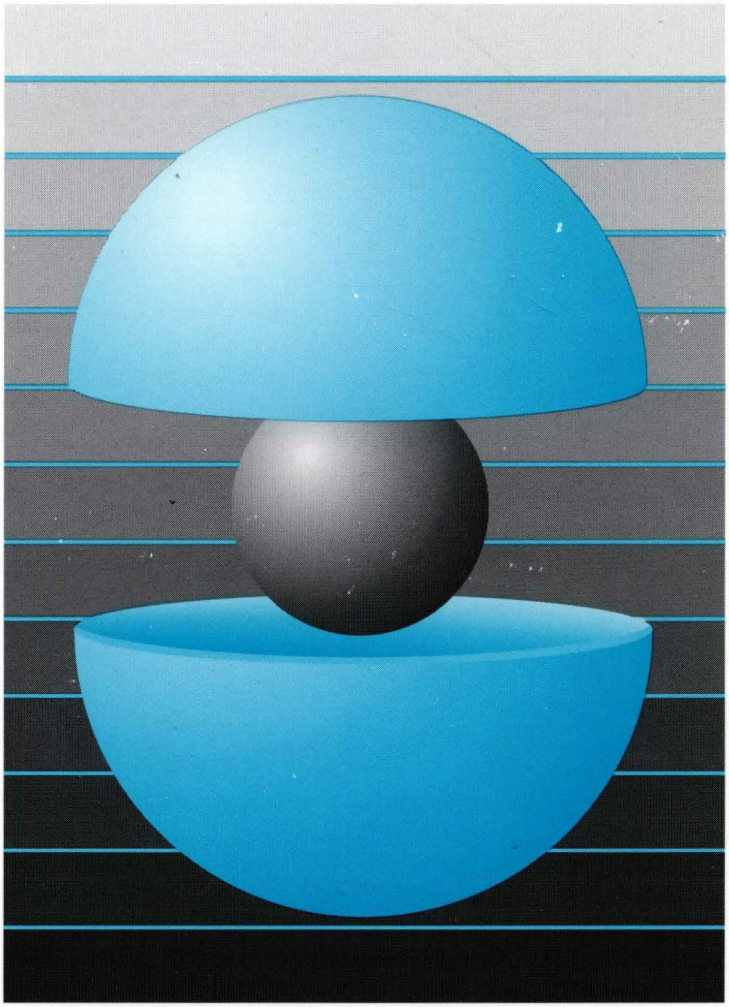
DEC OSF/1

digital

Technical Overview



Part Number: AA-Q0R1B-TE

DEC OSF/1

Technical Overview

Order Number: AA-Q0R1B-TE

August 1994

Product Version: DEC OSF/1 Version 3.0

This document describes the functionality in DEC OSF/1 Version 3.0.

Digital Equipment Corporation
Maynard, Massachusetts

Restricted Rights: Use, duplication, or disclosure by the U.S. Government is subject to restrictions as set forth in subparagraph (c) (1) (ii).

Digital Equipment Corporation makes no representations that the use of its products in the manner described in this publication will not infringe on existing or future patent rights, nor do the descriptions contained in this publication imply the granting of licenses to make, use, or sell equipment or software in accordance with the description.

Possession, use, or copying of the software described in this publication is authorized only pursuant to a valid written license from Digital or an authorized sublicensor.

© Digital Equipment Corporation 1994
All rights reserved.

The following are trademarks of Digital Equipment Corporation:

ALL–IN–1, Alpha AXP, AlphaGeneration, AXP, Bookreader, CDA, DDIS, DEC, DEC Ada, DEC Fortran, DEC FUSE, DECnet, DECstation, DECsystem, DECterm, DECUS, DECwindows, DTIF, MASSBUS, MicroVAX, OpenVMS, POLYCENTER, Q–bus, TURBOchannel, RRD42, ULTRIX, ULTRIX Mail Connection, ULTRIX Worksystem Software, UNIBUS, VAX, VAXstation, VMS, VR160, XUI, and the DIGITAL logo.

BSD is a trademark of Uunet Technologies. Prestoserve is a trademark of Legato Systems, Inc.; the trademark and software are licensed to Digital Equipment Corporation by Legato Systems, Inc. NFS is a registered trademark of Sun Microsystems, Inc. ONC is a trademark of Sun Microsystems, Inc. Open Software Foundation, OSF, OSF/1, OSF/Motif, and Motif are trademarks of the Open Software Foundation, Inc. Sun is a registered trademark of Sun Microsystems, Inc. UNIX is a registered trademark in the United States and other countries licensed exclusively through X/Open Company Ltd. X/Open is a trademark of X/Open Company Ltd.

All other trademarks and registered trademarks are the property of their respective holders.

Contents

About This Manual

1 Introduction

2 Symmetrical Multiprocessing

3 Networking

4 File System

5 Virtual Memory

6 I/O Subsystem

7 Development Environment

8 Windowing Environment

11 Security

12 Installation and System Setup

13 System Administration

About This Manual

This document provides a brief technical overview of the functionality in DEC OSF/1 Version 3.0, and provides a list of the various system limits of the operating system.

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 DEC OSF/1 Version 3.0 that Digital supports.

Audience

This manual is for anyone who is interested in the functionality in DEC OSF/1 Version 3.0.

Organization

This document contains the following chapters and appendixes:

Chapter 1	Introduction to DEC OSF/1 Version 3.0
Chapter 2	Symmetrical Multiprocessing
Chapter 3	Networking
Chapter 4	The File System Subsystem
Chapter 5	The Virtual Memory Subsystem
Chapter 6	The I/O Subsystem
Chapter 7	The Development Environment
Chapter 8	The Windowing Environment
Chapter 9	System V Functionality
Chapter 10	Internationalization
Chapter 11	Security
Chapter 12	Installation and System Setup
Chapter 13	System Administration

Appendix A Maximum System Limits for DEC OSF/1 Version 3.0

Appendix B Conformance to Internet Host Requirements

Related Documents

You should have access to the *Software Product Description* (SPD), the *Systems and Options Catalog*, and the entire DEC OSF/1 Version 3.0 documentation suite.

The printed version of the DEC OSF/1 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	Teal
System Administrators	S	Red
Network Administrators	N	Yellow
Programmers	P	Blue
Reference Page Users	R	Black

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* provides information on all of the books in the DEC OSF/1 documentation set.

Reader's Comments

Digital welcomes your comments on this or any other DEC OSF/1 manual. A Reader's Comment form is located in the back of each printed DEC OSF/1 manual and on line in the following location:

```
/usr/doc/readers_comment.txt
```

You can send your comments in the following ways:

- Internet electronic mail: `readers_comment@zk3.dec.com`

- Fax: 603-881-0120 Attn: UEG Publications, ZK03-3/Y32
- Mail:

  Digital Equipment Corporation
  UEG Publications Manager
  ZK03-3/Y32
  110 Spit Brook Road
  Nashua, NH 03062-9987

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

If you have suggestions for improving particular sections or find any errors, please indicate the manual title, order number, and section numbers. Digital also welcomes general comments.

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 DEC OSF/1 Version 3.0

DEC OSF/1 Version 3.0 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. DEC OSF/1 Version 3.0 also ships with Motif Version 1.1.3 to ensure backwards compatibility with applications that link against those libraries. In addition, DEC OSF/1 Version 3.0 supports the full features of the X Window System, Version 11, Release 5 (X11R5) from MIT.

The DEC OSF/1 Version 3.0 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 itself.

DEC OSF/1 Version 3.0 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 and optimized TCP/IP; and quickstarted shared libraries.

In addition, DEC OSF/1 Version 3.0 provides enhancements to greatly simplify system administration tasks, including an update installation that does not overwrite system files, support for loadable drivers and other kernel subsystems, support for dynamic system configuration, and a variety of setup scripts to tune and configure the system.

