.2 for more information.

For more information about system setup in general, see the *Installation Guide*, the *System Administration* guide , the *Network Configuration* guide , the *Software License Management* guide, and the `setup`(8) reference page.

System Administration 13

13.1 Overview

Digital UNIX Version 4.0 supports all the customary UNIX system administration utilities from OSF/1 Version 2.0, including `tar`, `dump/restore`, and `dd` for performing backups and restores; `df`, `du`, `scu`, `radisk`, and `disklabel` for managing disks and disk usage; `tunefs`, `newfs`, and `fsck` for managing file systems; `dbx` for performing kernel debugging; and `adduser` for creating user accounts. For more information on these utilities, see the *System Administration* guide and the appropriate reference page for each utility.

In addition to providing system administration utilities from the OSF, Digital UNIX adds a number of other useful Digital-specific utilities, such as the following:

- SysMan Tools
- Software Subset Management Utility `setld`
- Event Management Utility (DECevent)
- Analysis Tools with Object Modification
- Enhanced Kernel Debugging
- Dynamically Loadable Subsystems
- Dynamic System Configuration
- Dynamic Device Recognition
- Dataless Management Services
- Monitoring Performance History
- Bootable Tape

The following sections describe the Digital-specific utilities.

13.2 SysMan Tools

With the release of Digital UNIX Version 4.0, the SysMan Tools become the preferred system administration utilities for the operating system.

SysMan makes your job as a system administrator easier by providing you with a graphical user interface for each of your adminstration tasks, such as installation, configuration, daily administration, monitoring, kernel/process tuning, storage management, and more. These utilities can be accessed through the `System_Admin` folder in the CDE Application Manager.

While the SysMan Tools were designed to take advantage of the Common Desktop Environment (CDE), most of the utilities will work outside of CDE with other window or display managers. Users who are not running CDE can access the utilities individually by invoking them from the command line, provided the `DISPLAY` environment variable is properly set on their systems. For instance:

```
# netconfig
```

It should be noted that many of the SysMan Tools are also available in text-based interfaces that can be displayed on systems that only have character-cell displays. See Section 13.2.2 for more information.

13.2.1 SysMan Utilities

SysMan offers the following system administration utilities:

Note

The contents of each folder in SysMan can vary depending on which subsets you have installed.

- Configuration Checklist

  You can use these applications to set up your system for general use after Digital UNIX has been installed. See Section 12.3 for more information.

- Configuration Applications

  You can use these applications to perform:

  – Network Configuration with `netconfig`
  – BIND Configuration with `bindconfig`
  – NFS Configuration with `nfsconfig`
  – Mail Configuration with `mailconfig`
  – Print Configuration with `printconfig`
  – Disk Configuration with `diskconfig`

  - LAT Configuration with `latsetup`
  - NIS Configuration with `nissetup`
- Daily Administration Applications

  After a system has been configured, you can use these applications to perform routine administrative tasks:
  - Account Manager with `dxaccounts`
  - Archiver with `dxarchiver`
  - File Sharing with `dxfileshare`
  - Host Manager with `dxhosts`
  - License Manager with `dxlicenses`
  - Shutdown Manager with `dxshutdown`
  - System Information with `dxsysinfo`
  - Audit Manager with `dxaudit`
  - Power Management with `dxpower`
  - DHCP Configuration with `dhcpconf`
  - Display Window with `dxdw`
- Monitoring and Tuning Applications

  While a system is running, you can use these applications to monitor and tune its resources:
  - Kernel Tuner with `dxkerneltuner`
  - Process Tuner with `dxproctuner`
- Storage Management Applications

  While a system is running, you can use these applications to configure its file systems:
  - Bootable Tape with `btcreate`
  - Prestoserve with `dxpresto`
  - Logical Storage Manager with `dxlsm`
- Tools

  Use these applications to check the status of the system:
  - Network Statistics with `netstat`
  - Virtual Memory Statistics with `vmstat`
  - I/O Statistics with `iostat`
  - Who? with `who`

For more information on the SysMan Tools, please click on the `Welcome to SysMan` icon in the `System_Admin` folder.

13.2.2 Text-Based Interfaces

Text-based interfaces are provided for those users who prefer to use a non-graphical interface or cannot display a graphical interface because they do not have the necessary hardware.