DEC OSF/1 Version 3.0 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.

To ensure a high level of compatibility with Digital's ULTRIX operating system, the DEC OSF/1 Version 3.0 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), DEC OSF/1 Version 3.0 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, DEC OSF/1 Version 3.0 is compliant with the OSF Application Environment Specification (AES) that specifies the interface to support these portable applications. In addition, the DEC OSF/1 Version 3.0 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 DEC OSF/1 Version 3.0 supports, see the SPD.

1.2 Packaging

DEC OSF/1 Version 3.0 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 DEC OSF/1 Version 3.0 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 DEC OSF/1 Version 3.0 documentation produced by Digital ships as Bookreader files on the DEC OSF/1 Version 3.0 CD-ROM and as ASCII reference pages accessible from the `man` command.

The following hardcopy documentation ships with the DEC OSF/1 Version 3.0 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 optionally available. For more information on optionally available documentation and the makeup of the documentation set itself, see the *Documentation Overview, Glossary, and Master Index.*

The remaining chapters and appendices discuss the following components of DEC OSF/1 Version 3.0:

- 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 and affords customers the opportunity to extend the life and increase the cost-effectiveness of their multiprocessor systems, since customers can now get additional compute power by adding CPU cards to their multiprocessors rather than having to buy more systems.

DEC OSF/1 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, DEC OSF/1 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 DEC OSF/1 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 DEC OSF/1 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.

DEC OSF/1 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 DEC OSF/1 development environment and the threads package that DEC OSF/1 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 DEC OSF/1 Version 3.0 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 DEC OSF/1 Version 3.0 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 DEC OSF/1 Version 3.0, 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 DEC OSF/1 Version 3.0 supports the following:

- Internet Protocol Suite
  - Application Protocols
    * Domain Name Protocol (DOMAIN)
    * Exterior Gateway Protocol (EGP)
    * 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)
- Supported Networks
  - 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)
- Network Administration Software
  - The entire suite of network commands and utilities from OSF Version 1.2
  - Ethernet packet filtering
  - SNMP Agent (POLYCENTER Common Agent)
  - Several popular packet filter applications in the public domain (`rarpd`, `tcpdump`, `tpslice`, `nfswatch`, `nfslogsum`)
  - The `screend` security policy daemon (developed at Digital)
  - UUCP from HoneyDanBer
  - Local Area Transport (LAT)

- 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 DEC OSF/1 Version 3.0.

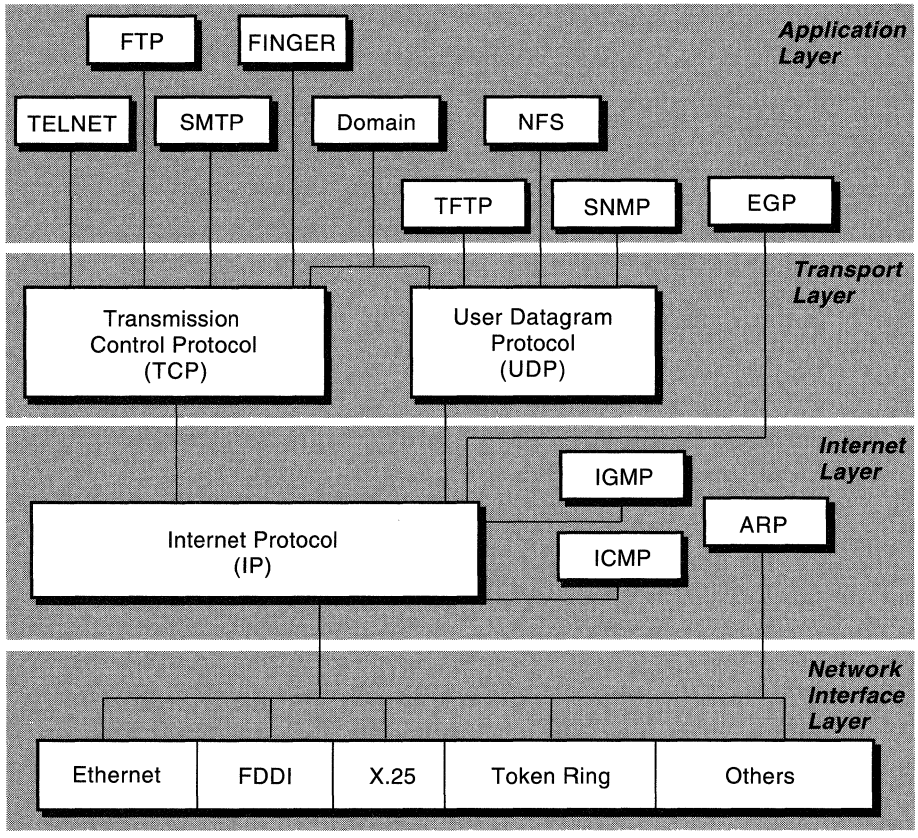
3.2 The Internet Protocol Suite

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, Exterior Gateway Protocol (EGP), 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



ZK-0819U-R

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 interface, 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, 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 DEC OSF/1 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 Exterior Gateway Protocol

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.

3.2.1.3 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 keeps the transport secure 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

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 reexecuted 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 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.

3.2.1.6 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.

3.2.1.7 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.

3.2.1.8 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.

3.2.1.9 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). The DEC OSF/1 operating system includes an SNMP agent that allows a host to be managed by a network manager.

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.

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 – DEC OSF/1 Version 3.0 supports IP Multicasting on a Local Area Network (LAN), as described in RFC 1112. 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) – DEC OSF/1 Version 3.0 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.

3.2.3.2 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) because SLIP has 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

DEC OSF/1 Version 3.0 supports interfacing to the following networks:

- Ethernet
- FDDI
- Token Ring

3.3.1 Ethernet

DEC OSF/1 Version 3.0 supports TURBOchannel, EISA, and XMI-based machines on 10 Mb/s Ethernet networks.

At the physical and IP levels, DEC OSF/1 Version 3.0 supports an Ethernet network with a Maximum Transfer Unit (MTU) of 1500 bytes at a maximum of 10 Mb/s.

In conformance with Ethernet standards, DEC OSF/1 Version 3.0 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/s, although this value can be decreased using the `ifconfig` command.

3.3.2 FDDI

DEC OSF/1 Version 3.0 supports TURBOchannel, Futurebus+, EISA, and XMI-based machines on 100 Mb/s FDDI networks in conformance with RFC 1042 and RFC 1188.

At the physical level, DEC OSF/1 Version 3.0 supports an FDDI network with a Maximum Transfer Unit (MTU) of 4500 bytes at a maximum of 100 Mb/s. At the IP level, the MTU is 4352 bytes at a maximum of 100 Mb/s.

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

3.3.3 Token Ring

DEC OSF/1 Version 3.0 supports TURBOchannel-based machines on 4 Mb/s and 16 Mb/s token ring networks and source routing in conformance with RFC 1042. 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, in compliance with RFC 1042, DEC OSF/1 Version 3.0 supports a token ring network with a Maximum Transfer Unit (MTU) of 4136 bytes at a maximum of 4 Mb/s and 8232 bytes at a maximum of 16 Mb/s. At the IP level, the MTU is 4092 bytes at a maximum of 4 Mb/s and 8188 bytes at a maximum of 16 Mb/s.