For instance, a set of text-based, menu-driven interfaces can be accessed through the `setup` utility:

```
# /usr/sbin/setup

Use this menu to set up your system and network.  When you
select an item, you will be asked a series of questions.

For more information about the items on the menu and the
questions you must answer, see the System Administration
and Network Administration guides.


         1) Network Configuration Application
         2) BIND Configuration Application
         3) NIS - Network Information Service
         4) NFS Configuration Application
         5) License Manager
         6) Mail Configuration Application
         7) LAT - Local Area Transport
         8) UUCP - UNIX-to-UNIX Copy System
         9) NTP - Network Time Protocol
        10) Printer Configuration Application
        11) Security (BSD/C2)
        12) Security Auditing
        13) Prestoserve I/O Acceleration
        14) Update Administration Utility
        15) Graphical UI Selection Facility
        16) Exit
Enter the menu item number that you want:
```

The same menu-driven utilities can be accessed individually by invoking the necessary application from the command line with a `-ui menu` switch. For example:

```
# netconfig -ui menu

                        Main Menu

               1  Network Interfaces

               2  Daemons

               3  Configuration Files
```

```
                4  Static Routes

                5  IP Router

                0  Exit

Enter the number of your choice:
```

The text-based command line interfaces for each command can be accessed by invoking the necessary application from the command line with a `-ui cli` switch. The `n`s in the following example would be numbers indicating the hosts' IP addresses:

```
# netconfig -ui cli -host list

host1:
name                localhost
address             nnn.n.n.n

host2:
name                elmo
address             nn.nnn.nn.nnn

host3:
alias               presto
comment             Bind Nameserver
name                presto.federation.com
address             nn.nnn.nnn.n

host4:
alias               voyager
comment             System name
name                voyager.federation.com
address             nn.nnn.nnn.nnn

#
```

More information about these interfaces can be found in the reference pages for each utility that supports them.

13.3 The setld Utility

The `setld` utility allows system administrators to install, inventory, and delete software subsets that are formatted according to the guidelines set forth in the *Guide to Preparing Product Kits*. For example, a system administrator might use the `setld` utility to install optional subsets that were not installed during the full or update installation of the operating system.

Digital requires application programmers to use the Digital kitting process when packaging software subsets designed to be installed on Digital UNIX

systems and explains, in the *Guide to Preparing Product Kits* how to create kits that are compatible with the `setld` utility.

For more information on the `setld` utility, see the *Guide to Preparing Product Kits* the *Installation Guide* and the `setld`(8) reference page.

13.4 DECevent Event Management Utility

DECevent is an event-management utility for Digital UNIX that translates system event log files into formatted ASCII reports. DECevent supports both a command-line and a graphical user interface (GUI). Event report information can be filtered by event types, date, time, and event entry numbers. Event report formats can be selected from full disclosure to very brief information messages. The `-i` (include) and the `-x` (exclude) options provide a wide range of selection criteria to narrow down the focus of event searches.

The DECevent utility also offers an interactive command shell interface, accessible with the command `--int`, that recognizes the same commands used at the command line. From the interactive command shell users can customize, change, or save system settings.

DECevent uses the system event log file `/usr/adm/binary.errlog` as the default input file for event reporting, unless another file is specified

Unless the event log file privileges have been changed to allow all users to read the event log file, which is a rare practice, Digital UNIX users need superuser privileges to use DECevent.

For more information on DECevent, see the *DECevent Translation and Reporting Utility*.

13.5 Analysis Tools with Object Modification

The Analysis Tools with Object Modification (ATOM) Advanced Developer's Kit, an optional subset that ships with Digital UNIX, enables programmers to perform standard program and performance analysis (procedure tracing, instruction profiling, data address tracing). For more information on ATOM, see the PostScript documentation that ships in the `/usr/lib/atom/doc` directory and the `atom`(1) and `atomtools`(5) reference pages.

13.6 Enhanced Kernel Debugging

The `dbx` debugger, as it comes from the OSF, supports a read-only examination of a locally running kernel, as well as the debugging of kernel core files through the use of the `-k` switch.

Digital added the following two features to `dbx`:

- A `-remote` switch to enable the remote, breakpoint debugging of a running kernel across a serial line

  The protocol is multibyte, and cashing as well as a multithread extension are supported.

- A front-end to `dbx`, called `kdbx`, which supports not only the entire suite of `dbx` commands, but a C library API that allows programmers to write C programs to extract and format kernel data more easily than they can with just `dbx -k` or `dbx -remote`.