The default MTU at the IP level is always 4092 for both 4 and 16 Mb/s, 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 DEC OSF/1 Version 3.0:

- 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)

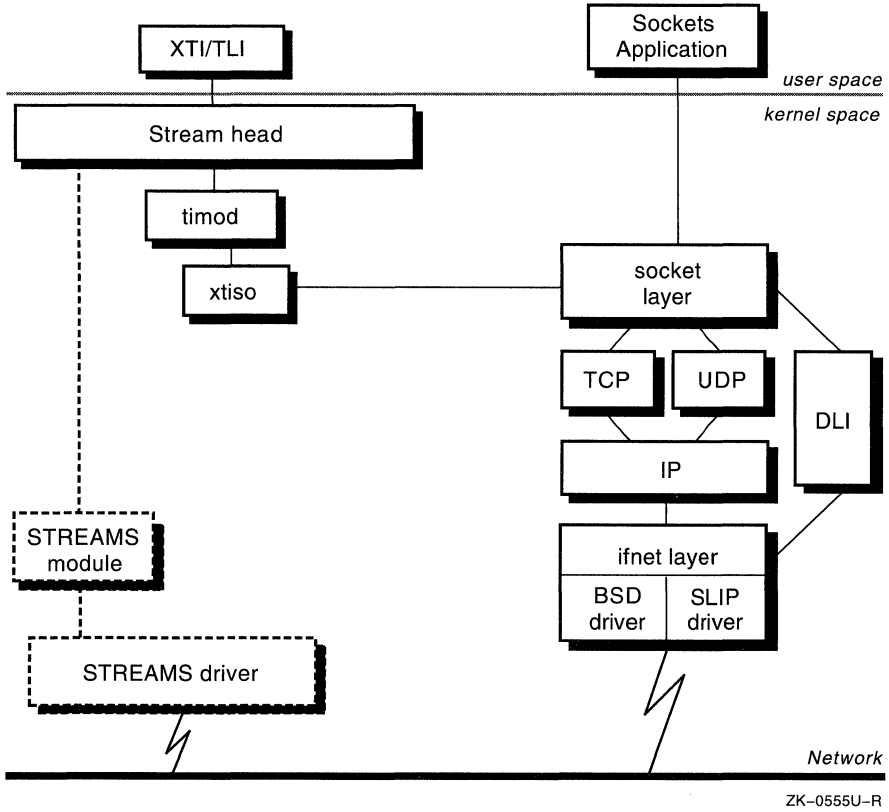
For more detailed information on the network programming environment, see the *Network Programmer's Guide.*

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.

Note

DEC OSF/1 Version 3.0 does not provide any STREAMS-based transport providers.

2. If a socket-based transport provider is specified (TCP or UDP), 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 de facto industry standard programming interface. DEC OSF/1 Version 3.0 implements the 4.3BSD socket interface as its default. 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* and the *Network Programmer's Guide.*

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`. DEC OSF/1 Version 3.0 supports System V Release 4.0 STREAMS from the OSF Version 1.2 code base, which provides support for the STREAMS `tty` interface (although DEC OSF/1 Version 3.0 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

DEC OSF/1 Version 3.0 provides the `ifnet` STREAMS module to allow programs using DEC OSF/1 Version 3.0'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 DEC OSF/1 Version 3.0.

3.4.5 Data Link Interface (DLI)

DLI is provided on DEC OSF/1 Version 3.0 as a backward compatibility feature to ULTRIX. DLI support on DEC OSF/1 Version 3.0 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 DEC OSF/1 Version 3.0. For more information, see the *Network Programmer's Guide.*

3.5 Network Administration Software

DEC OSF/1 Version 3.0 supports a variety of network administration software which is briefly described in the following sections.

3.5.1 Networking Commands and Utilities

DEC OSF/1 Version 3.0 supports the entire suite of networking commands from OSF Version 1.2, including: `bootpd`, `gated`, `finger`, `ftp`, `rdump`, `rdist`, `routed` and the complete suite of remote commands, `snmp`, `smtp`, `telnet`, and `tftp`. Additionally, DEC OSF/1 Version 3.0 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.

DEC OSF/1 Version 3.0 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 DEC OSF/1 Version 3.0 server, they can do so with the `rarpd` daemon.

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

  DEC OSF/1 Version 3.0 supports Version 2.2.1 of the `tcpdump` utility. This version of `tcpsump` 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/tpslice` – Log file tool

  The `tpslice` 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

  DEC OSF/1 Version 3.0 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 SNMP Agent

The SNMP Agent allows management of the Internet, FDDI, system resources, and network resources using the Simple Network Management Protocol (SNMP) and is comprised of a Base System Kit that ships with DEC OSF/1 Version 3.0 and a Software Development Kit that is available as a layered product and is orderable separately.

In previous releases of DEC OSF/1, Digital shipped an `snmpd` daemon and customers would then buy both the Base System Kit and the Software Development Kit as a layered product (the POLYCENTER Common Agent). In DEC OSF/1 Version 3.0, Digital is shipping the Base System Kit with the operating system as a mandatory subset which supports the following functionality:

- Full SNMP Version 1.0 agent capabilities
- Managed Object Modules (MOMs) for managing Internet MIB-2 objects, FDDI objects, and Token Ring objects on DEC OSF/1 Version 3.0

  Note that the ability to develop MOMs is supported only on the POLYCENTER Common Agent Software Development Kit which is available as a layered product.

3.5.4 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 allows the system manager to control which packets are forwarded 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.

3.5.5 UNIX-to-UNIX Copy Program

The UNIX-to-UNIX Copy Program (UUCP) program is actually a group of programs that support 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 itself 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 the *Network Configuration* guide.

3.5.6 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 previous releases of the operating system, the LAT driver was a clist-based implementation and was hard-coded to allow 620 users. In DEC OSF/1, the LAT driver is STREAMS-based and supports up to 1500 users, with a theoretical limit of 5000 users.

Note

In DEC OSF/1, 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 *Network Configuration* guide.

3.6 Naming Services

DEC OSF/1 Version 3.0 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 DEC OSF/1 Version 3.0 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. DEC OSF/1 Version 3.0 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.

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

DEC OSF/1 Version 3.0 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.

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 1129— *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

DEC OSF/1 Version 3.0 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). For more information on the UBC, see Chapter 5.

In addition, DEC OSF/1 Version 3.0 supports the Logical Volume Manager (LVM), the Logical Storage Manager (LSM), and the Prestoserve file system accelerator.

The following sections briefly discuss VFS, the file systems supported in DEC OSF/1 Version 3.0, the Logical Volume Manager, 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 uniformed interface abstracted from the file system layer which allows common access to files, no matter what file system the files reside on. 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 (ACLs) 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. For more information on XFAs, see `setproplist`(2). For more information on `pax`, see `pax`(1).

Information for File System Developers

In DEC OSF/1 Version 3.0, 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 DEC OSF/1, you do not need to make any changes to your `VOP_READDIR` interface routine for DEC OSF/1 Version 3.0, 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.

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 DEC OSF/1 Version 3.0 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 OSF/1; 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.

DEC OSF/1 Version 3.0 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 DEC OSF/1, SUN 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 GB 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
  - Guaranteed exclusive creation of files
  - 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

Since DEC OSF/1 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 DEC OSF/1 Version 3.0 client will use NFS Version 3 when it is served by a DEC OSF/1 Version 3.0 NFS server; however, when it is served by an NFS server running an earlier version of DEC OSF/1, the DEC OSF/1 Version 3.0 NFS client will use NFS Version 2.

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

4.4.2 Digital Enhancements to NFS

In addition to the NFS Version 3.0 functionality, DEC OSF/1 supports the following Digital enhancements to NFS:

- 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

DEC OSF/1 Version 3.0 supports the ISO-9660 CDFS standard for data interchange between multiple vendors; High Sierra Group standard for backwards 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, DEC OSF/1 Version 3.0 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

DEC OSF/1 Version 3.0 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 DEC OSF/1 Version 3.0 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, 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 especially suited to large-scale storage systems. The POLYCENTER Advanced File System provides faster crash recovery than the UNIX File System (UFS) which implements crash recovery using the `fsck` utility; ensures that file structures are recovered consistently; extends file and file-set sizes to greater than 2 GBs; creates, deletes, and renames files faster than UFS; and provides enhanced backup utilities (`dump` and `restore`). The POLYCENTER Advanced File System also supports Extended File Attributes (XFAs). For more information on XFAs, see Section 4.2.

In addition to the Advanced File System provided with the operating system, the POLYCENTER Advanced File System Utilities are available as a layered product. These utilities provide capabilities for adding or deleting volumes while the system is running, performing online backups, restoring deleted files, striping files, and improving performance.

The Advanced File System Utilities require a separate license PAK. Contact your Digital representative for additional information on the POLYCENTER Advanced File System Utilities product. For more information, see the *System Administration* guide and the *POLYCENTER Advanced File System Utilities Technical Summary.*

4.11 Logical Volume Manager

DEC OSF/1 Version 3.0 supports the Logical Volume Manager (LVM), a subsystem that expands and enhances the standard UNIX system mechanism for data storage, data retrieval, and data protection.

As a layer that fits between physical devices and file systems, LVM provides a virtual disk that enables you to store data without regard to the physical boundaries of individual disks. The virtual disk, known as a logical volume, looks and behaves like a single disk to the file system and to applications. LVM in DEC OSF/1 Version 3.0 supports the following functionality:

- Disk spanning

  Disk spanning allows you to combine multiple physical disk devices into one logical device. So, for example, you could "combine" two RZ26s and have them contain the `/usr` file system.

- Dynamic resizing

  Dynamic resizing allows you to add physical disks to an existing logical volume to increase its capacity.

For more information on LVM, see the *System Administration* guide. Note that LVM will be retired after the next release to be replaced by the Logical Storage Manager (LSM).

4.12 Logical Storage Manager

DEC OSF/1 Version 3.0 supports the Logical Storage Manager (LSM), a more robust logical storage manager than LVM, which it is designed to replace. Unlike LVM, LSM supports all of the following:

- Disk spanning

  Disk spanning allows you to combine multiple physical disk devices into one logical device. So, for example, you could "combine" two RZ26s and have them contain the `/usr` file system.

- Disk mirroring

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

- Disk striping

  Disk 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, allow you to perform backups and change the disk configuration without disrupting users while the system is up and running.

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 logical volumes that preserves all of the LVM data. After the migration is complete, administrators can add disk mirroring if they so desire.

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

Note that LSM 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*.

4.13 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 DEC OSF/1 Version 3.0 and once it is installed can be accessed with a Product Authorization Key (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 NFS 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 order to improve upon the Mach design adopted by the OSF. Specifically, Digital added the following functionality to improve performance and maintainability:

- Improved upon the lazy allocation policy
- Added an eager reservation policy
- Added a Unified Buffer Cache
- Added a round-robin swapping algorithm
- Added page in and page out clustering
- Added a memory-mapped device interface
- Rewrote the 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
- Added support for shared text segments
- Added support for page coloring
- Added a new kernel memory allocator
- Eliminated support for an external pager
- Added an improved memory reclamation policy
- Rewrote 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 DEC OSF/1 Version 3.0 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 manger 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 2% of physical memory rather than 25% in the OSF/1 implementation.

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/1 paging algorithm to support simultaneous paging to multiple swap partitions. By contrast, OSF/1 supports paging 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 AXP CPU, system administrators can greatly reduce this disparity in speed by spreading swap partitions out among different disks and different controllers. In fact, by supporting simultaneous paging to multiple swap partitions, DEC OSF/1 Version 3.0 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 added 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 DEC OSF/1 Version 3.0.

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. As a result, all processes that share the same text segment benefit from one process talking a page fault–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. DEC OSF/1 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 Kernel Memory Allocator

A new kernel memory allocator (kernel `malloc`) was added to DEC OSF/1 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.

5.13 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 DEC OSF/1 Version 3.0 memory management system makes it impractical at this time to provide a sensible interface for an external pager.

5.14 Improved Memory Reclamation Policy

A new memory reclamation policy was added in DEC OSF/1 Version 3.0 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). Thus, since 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 DEC OSF/1 Version 3.0, the page reclamation policy was changed to take advantage of the many unreferenced pages that are always in the 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 available 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.

5.15 Rewrote Swap Allocation Mechanism

In DEC OSF/1 Version 3.0, the swap allocation mechanism was rewritten to 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. The gain in performance can be as great as 50%.

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 DEC OSF/1 Version 3.0 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
DEC 2000 series processors	EISA Bus SCSI Bus
DEC 2100 series processors	PCI EISA SCSI
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

Note

DEC OSF/1 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 DEC OSF/1 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).

All of the device driver code is written by Digital and, with the exception of the drivers for the Digital proprietary XMI bus, 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 DEC OSF/1 Version 3.0. For more information on how to write device drivers for DEC OSF/1 Version 3.0, 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 DEC OSF/1 Version 3.0 supports, refer to the *Alpha AXP Systems Handbook*, the *Software Product Description* (SPD), and the *Systems and Options Catalog*.

The following sections briefly discuss the buses supported in DEC OSF/1 Version 3.0 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, DEC OSF/1 Version 3.0 supports the following buses:

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

6.2.1 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 DEC 2000

series processors. DEC OSF/1 Version 3.0 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/s, supports a 4 GB address space, 8 DMA channels, and is backwards compatible with the Industry Standard Architecture (ISA) bus.

Table 6-2 illustrates the frequency and transfer rate for various DEC 2000 series processors that support the EISA bus.

Table 6-2: EISA Bus Frequency and Transfer Rates

Processor	EISA Frequency	EISA Transfer Rate
DEC 2000 Model 300	8.33 MHz	33 MB/s
DEC 2000 Model 500	8.33 MHz	33 MB/s
DEC 2100 Model 500	8.33 MHz	33 MB/s

Table 6-3 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 DEC OSF/1 Version 3.0, see the SPD. Note that because of the open nature of the EISA bus, Digital does not control the development and availability of adapters.

Table 6-3: EISA Bus Adapters and Interconnects

Interconnect	Adapters
FDDI, Single Attachment Station	Digital Equipment Corporation (DEFEA-AA)
FDDI, Dual Attachment Stations	Digital Equipment Corporation (DEFEA-DA)
NI/Ethernet	Digital Equipment Corporation (DE422-SA)
SCSI-2	Adaptec AHA-1740A (high-performance)
SCSI-2, with FDI controller	Adaptec AHA-1742A (high-performance)
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
UTP, Single Attachment	Digital Equipment Corporation (DEFEA-UA)

Table 6-3: (continued)

Interconnect	Adapters
NVRAM	Digital Equipment Corporation (PB2SX-AA)
RAID	SWXCR-EX (mulitiport)
SVGA, 1024x768, 72Hz	Compaq Qvision 1024/E

For more specific information on supported EISA adapters and interconnects in DEC OSF/1 Version 3.0, see the SPD.

6.2.1.1 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). DEC OSF/1 Version 3.0 supports an EISA RAID controller (SWXCR) whose devices present 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 RAID controller supports RAID levels 0, 1, 5, 6, and 7, with levels 6 and 7 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)
- Level 7 (JBOD — Just a Bunch of Disks)

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

6.2.2 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/s; for the 128-bit bandwidth, 180 MB/s); 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 6-4 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 DEC OSF/1 Version 3.0, see the SPD. Note that because of the open nature of Futurebus+, Digital does not control the development and availability of adapters.

Table 6-4: 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 6-4, see the *Alpha AXP Systems Handbook* and the SPD.