  The `kdbx` front-end ships with several ready-made extensions in the file `/usr/var/kdbx`.

For more information on kernel debugging, see the guide *Kernel Debugging*.

13.7 Dynamically Loadable Subsystems

Digital UNIX Version 4.0 introduces the ability to package, load, and manage kernel subsystems on Digital UNIX systems.

Instructions on how to write and package loadable device drivers so that they will install and execute on Digital UNIX Version 4.0 systems are discussed in the guide *Writing Device Drivers: Tutorial*. The *Programmer's Guide* explains how to write and package loadable kernel subsystems so that they too will install and execute on Digital UNIX Version 4.0 systems. The *Programmer's Guide* also discusses in some detail the framework that supports the dynamic configuration and tuning of kernel attributes.

You should refer to those guides for more specific information on how to write and package loadable drivers and kernel subsystems, as well as how to construct an attribute table.

13.8 Dynamic System Configuration

In an effort to simplify system tuning, Digital UNIX Version 4.0 allows you to change certain kernel attributes without having to edit the the system configuration file or the file `param.c`, and without having to rebuild and reboot a target kernel for the changes to take affect. Through the use of **attribute tables**, each kernel subsystem—whether a Digital UNIX kernel subsystem or one developed by a third-party vendor—can define kernel attributes that can be changed at run-time by using the `/sbin/sysconfig` command with the `-r` option (if the kernel attribute supports run-time reconfiguration), or at boot-time by adding or modifying entries in the kernel attribute database, `/etc/sysconfigtab` and rebooting.

For more information, see the *System Administration* guide and the *System Tuning and Performance Management* guide.

13.9 Dynamic Device Recognition

Dynamic Device Recognition is a framework for describing the operating parameters and characteristics of SCSI devices to the SCSI CAM I/O subsystem. You can use DDR to include new and changed SCSI devices into your environment without having to reboot the operating system.

Beginning with Digital UNIX Version 4.0, DDR is preferred over the current static method for recognizing SCSI devices, because DDR will not disrupt user services and processes as happens with static methods of device recognition.

13.10 Dataless Management Services

Digital UNIX Version 4.0 supports dataless management services (DMS) which allows the `root`, `/usr`, and `/var` partitions of a system to live on a DMS server and be served over the network to a DMS client. The `root` and `/var` partitions are unique to each DMS client, while `/usr` is shared. The DMS client swaps and dumps locally, and can mount staff areas locally using NFS.

DMS reduces disk needs and simplifies system administration, since administrators can administer and backup their DMS clients on the DMS server. The code was developed by Digital.

13.11 Monitoring Performance History

The Monitoring Performance History (MPH) utility gathers timely and accurate inofrmation on the reliability and availability of the Digital UNIX operating system and associated platforms.

The MPH utility was previously included on the Complementary Products CD-ROM. This utility is now included on the Digital UNIX Operating System CD-ROM.

For more information, see the *Installation Guide.*

13.12 Bootable tape

Digital UNIX Version 4.0 introduces the ability to create a standalone bootable tape of the operating system. You can boot from the bootable tape as easily as you can boot from a CD-ROM or a RIS area, but without the overhead of selecting or installing subsets. When you restore your system from the bootable tape, you must reconfigure your system using the System

Management applications.

See the `btcreate`(8) and `btextract`(8) reference pages for more information.

Conformance to Internet Host Requirements A

This appendix addresses the conformance of Digital UNIX to Internet host requirements as specified by Request for Comments (RFC) 1122: *Requirements for Internet Hosts – Communication Layers* and RFC 1123: *Requirements for Internet Hosts – Applications and Support*, and the RFCs that they reference. (RFCs 1122 and 1123 are referred to as Host Requirements RFCs throughout this appendix.)

Note

Although there are RFCs that specify internetworking protocols not referenced in the Host Requirements RFCs, the conformance of Digital UNIX to the requirements specified in those RFCs is beyond the scope of this appendix.

The Digital UNIX Software Product Description (SPD) contains additional technical information about the networking component of Digital UNIX and a complete list of RFCs implemented in Digital UNIX.

This appendix contains the following information:

- Background information that briefly describes what RFCs are and, in particular, describes the Host Requirements RFCs.
- Tables that list the RFCs that the Host Requirements RFCs reference and against which Digital UNIX was validated.