6.2.3 PCI Bus

The Peripheral Componenet Interconnect Local Bus (PCI) is an open, high-performance 32-bit or 64-bit synchronous bus with multiplexed address and data lines, numerous compatible hardware implementations and, as Table 6-1 indicates, is supported on DEC 2100 series processors.

Table 6-5 illustrates the frequency and transfer rate the DEC 2100 series processors that support the PCI bus.

Table 6-5: PCI Bus Frequency and Transfer Rates

Processor	PCI Frequency	PCI Transfer Rate
DEC 2100 Model 500	33 MHz	264 MB/s

Table 6-6 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 DEC OSF/1 Version 3.0, 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 6-6: PCI Bus Adapters and Interconnects

Interconnect	Adapters
NI	DE435-AA

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

6.2.4 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.

DEC OSF/1 Version 3.0 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 DEC OSF/1 Version 3.0 support for SCSI. For more information on SCSI/CAM, see *Writing Device Drivers for the SCSI/CAM Architecture Interfaces.*

DEC OSF/1 Version 3.0 supports fast SCSI buses (maximum transfer rate of 40 MHz per data unit) and slow SCSI buses (maximum transfer rate of 25 MHz per data unit) which can be either wide (16 bits or 32 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/s device and a 10 MB/s device may share a fast narrow bus. When the 4 MB/s device is using the bus, the transfer rate is 4 MB/s. When the 10 MB/s device is using the bus, the transfer rate is 10 MB/s. 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 or KZTSA) 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
DEC 3000 Model 300 DEC 3000 Model 400 DEC 3000 Model 500	Slow/Narrow	5 MB/s
DEC 3000 Model 600 DEC 3000 Model 800	Fast/Narrow	10 MB/s
DEC 4000 series	Slow/Narrow Fast/Narrow	3.5 MB/s 5 MB/s (with an external connector) 10 MB/s (with no external connector)
DEC 7000/10000 series	Slow/Narrow Fast/Narrow	5 MB/s 10 MB/s

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

Table 6-8: SCSI Adapter Frequency and Transfer Rates

Processor	Bus Size	SCSI Transfer Rate
DEC 2000 series	Fast/Narrow	10 MB/s
DEC 2100 Model 500	Fast/Narrow	10 MB/s
DEC 3000 (all models)	Slow/Narrow Fast/Narrow Fast/Wide	5 MB/s 10 MB/s 10/20 MB/s

For more specific information on supported SCSI buses in DEC OSF/1 Version 3.0, see the SPD.

DEC OSF/1 Version 3.0 also supports the following functionality on SCSI buses:

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

6.2.4.1 Command Tagged Queueing

Command Tagged Queueing, supported on all DEC 3000 and DEC 4000 series processors, 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 also allows for improved "pipelining" of requests into the device.

6.2.4.2 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). DEC OSF/1 Version 3.0 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 7. 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

6.2.5 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/s. At 25 MHz, the peak data rate is 100 MB/s.

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 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	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 (SNS) KZTSA Fast Wide Differential (FWD)
FDDI	DEFTA-AA DEFZA-AA
Token Ring	DETRA
IPI	IPI-240T
NI	PMAD-AA
HX	PMAGB-BE
HX+	TBD
TX	PMAGB-JA
PXG+	PMAGB-DA (8 plane) PMAGB-EA (24-plane, 24-bit Z-buffer) PMAG-GB (24-plane upgrade) PMAG-HA (24-bit Z-buffer upgrade)
PXGT+	PMAG-FA (24-plane frame, 24-bit Z-buffer)
NVRAM	PMTNV-AA

For more specific information on supported TURBOchannel adapters and interconnects in DEC OSF/1 Version 3.0, see the SPD.

6.2.6 XMI Bus

The XMI bus is a 64-bit wide parallel bus that can sustain 100 MB/s bandwidth in a single processor configuration, and, as Table 6-1 indicates, is supported on the DEC 7000 and 10000 series processors. The bandwidth is exclusive of addressing overhead; the XMI bus can transmit 100 MB/s 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 DEC OSF/1 Version 3.0, see the SPD.

Table 6-11: XMI Adapters and Interconnects

Interconnect	Adapter
CI	CIXCD-AX
FDDI	DEMFA
SCSI	KZMSA
SDI/STI, DSSI, UQPORT	KDM70
NI	DEMNA

6.2.6.1 CI and KDM Controllers

The Computer Interconnect (CI) and KDM controllers, supported on the XMI bus of the DEC 7000/10000 series processors, interconnect multiple CPUs with various RA disks and TA tapes. The CI, through the CIXCD-AX adapter, allows DEC 7000/10000 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/s per channel and DEC OSF/1 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 7000/10000 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.

Development Environment 7

7.1 Overview

The development environment in DEC OSF/1 Version 3.0 is fully ANSI C compliant; offers the programming features of both BSD and System V UNIX; features debuggers that support C, Assembler, FORTRAN, 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 AXP architecture.

In addition, DEC OSF/1 Version 3.0 supports internationalization, standard UNIX development tools such as `awk`, `make`, `pixie`, 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 DEC OSF/1 Version 3.0 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.

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 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
- Supports the DEC C GEM-based compiler, accessible through the `-migrate` switch, which provides improved run-time performance.

For more information on the DEC OSF/1 compiler, see the `cc`(1) reference page.

7.3 Debuggers

DEC OSF/1 Version 3.0 supports the following two source code debuggers:

- dbx
- DECladebug

7.3.1 The dbx Debugger

The dbx debugger supports debugging programs written in C, FORTRAN, and Assembler. 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.

7.3.2 DECladebug Debugger

The DECladebug debugger is a source level, object-oriented symbolic debugger developed by Digital that supports the following features:

- Debugging programs written in C++, C, Fortran 77, Fortran 90, Ada, and Cobol
- Debugging machine level code
- Debugging running programs or core dumps
- Debugging multi-threaded applications
- Debugging Shared Objects
- Evaluating expressions using the syntax of the source programming language
- Remote debugging of programs running on the EB64, EB64+, and EB66 AXP evaluation boards from Digital's Semiconductor Engineering Group

  The evaluation boards are available through the technical OEM sales channel which supports module and chip level design business.

  The DECladebug remote debugging protocol is also available along with the C source code for a sample remote debugging server that adheres to the protocol. For more information, contact your Digital representative.

Whereas the DECladebug command-line interface is very similar to dbx, the DEC FUSE layered product, available separately, provides support for a graphical user interface.

DECladebug is a full C++ debugger which demangles C++ names and understands C++ expressions. As a result, DECladebug allows you to display individual object members and reference class members, hidden base class members, inherited members, members hidden by current scope, and global variables.

You can resolve overloaded function names by entering the function name with its full type signature, or by selecting the desired function from a menu that appears whenever you ambiguously specify an overloaded function.

The debugger displays the class members on instances of a derived class and the members inherited from a base class. When the optional verbose debugging mode is used, the debugger also displays virtual base class pointers for derived classes.

The DECladebug debugger supports stepping through and setting breakpoints in inline functions, all instances of a base class virtual function, as well as specific instances of virtual functions and templates. C++ exceptions are also supported.

The debugger itself was constructed using object-oriented techniques and is written in C++. DECladebug's foundation is a class library of reusable debugger components.

The DECladebug architecture is flexible and extensible, and new features, such as support for new languages, can be added easily to meet new customer requirements.

The debugger consists of the following three internal layers:

- The command-line interface

  The command-line interface allows the user to interact with DECladebug. The commands manipulate objects in the debugger state layer by using the public member functions provided by those objects. All manipulation of the user program is done using the facilities of the library.

- The debugger state

  The debugger state contains information describing the current debugging session. This state information is constructed from the principal objects exported by the class library such as programs, processes, address spaces, threads, and symbol tables.