  Associated with each table is a pointer to information in the Host Requirements RFCs that discusses requirements for each Internet layer or protocol.
- A discussion of how Digital UNIX systems can be configured conditionally to comply to Internet Host Requirements when acting as a host.

A.1 Background

The Internet Architecture Board (IAB) issues specifications, and updates to the specifications, in the form of RFCs. RFCs describe protocols (as well as other information) of interest to the Internet community.

The Host Requirements RFCs are a statement of requirements for host system implementations of the Internet protocol suite, when these host systems are connected to the Internet. They do the following:

- Reference other RFCs and documents describing the current specifications for the Internet protocols
- State a set of requirements for each referenced protocol
- Clarify issues where the source document may be confusing
- Correct errors in the referenced documents
- Describe specific provisions overriding the original documents

The Host Requirements RFCs also indicate whether the requirements they discuss are **must**, **must not**, **should**, **should not**, or **may** level requirements. If an implementation complies with all **must** and **should** level requirements, it is considered unconditionally compliant. If an implementation complies with all **must** level requirements, but not necessarily all **should** requirements, it is considered conditionally compliant.

Note

This appendix describes Digital UNIX's conformance to **must** and **must not** level requirements only. Although Digital UNIX complies with the vast majority of **should**, **should not**, and **may** level requirements described in the Host Requirements RFCs, compliance with **should**, **should not**, and **may** level requirements is beyond the scope of this appendix.

RFCs are frequently issued or updated. When updates are issued, earlier versions of the RFC are rendered obsolete. The Host Requirements RFCs were issued in October 1989. Where RFCs referenced by the Host Requirements RFCs have been updated since October 1989, the RFC number of the updated version, as well as the number of the version it rendered obsolete, are noted.

The following two RFCs are of general importance to the Internet community because they contain information that has an impact on all implementations of all protocols:

- RFC 1780: *Internet Official Protocol Standards*

  Describes the state of standardization of protocols used in the Internet. It lists recent changes in protocols, and also indicates a status of required, recommended, elective, limited use, or not recommended for each protocol described.

Digital UNIX was validated against RFC 1600. RFC 1600 renders RFCs 1540, 1500, 1410, 1360, 1280, 1250, 1200, 1140, 1130, 1100, and 1083 obsolete.

- RFC 1780: *Assigned Numbers*

  Lists the assigned values of the parameters used in the various protocols, for example, IP codes, TCP port numbers, TELNET option codes, ARP hardware types, and terminal type names.

  The Host Requirements RFCs reference RFC 1010, an earlier version of this RFC. Digital UNIX was validated against RFC 1340. RFC 1340 renders RFC 1060, 1010, as well as many earlier RFCs, obsolete.

A.2 The Host Requirements RFCs (RFC 1122 and RFC 1123)

RFC 1122 covers requirements for communication protocols for the data link, internetworking, and transport layers of host Internet software. RFC 1123 covers requirements for the application and support protocols for Internet host software.

This section contains tables that list and briefly describe the RFCs that Digital UNIX was validated against and that are referenced in the Host Requirement RFCs. Additionally, Table A-8 and Table A-9 list the total **must/must not** level requirements explicitly stated by the Host Requirements RFCs.

Table A-1 lists the RFCs that RFC 1122 references for the link layer. For a discussion of link layer requirements, see Chapter 2 of RFC 1122.

Table A-1: Referenced RFCs for the Link Layer

Referenced RFC[a]	Description
RFC 1042: *Internet Protocol on IEEE 802 (IP-IEEE)*	Specifies a standard method of encapsulating the IP datagrams and ARP requests and replies on IEEE 802 Networks.
RFC 894: *Internet Protocol on Ethernet Networks (IP-E)*	Specifies a standard method of encapsulating IP datagrams on an Ethernet.
RFC 826: *Address Resolution Protocol (ARP)*	Presents a method for converting protocol addresses (IP addresses) to local network addresses (Ethernet addresses).

Table A-1: (continued)

Table Notes:

a. Digital UNIX is also validated against the following RFCs that are not referenced in RFC 1122:
   - RFC 1103: *Transmission of IP over FDDI (IP-FDDI)*
   - RFC 1055: *A Nonstandard for Transmission of IP Datagrams Over Serial Lines: SLIP*

Table A-2 lists the RFCs that RFC 1122 references for the Internet layer. For a discussion of Internet layer requirements, see Chapter 3 of RFC 1122.

Table A-2: Referenced RFCs for the Internet Layer

Referenced RFC	Description
RFC 1112: *Host Extensions for IP Multicasting (IGMP)*	Specifies the extensions required of a host implementation of the IP to support multicasting.
RFC 1009[a]: *Requirements for Internet Gateways*	Documents the requirements for IP routers connected to the Internet.
RFC 950: *Internet Standard Subnetting Procedures*	Discusses standards for subnet addressing within internet networks.
RFC 792: *Internet Control Message Protocol (ICMP)*	Describes a protocol for exchanging informational and error messages between hosts, or between gateways and hosts.
RFC 791: *Internet Protocol (IP)*	Specifies an unreliable connectionless protocol for the delivery of datagrams between systems.

Table Notes:

a. RFC 1009: *Requirements for Internet Gateways* references the Exterior Gateway Protocol (EGP) and the Routing Information Protocol (RIP). The EGP is described in RFC 904 and the RIP is described in RFC 1058.

Table A-3 lists the RFCs that RFC 1122 references for the transport layer. For a discussion of transport layer requirements for UDP and TCP, see Chapter 4 of RFC 1122.

Table A-3: Referenced RFCs for the Transport Layer

Referenced RFC	Description
RFC 793: *Transmission Control Protocol (TCP)*	Specifies a connection-oriented reliable protocol for the delivery of stream data between ports.
RFC 768: *User Datagram Protocol (UDP)*	Defines an unreliable connectionless protocol for the delivery of data between ports.

Table A-4 lists the RFCs that RFC 1123 references for the TELNET protocol. For a discussion of TELNET protocol requirements, see Chapter 3 of RFC 1123.

Table A-4: Referenced RFCs for the TELNET Protocol

Referenced RFC	Description
RFC 1184[a]: *Telnet Linemode Option*	Describes terminal character processing on the client side of a Telnet connection.
RFC 1091: *Telnet Terminal Type Option*	Specifies a standard for hosts on the Internet that exchange terminal type information within the Telnet protocol.
RFC 1080: *Telnet Remote Flow Control Option*	Specifies a standard for hosts on the Internet that use remote flow control within the Telnet protocol.
RFC 1079: *Telnet Terminal Speed Option*	Specifies a standard for hosts on the Internet that exchange terminal speed information within the Telnet protocol.
RFC 1073: *Telnet Window Size Option*	Describes Telnet options that allow a client to convey window size to a Telnet server.
RFC 861: *Telnet Extended Options List*	Describes an extended Telnet option that allows an additional 256 Telnet options.

Table A-4: (continued)

Referenced RFC	Description
RFC 860: *Telnet Timing Mark Option*	Provides a mechanism for a user or process at one end of a Telnet connection to be sure that previously transmitted data has been completely processed, printed, discarded, or otherwise disposed of.
RFC 859: *Telnet Status Option*	Allows one end of a Telnet connection to verify the current status of Telnet options (for example, echoing) as viewed by the other end of the connection.
RFC 858: *Telnet Suppress Go Ahead (SGA) Option*	Allows a full duplex connection between a full duplex terminal and a host optimized to handle full duplex terminals to suppress the transmission of GO AHEADS.
RFC 857: *Telnet Echo Option*	Allows the two ends of a Telnet connection to agree on NVT keyboard characters.
RFC 856: *Telnet Binary Option*	Provides the option of binary transmission in a natural way for Telnet connections to INTERPRET the characters transmitted over a Telnet connection as binary data.
RFC 855: *Telnet Option Specification*	Specifies a method of option code assignment and standards for documenting options for the Telnet protocol.
RFC 854: *Telnet Protocol Specification*	Provides a general, bidirectional, 8-bit byte-oriented communications facility whose primary goal is to allow a standard method of interfacing terminal devices and terminal-oriented processes to each other.
RFC 736: *Telnet SUPDUP Option*	Allows a host to provide SUPDUP service on the normal Telnet socket (27 octal) instead of 137 (octal) which is the normal SUPDUP ICP socket.
RFC 734: *SUPDUP Protocol*	Describes a highly efficient display Telnet protocol.