- The DECladebug class library

  The DECladebug class library provides a rich set of primitives that can be used to construct applications that manipulate programs. The DECladebug debugger is itself an example of such an application. Applications are created by using classes from the library and by extending the class library to create new derived objects.

7.4 Shared Libraries

DEC OSF/1 Version 3.0 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.

DEC OSF/1 Version 3.0 supports the shared libraries described in Table 7-1.

Table 7-1: DEC OSF/1 Version 3.0 Shared Libraries

Library	Description
/usr/shlib	
libDXm.so	Digital Motif Extensions library
libDXterm.so	DECterm widget library, used by dxterm
libMrm.so	Motif Resource Manager library
libX11.so	Xlib 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)
libXimp.so	X Image display services library
libXm.so	Motif Widgets library
libXmu.so	X Miscellaneous utilities run-time library
libXt.so	X Intrinsics library
libXv.so	X video extension client-side run-time library
libaud.so	C2 security auditing library
libbkr.so	Motif Help System library
libc.so	C library
libc_r.so	Threads version of libc
libcda.so	CDA run-time library
libcdrom.so	Rock Ridge extensions to CDFS library
libchf.so	CDA/Imaging signal handling routines
libcmalib.so	CMA threads library
libcurses.so	Curses screen control library
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
libiconv.so	Internationalization codeset conversion routines
libids.so	Image display services library
libids_nox.so	Image display services not dependent on X

Table 7-1: (continued)

Library	Description
libimg.so	Image processing routines
libips.so	Image processing routines
liblwkdxm.so	LinkWorks run-time library
libm.so	Digital Portable Mathmatics library
libmach.so	Mach library
libmxr.so	Library used by mxr, the Ultrix binary interpreter for OSF/1
libots.so	Compiler run-time support
libproplist.so	VFS Extended File Attributes library
libpsres.so	Adobe Display PostScript resource utilities
libpthreads.so	DECthreads library
libsecurity.so	C2 security library
libsys5.so	System V compatibility library
libtli.so	XTI library
libxti.so	XTI library
/usr/shlib/X11	
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_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_triton.so	Device support routines for the QVision adapter
lib_dec_tx.so	Device support for the TX graphic adapter
lib_dec_vga.so	Common routines for the supported SuperVGA devices
lib_dec_ws.so	Low-layer operating system interface for the X server
lib_dec_xi_pcm.so	Dynamically-loadable X Input Extension library that supports the dial and box
lib_dec_xie.so	X Imaging Extension V3 library
lib_dec_xtrapext.so	X Trap Extension library

Table 7-1: (continued)

Library	Description
`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
`libdix.so`	Device-independent portion of the X Server
`libdixie.so`	With `libmixie.so`, supports the X Image Extensions (XIE)
`libext.so`	Minor extension support 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
`libxv.so`	X Video Extension library

Note

DEC OSF/1 Version 3.0 also provides static versions of these libraries.

7.4.1 Quickstart

DEC OSF/1 Version 3.0 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.

7.4.2 Dynamic Loader

DEC OSF/1 Version 3.0 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

7.4.3 Versioning

DEC OSF/1 Version 3.0 supports full and partial duplication of shared libraries. The loader looks for backwards 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, DEC OSF/1 Version 3.0 supports both Motif 1.1.3 and Motif 1.2.3 shared libraries in DEC OSF/1 Version 3.0 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*.

7.5 Run-Time Libraries

DEC OSF/1 Version 3.0 supports the following run-time libraries

- DEC Ada Run-Time Libraries (RTL)

  The DEC Ada run-time libraries (`libada.a` and `libada.so`) enable 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 libraries (`libpas.a`, `libpas.so`, and `libpas_msg.cat`) enable 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.

7.6 Development Commands

DEC OSF/1 Version 3.0 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 DEC OSF/1 Version 3.0 supports both the OSF/1 `make` command and the ULTRIX version of `make`, since the ULTRIX `make` command is POSIX 1003.2 compliant and more robust.

7.7 DECthreads

DECthreads is a library of routines, built on the basic Mach threads capabilities of OSF/1, that support the development of multithreaded applications on DEC OSF/1 Version 3.0. While DECthreads is an implementation of draft 4 of the proposed POSIX 1003.4a specification, it also provides a proprietary API to aid in porting applications from other

Digital platforms such as OpenVMS. 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 DEC OSF/1 kernel, providing SMP capabilities for multithreaded applications and realtime scheduling policies and priorities for multithreaded realtime applications.

7.8 Memory-Mapped File Support (mmap)

DEC OSF/1 Version 3.0 supports the Berkeley `mmap` function and therefore allows an application to access data files with memory operations rather than file I/O operations.

7.9 Realtime

DEC OSF/1 Version 3.0 supports a realtime user and programming environment, developed by Digital and shipped as a series of optional realtime subsets. The DEC OSF/1 Version 3.0 realtime programming environment provides many of the interfaces required by the POSIX 1003.1b-1993 standard for realtime (formerly 1003.4) which allow 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.

Unlike previous releases of the operating system, the DEC OSF/1 Version 3.0 kernel has realtime capability built into it, so that realtime does not have to be configured separately. However, you must explicitly enable realtime features when you configure and build your target kernel by uncommenting realtime kernel options in your system's target configuration file. These realtime kernel options make it possible for the operating system to guarantee that your application has access to resources in a timely and predictable manner.

If you configure your kernel to use the realtime preemption features, 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 DEC OSF/1 Version 3.0 realtime programming environment supports the following POSIX 1003.1b functionality:

- Realtime clocks and timers
- Fixed priority scheduling policies
- Kernel-mode process preemption

- Semaphores
- Process communication facilities
- Shared memory
- Process memory locking
- Asynchronous 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

DEC OSF/1 Version 3.0 supports a full-featured implementation of the X Consortium's X Window System, Version 11, Release 5 (X11R5) up to and including public patch 26, as well as the complete Motif Toolkit from OSF/1 Version 1.2.3. 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 Consortium and the OSF.

The DEC OSF/1 Version 3.0 implementation of X11R5 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 Chapter 7.

The following sections briefly discuss the X11R5 and Motif components of the DEC OSF/1 Version 3.0 windowing environment.

8.2 X Window System

The X11R5 windowing software consists of the following components:

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

8.2.1 X Client Libraries

DEC OSF/1 Version 3.0 supports the complete set of X11R5 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.*

8.2.2 X Server

Through the extensive use of shared libraries, DEC OSF/1 Version 3.0 supports a single X11R5 X server image for all graphic options. The DEC OSF/1 Version 3.0 X server loads only those server components required by a specific system configuration.

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

8.2.2.1 Multihead Graphic Support

Multihead graphic support is transparent in DEC OSF/1 Version 3.0, provided the proper option cards are installed and the additional graphic adapters are built into the kernel.

Table 8-1 lists the number of graphic heads supported by DEC OSF/1 Version 3.0.

Table 8-1: Multihead Graphic Support

Processor	Supported Graphic Heads
DEC 2000 series processors	
Model 300	4
Model 500	4
DEC 3000 series processors	
Model 300/300X	3
Model 300L/300LX	1
Model 400	3
Model 500	7

Table 8-1: (continued)

Processor	Supported Graphic Heads
Model 500X	6
Model 600	3
Model 800	6

For more information on the graphic options supported in DEC OSF/1 Version 3.0, see the *Systems and Options Catalog*.

8.2.2.2 X Server Extensions

DEC OSF/1 Version 3.0 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.

- Access-X

  Helps people with physical disabilities to interact with their windowing systems by making single keyboard operations extremely powerful, eliminating the need for multiple-key commands and mouse clicks.

  The code for Access-X was developed at Digital.

- DPS (Display PostScript Extension)

  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.