Table A-4: (continued)

Referenced RFC	Description
RFC 732: *Data Entry Terminal Option*	Describes the DET option to Telnet. The DET option uses five classes of subcommands: 1) to establish the requirements and capabilities of the application and the terminal, 2) to format the screen, and to control the 3) edit, 4) erasure, and 5) transmission functions.

Table Notes:

a. RFC 1184: *Telnet Linemode Option* renders RFC 1116: *Telnet Linemode Option* obsolete, and it references RFC 885: *TELNET End of Record Option*.

Table A-5 lists the RFCs that RFC 1123 references for the file transfer protocols FTP and TFTP. For a discussion of file transfer protocol requirements, see Chapter 4 of RFC 1123.

Table A-5: Referenced RFCs for the File Transfer Protocols

Protocol	Referenced RFC	Description
FTP	RFC 959: *File Transfer Protocol (FTP)*	Specifies a protocol whose objectives are: 1) to promote sharing of files (computer programs and/or data), 2) to encourage indirect or implicit (via programs) use of remote computers, 3) to shield a user from variations in file storage systems among hosts, and 4) to transfer data reliably and efficiently.
	RFC 678: *Standard File Formats*	Specifies standard formats for files and describes the expected final form for printed copies of such files.
TFTP	RFC 1350[a]: *The TFTP Protocol*	Describes a very simple protocol used to transfer files in which each nonterminal packet is acknowledged separately.

Table Notes:

a. RFC 1350: *The TFTP Protocol* renders RFC 783: *The TFTP Protocol* obsolete.

Table A-6 lists the RFCs that RFC 1123 references for the SMTP protocol. For a discussion of SMTP protocol requirements, see Chapter 5 of RFC 1123.

Table A-6: Referenced RFCs for the SMTP Protocol

Referenced RFC	Description
RFC 1049: *A Content-Type Field for Internet Messages*	Specifies additions to the *Internet Mail Protocol*, RFC-822, for the Internet community.
RFC 1047: *Duplicate Messages and SMTP*	Examines a synchronization problem in the SMTP that can cause a message to be delivered multiple times.
RFC 974: *Mail Routing and the Domain System*	Describes how mail systems on the Internet are expected to route messages based on information from the domain system described in RFCs 882, 883, and 973.
RFC 822: *Standard for the Format of ARPA Internet Text Messages*	Specifies a syntax for text messages that are sent among computer users within the framework of electronic mail.
RFC 821: *Simple Mail Transfer Protocol (SMTP)*	Describes a protocol designed to be independent of the particular transmission subsystem and that requires only a reliable ordered data stream channel to transfer mail reliably and efficiently.

Table A-7 lists the RFCs that RFC 1123 references for the Domain Name System, Host Initialization, and Remote Management support services. For a discussion of Domain Name System, Host Initialization, and Remote Management requirements, see Chapter 6 of RFC 1123.

Table A-7: Referenced RFCs for the Support Services

Service	Referenced RFC	Description
Domain Name System	RFC 1035: *Domain Names - Implementation and Specification*	Describes the details of the domain system and protocol. Assumes that the reader is familiar with the concepts discussed in a companion RFC, *Domain Names - Concepts and Facilities*.
	RFC 1034: *Domain Names - Concepts and Facilities*	Introduces the Domain Name System (DNS).
	RFC 974: *Mail Routing and the Domain System*	Describes how mail systems on the Internet are expected to route messages based on information from the domain system described in RFCs 882, 883, and 973.
Host Initialization	RFC 903: *A Reverse Address Resolution Protocol*	Describes a link-layer protocol that allows a host to find its IP address.
Network Management	RFC 1213[a]: *Management Information Base for Network Management of TCP/IP-based internets: MIB II*	Defines the second version of the Management Information Base (MIB-II) for use with network management protocols in TCP/IP-based Internets.
	RFC 1157[b]: *A Simple Network Management Protocol (SNMP)*	Defines a simple protocol by which management information for a network element can be inspected or altered by logically remote users.

Table Notes:

a. Digital UNIX is validated against RFC 1213: *Management Information Base for Network Management of TCP/IP-based Internets: MIB-II.*

   RFC 1123 references RFC 1066: *Management Information Base for Network Management of TCP/IP-based Internets.* RFC 1158 renders both RFC 1066 and RFC 1156: *Management Information Base for Network Management of TCP/IP-based Internets* obsolete. RFC 1213, in turn, renders RFC 1158 obsolete.

b. RFC 1157: *A Simple Network Management Protocol (SNMP)* supercedes RFC 1098: *A Simple Network Management Protocol (SNMP)*, which is referenced in RFC 1123.

Table A-8 summarizes the **must/must not** level requirements, per layer, explicitly stated by RFC 1122.

Table A-8: Total must/must not Requirements in RFC 1122

Link Layer	Internet Layer	Transport Layer
10	92	79
(7 must, 3 must not)	(80 must, 12 must not)	(74 must, 5 must not)

Table A-9 summarizes the **must/must not** level requirements explicitly stated by RFC 1123.

Table A-9: Total must/must not Requirements in RFC 1123

General Requirements	TELNET Protocol	File Transfer Protocols	SMTP Protocol	DNS Protocol
9	24	50	41	33
(9 must)	(22 must, 2 must not)	(46 must, 4 must not)	(32 must, 9 must not)	(29 must, 4 must not)

A.3 Configuring Digital UNIX to Conditionally Comply to the Host Requirements RFCs

The following sections describe how to configure your system to conditionally comply to the specifications described in the Host Requirements RFCs when your system is acting as an Internet host. Under each heading is a description of a **must** level item with which Digital UNIX does not comply by default. Along with each item is a discussion about why Digital UNIX does not comply, and information about how to configure your system to comply with that item.

A.3.1 Internet Layer[1] (RFC 1122)

When the IP datagram reassembly timeout expires, the partially reassembled datagram must be discarded and an ICMP Time Exceeded message sent to the source host, if fragment zero has been received (RFC 1122, Section 3.3.2).

Digital UNIX discards the partially reassembled datagram when the reassembly timeout expires. However, by default, Digital UNIX does not generate an ICMP Time Exceeded message.

At the time this host requirement was written, it was believed that a form of path MTU discovery procedure might find this message useful (RFC 1122, Section 3.2.2.4). RFC 1191: *Path MTU Discovery*, however, does not use this mechanism.

Receiving an ICMP Time Exceeded message may be useful for TCP connections, because TCP is required to act on receipt of ICMP error messages. UDP has no such requirement. While fragmentation is now generally prevented with TCP, this is not the case with UDP. Large UDP messages, for example those generated by NFS, can cause storms of ICMP Time Exceeded messages, if these messages were generated by default.

For these reasons, Digital does not recommend that Digital UNIX be configured to generate ICMP Time Exceeded messages. Digital UNIX records the number of fragments dropped due to reassembly timeout; you can run the `netstat -p` command to display this number. The following example shows the display for IP of the `netstat -p` command. Ten fragments were dropped due to reassembly timeout:

```
% /usr/sbin/netstat -p ip

ip:
        1831450 total packets received
        0 bad header checksums
        0 with size smaller than minimum
        0 with data size < data length
        0 with header length < data size
        0 with data length < header length
        3542 fragments received
        0 fragments dropped (dup or out of space)
        10 fragment dropped after timeout
        0 packets forwarded
        0 packets not forwardable
```

[1] Digital UNIX can be configured to comply with all **must/must not** level requirements for systems acting as Internet hosts.

The Internet Engineering Steering Group (IESG) recommended to the IAB (in September, 1991) that the "Requirements for Internet IP Routers" specify the routing protocol Open Shortest Path First (OSPF) as "MUST IMPLEMENT". Digital UNIX contains latent support for OSPF as part of Cornell University's `gated` daemon.

```
        0 redirects sent
```

A.3.1.1 Configuration Information

To configure the system to send ICMP Time Exceeded on Reassembly messages, set the kernel variable `ipsendreastimo` to 1. The default for this variable is zero (0).

To set the `ipsendreastimo` variable, become superuser and patch the kernel disk image using the `dbx patch` command as follows:

```
# dbx –k /vmunix
dbx version 11.0.1
Type 'help' for help.

stopped at  [thread_block:1403 ,0xfffffc000032d860] \
                               Source not available
(dbx) patch ipsendreastimo = 1
1
(dbx) quit
```

Reboot your system with the `shutdown -r` command to have the change take effect. For more information, see the `shutdown`(8) reference page.

A.3.2 Transmission Control Protocol (RFC 1122)

The Urgent Data pointer must point to the last octet of the sequence of urgent data (RFC 1122, Section 4.2.2.4).