  DEC OSF/1 Version 3.0 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 previous releases of DEC OSF/1, support for input devices was built into the X server, and it was impossible for anyone to write drivers for third-party input devices unless they had access to the source code.

  Sample code showing how such a driver should be written is included in the `/usr/examples` directory. X-Input was developed by Digital.

- XKME (X Keyboard Management Extension)

  Internal extension for better support of international X clients.

- 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.

8.2.3 Display Manager

DEC OSF/1 Version 3.0 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 DEC OSF/1 Version 3.0 environment. For more information on using `Xdm`, see the *System Administration* guide and the `xdm`(1X) reference page.

8.2.3.1 xmodmap Keymap Format

The keymaps supplied with DEC OSF/1 Version 3.0 use the `xmodmap` keymap format, the de facto industry standard. Unlike the format of the keymaps supplied in earlier versions of DEC OSF/1, which was difficult to read and edit because it was written 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.

8.2.3.2 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 DEC OSF/1, the keys were in the `MIT-MAGIC-COOKIE-1` format and were not encrypted.

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

8.2.4 Font Server

DEC OSF/1 Version 3.0 supports a standard scalable font server that supplies a network of systems with access to fonts resident on any DEC OSF/1 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.

8.2.4.1 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.

DEC OSF/1 Version 3.0 supports loadable font renderers, so that anyone who adheres to the X11R5 standard can write their own font renderer for their own set of fonts and install them on a DEC OSF/1 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.

8.2.5 X Clients

DEC OSF/1 Version 3.0 supports the entire suite of X clients that ship with X11R5, including appres, atobm, bdftopcf, bitmap, bmtoa, editres, fs, fsinfo, fslsfonts, fstobdf, getcons, ico, imake, listres, lndir, mkfontdir, oclock, pswrap, 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,

xkill, xload, xlogo, xlsatoms, xlsclients, xlsfonts, xmag, xman, xmbind, xmh, xmkmf, xmodmap, xon, xpr, xprop, xrdb, xrefresh, xset, xsetroot, xstdcmap, xterm, xwd, xwininfo, and xwud.

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

8.3 Motif

DEC OSF/1 Version 3.0 supports the entire suite of OSF Version 1.2 Motif 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 (demos).

In Motif Version 1.2.3, 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, DEC OSF/1 Version 3.0 provides the Motif Version 1.1.3 libraries through its versioning functionality to allow Motif 1.1.3 applications to compile and run. 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.

8.3.1 Digital Extended Widget Set

In addition to the Xm widget library, DEC OSF/1 Version 3.0 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

8.3.2 Digital X Clients

In addition to the entire suite of X clients that ship with X11R5, DEC OSF/1 Version 3.0 supports a variety of Digital X clients including `dxbook`, `dxcalc`, `dxcalendar`, `dxcardfiler`, `dxclock`, `dxconsole`, `dxdiff`, `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

DEC OSF/1 Version 3.0 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 DEC OSF/1 Version 3.0 libraries to contain System V functionality, allows DEC OSF/1 Version 3.0 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 DEC OSF/1 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 DEC OSF/1 Version 3.0 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 DEC OSF/1 Version 3.0 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

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 DEC OSF/1 "look and feel." However, through manipulating each user's `.profile` file, each environment, System V and DEC OSF/1, 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 DEC OSF/1 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 DEC OSF/1 V 3.0 User's Guide* and the *SVR4 to SVE for DEC OSF/1 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 corresponds to a particular geographic/linguistic area, like China or France, and supports the language as it is used in that area. So by selecting a Chinese locale, for example, all commands, system messages, and keystrokes will be in Chinese characters and displayed in a way idiomatic to native Chinese.

DEC OSF/1 Version 3.0 is an internationalized operating system that not only allows users to interact with DEC OSF/1 Version 3.0 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 DEC OSF/1 Version 3.0 is based on POSIX 1003.2 and XPG4 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 support Asian languages have been included in the base system. In addition, DEC OSF/1 Version 3.0 supports the X Input Method (XIM) to facilitate input of local language characters, text drawing, measurement, and inter-client communication which is implemented according to the X11R5 specification.

Note that DEC OSF/1 Version 3.0 also supports a 32 bit `w_char` datatype for Asian languages, 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 DEC OSF/1 Version 3.0 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 DEC OSF/1 Version 3.0, 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
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
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
Finnish	fi_FI.ISO8859-1
German	de_DE.ISO8859-1
Swiss German	de_CH.ISO8859-1

Table 10-1: (continued)

Language	Locale Name
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.SJIS ja_JP.deckanji ja_JP.eucJP ja_JP.sdeckanji
Korean *	ko_KR.deckorean 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
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.*

The following sections briefly discuss additional internationalization functionality.

10.3 Internationalized Terminal Subsystem

DEC OSF/1 Version 3.0 extends the base `tty` terminal driver subsystem to include additional line disciplines for processing data in all supported languages. The line discipline used 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
- Software on-demand-loading for user-defined characters
- Code conversion between terminal codeset and application codeset

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

10.4 Codeset Conversion and the iconv Utility

DEC OSF/1 Version 3.0 supports the `iconv` utility, which converts text files from one locale's language to another, thereby assisting programmers in the writing of internationalized applications.

Code conversion is also implemented in the terminal driver and printing subsystem to allow the use of terminals and printers with different codesets; in mail utilities for mail interchange with systems using different codesets; and in the X Window Toolkit for text input, drawing, and interclient communication. For more information on the `iconv` utility, see the `iconv`(1) reference page.

10.5 Printing

DEC OSF/1 Version 3.0 supports the printing of ASCII 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 fonts are typically very large, DEC OSF/1 Version 3.0 makes use of a unique font faulting technology which substantially reduces memory requirements on the supported PostScript printers. For more information, on printing, see *Writing Software for the International Market.*

10.6 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.

10.7 Special Support for Ideogrammatic Languages

The following sections discuss special support in DEC OSF/1 Version 3.0 for ideogrammatic languages, like Chinese and Japanese.

10.7.1 Sorting and the `asort` Utility

DEC OSF/1 Version 3.0 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 the `asort`(1) reference page.

10.7.2 Mail and 8-Bit Support

DEC OSF/1 Version 3.0 provides support for ideogrammatic languages in `mailx`, `MH`, and `comsat`.

Note that when one or more language variants are installed, 8-bit mail is enabled.

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

10.7.3 User-Defined Characters

DEC OSF/1 Version 3.0 provides support for creating User-Defined Characters (UDC) 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.

For more information on internationalization, see *Writing Software for the International Market.*

10.8 Internationalization and Motif

Motif Version 1.2.3 takes advantage of the X11R5 internationalization features and relies upon the X11R5 and C libraries 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.

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 X11R5 `XFontSet` functionality and 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.

10.8.1 Internationalized Motif Widgets

Table 10-2 lists 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.

Table 10-2: Internationalized Motif Widgets

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

10.8.2 Internationalized DECwindows X Clients

DEC OSF/1 Version 3.0 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

DEC OSF/1 Version 3.0 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 called the *Orange Book.* The C2 security features ship as optional subsets. After the security subsets are installed, you can configure a C2 kernel and access secure commands and utilities.

The C2 security code came from the OSF and was enhanced by Digital.

11.2 C2 Functionality and TCSEC

The following C2 requirements specified in the *Orange Book* are supported by DEC OSF/1 Version 3.0:

- 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 DEC OSF/1 Version 3.0:

- 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 postanalysis 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))
- OSF/Motif-based interfaces

The audit system is set up from the command line. Maintenance for the audit subsystem is done from the command line or with OSF/Motif-based interfaces (XIsso and XSysAdmin).

DEC OSF/1 Version 3.0 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 default system or the trusted system. By using the `secsetup` command, you can configure your system to use either C2 or non-C2 based commands.