RFC 793: *Transmission Control Protocol* contains conflicting statements about the octet that is referenced by the urgent pointer in a sequence of urgent TCP data. The first of these statements indicates that the urgent pointer "points to the sequence number of the octet following the urgent data."

RFC 1122, Section 4.2.2.4, however, indicates that the "urgent pointer points to the sequence number of the LAST octet (not LAST + 1) in a sequence of urgent data." This requirement reflects the second, and conflicting, definition of the urgent pointer as described in RFC 793.

BSD has traditionally applied the first definition of the urgent pointer that appears in RFC 793. To maximize interoperability, Digital UNIX uses the BSD default which means that the urgent pointer points to the sequence number of the LAST octet plus one in a sequence of urgent data. This behavior is controlled by the `tcp_urgent_42` kernel variable which applies system-wide and therefore affects all TCP connections.

A.3.2.1 Configuration Information

To configure the system to point to the last octet in a sequence of urgent data, set the kernel variable `tcp_urgent_42` to zero (0). The default for this variable is 1.

To set the `tcp_urgent_42` variable, become superuser and patch the kernel disk image using the `dbx patch` command as follows:

```
# dbx –k /vmunix
dbx version 11.0.1
Type 'help' for help.

stopped at [thread_block:1403 ,0xfffffc000032d860] \
                                    Source not available
(dbx) patch tcp_urgent_42 = 0
0
(dbx) quit
```

Reboot your system with the `shutdown -r` command to have the change take effect. For more information, see the `shutdown`(8) reference page.

Index

D

E

F

G

H

I

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

How to Order Additional Documentation

Technical Support

If you need help deciding which documentation best meets your needs, call 800-DIGITAL (800-344-4825) before placing your electronic, telephone, or direct mail order.

Electronic Orders

To place an order at the Electronic Store, dial 800-234-1998 using a 1200- or 2400-bps modem from anywhere in the USA, Canada, or Puerto Rico. If you need assistance using the Electronic Store, call 800-DIGITAL (800-344-4825).

Telephone and Direct Mail Orders

Your Location	Call	Contact
Continental USA, Alaska, or Hawaii	800-DIGITAL	Digital Equipment Corporation P.O. Box CS2008 Nashua, New Hampshire 03061
Puerto Rico	809-754-7575	Local Digital subsidiary
Canada	800-267-6215	Digital Equipment of Canada Attn: DECdirect Operations KAO2/2 P.O. Box 13000 100 Herzberg Road Kanata, Ontario, Canada K2K 2A6
International	————	Local Digital subsidiary or approved distributor
Internal[a]	————	SSB Order Processing – NQO/V19 *or* U. S. Software Supply Business Digital Equipment Corporation 10 Cotton Road Nashua, NH 03063-1260

[a] For internal orders, you must submit an Internal Software Order Form (EN-01740-07).

Reader's Comments

Digital UNIX
Technical Overview
AA-QTLLA-TE

Digital welcomes your comments and suggestions on this manual. Your input will help us to write documentation that meets your needs. Please send your suggestions using one of the following methods:

- This postage-paid form
- Internet electronic mail: `readers_comment@zk3.dec.com`
- Fax: (603) 881-0120, Attn: UEG Publications, ZKO3-3/Y32

If you are not using this form, please be sure you include the name of the document, the page number, and the product name and version.

Please rate this manual:	Excellent	Good	Fair	Poor
Accuracy (software works as manual says)				
Completeness (enough information)				
Clarity (easy to understand)				
Organization (structure of subject matter)				
Figures (useful)				
Examples (useful)				
Index (ability to find topic)				
Usability (ability to access information quickly)				

Please list errors you have found in this manual:

Page	Description

Additional comments or suggestions to improve this manual:

What version of the software described by this manual are you using?

Name/Title Dept.

Company Date

Mailing Address

Email Phone

Do Not Cut or Tear − Fold Here and Tape

digital™

NO POSTAGE
NECESSARY IF
MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL
FIRST−CLASS MAIL PERMIT NO. 33 MAYNARD MA

POSTAGE WILL BE PAID BY ADDRESSEE

DIGITAL EQUIPMENT CORPORATION
UEG PUBLICATIONS MANAGER
ZKO3−3/Y32
110 SPIT BROOK RD
NASHUA NH 03062−9987

Do Not Cut or Tear − Fold Here

Cut on
Dotted
Line