The following I&A features are provided in DEC OSF/1 Version 3.0:

- Password control
  - Configurable maximum password length, 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" and of whether it is "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.

For more information, see the guide *Security*.

11.2.3 Object Reuse

Object reuse is a standard feature of DEC OSF/1 Version 3.0. 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. For more information, see the *OSF/1 Security Object Reuse Study*.

11.2.4 Discretionary Access Controls

Discretionary access controls (DAC) are a standard feature of DEC OSF/1 Version 3.0. 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. Setting permissions is discussed in the guide *Security*.

11.2.5 System Architecture

DEC OSF/1 Version 3.0 maintains a separate execution domain for the thread control block (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.

DEC OSF/1 Version 3.0 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.

DEC OSF/1 Version 3.0 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

DEC OSF/1 Version 3.0 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.

DEC OSF/1 Version 3.0 supports separate operator, administrator, and security officer functions.

11.2.7 Enhanced Security Administration

The DEC OSF/1 Version 3.0 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 C2 security features. The audit subsystem is configurable at kernel link time, regardless of the security level of the system.

11.2.7.2 Windows-Based Administration Utilities

Two window-based utilities are provided to deal with the day-to-day security administration on the local machine. Based on OSF/Motif, the `XSysAdmin` command creates user accounts and the `XIsso` command allows for the modification of system defaults, user accounts, and all of the audit interfaces and devices. Administrators have the flexibility to configure the audit subsystem without the requirement of installing additional C2 security features. The I&A is configured at boot time, so that the system administrator can configure the security level of the system.

For more information, see the `XSysAdmin`(8) and `XIsso`(8) reference pages.

11.3 Digital Security Enhancements

Digital enhanced the security code from the OSF to add support for the following functionality:

- Event and application auditing and an audit-trail analysis tool

  The audit daemon (`auditd`) was ported from ULTRIX, replacing the less robust audit daemon that shipped in OSF/1. The audit-trail analysis tool (`audit_tool`) was also ported from ULTRIX, since OSF/1 did not provide this functionality.

- An improved identification/authentication scheme

  Through the use of a middle-layer interface, the Security Integration Architecture (SIA), DEC OSF/1 Version 3.0 allows users to toggle back and forth between the secure and the nonsecure commands and utilities using the `secsetup` command. By contrast, the OSF/1 security software, when installed, overwrote the unsecure commands and utilities with the secure ones.

- Network Information Service (NIS)

  Although the code as it originally came from the OSF had very little networking support, Digital added support for accessing NIS distributed databases.

  Users on a DEC OSF/1 Version 3.0 C2 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 database is not the same as the encrypted password on the secure system which cannot be viewed by unprivileged users.

  In addition, on a DEC OSF/1 Version 3.0 C2 system NIS creates two version of each NIS map—a protected version, which only privileged users can view; and an unprotected version, which any user can view, but which does not contain all the fields that the protected version contains.

- DECnet

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

- The Distributed Computing Environment (DCE)

  Through SIA, DEC OSF/1 Version 3.0, when configured for C2 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 DEC OSF/1 Version 3.0 secure system and then log in again to DCE.

- A `secsetup` configuration and setup script

  DEC OSF/1 Version 3.0 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.

- Two window-based utilities for local machines

  DEC OSF/1 Version 3.0 supports the `XIsso` utility that permits the modification of system defaults, user accounts, and all of the audit interfaces and devices; and the `XSysAdmin` utility that creates user accounts.

11.4 Performance

With all security options configured and running (including auditing), DEC OSF/1 Version 3.0 shows a performance degradation of only 3%. With auditing turned off, there is no measurable performance degradation. With C2 security configured but not turned on, there is no performance degradation whatsoever.

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

Installation and System Setup 12

12.1 Overview

DEC OSF/1 Version 3.0 supports both a full installation for new systems and an update installation that allows users who already have a version of the operating system installed to update to DEC OSF/1 Version 3.0 without overwriting system files. In addition, DEC OSF/1 Version 3.0 supports a variety of setup scripts that allow users to configure their systems quickly and with relative ease.

12.2 Installation

DEC OSF/1 Version 3.0 supports both a full and an update installation either from a CD-ROM or across the network from a Remote Installation Services (RIS) server. For more information on RIS, see the 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 SPD.

- Full Installation

  A full installation allows users to install DEC OSF/1 Version 3.0 on new systems and is divided into the following two procedures:

  - Basic Installation

    A Basic 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 -l` command.

    The Basic Installation is intended for those users who do not customize their disk partitions, install across multiple disks, and who want to get up and running quickly and easily.

- Advanced Installation

  An Advanced 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 Advanced Installation allows users to select from a menu of optional software subsets, rather than automatically installing only the mandatory subsets.

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

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.

  An interface to the `setld` utility, called `installupdate`, performs the update installation and does the following:

  - Installs new system software
  - Does not overwrite customized system files, like `/etc/passwd` and `/etc/fstab`

    A complete list of protected files is provided in the *Installation Guide*.
  - 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 by hand.
  - 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 -l` command.

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

12.3 System Setup

DEC OSF/1 Version 3.0 provides setup scripts developed by Digital to assist you in configuring the following components of your system:

- Internet Networking
- Local Area Transport (LAT)
- Berkeley Internet Name Domain Service (BIND)
- Network Information Service (NIS)
- Network File System (NFS)
- UNIX-to-UNIX Copy Program (UUCP)
- Network Time Protocol (NTP)
- Mail
- Simple Network Management Protocol (SNMP)
- Printers
- Streams
- License Management Facility (LMF)
- Verifier/Exerciser Tool (VET)
- Prestoserve I/O Acceleration
- Security (BSD C2)
- Security Auditing
- Update Administration Utility

The setup scripts are menu-driven, character-cell scripts and are accessible as superuser from either the command-line or from the System Setup selection on the Applications menu of the System Manager.

Note that these setup scripts can be accessed through a "Welcome Window" that appears during the first root log in after you have installed DEC OSF/1. In addition to providing access to the setup scripts, the "Welcome Window" also provides the following information:

- General, introductory information about configuring the system
- Information about loading license PAKs
- Information about accessing online documentation
- Information about adding users

The "Welcome Window" menu can be accessed after the initial root login by typing the following command as `root`:

```
# /usr/sbin/setup -a
```

For more information on setup scripts, see the *Installation Guide*, *Network Configuration* guide, *System Administration* guide, and the `setup`(8) reference page.

System Administration 13

13.1 Overview

DEC OSF/1 Version 3.0 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 the system administration utilities from the OSF, DEC OSF/1 Version 3.0 also supports the following Digital-specific system administration utilities:

- `setld`

  A utility designed for installing and deleting software kits

- DEC Verifier and Exerciser Tool (DEC VET)

  A system verifier and exerciser utility

- Analysis Tools with Object Modification (ATOM)

  Profiling and analysis tool.

- `uerf`

  A command that formats the binary error-logger files for monitoring system events

- Enhanced kernel debugging

  Support for remote debugging and for writing C extensions to use with the dbx kernel debugger

- Support for Dynamically Loadable Subsystems

  The ability to package, load, and manage kernel subsystems, including device drivers, on DEC OSF/1 systems

- Support for Dynamic System Configuration

  A series of utilities that allow administrators to configure kernel and subsystem attributes at either boot or runtime without having to rebuild the kernel

- Dataless

  A utility using the `bootp` protocol that allows dataless clients to have their `root`, `/var`, and `/usr` partitions served to them across the network

The following sections discuss these Digital utilities in greater detail.

13.2 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 *Programming Support Tools*. 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 encourages application programmers to use the `setld` format when packaging software subsets designed to be installed on DEC OSF/1 systems and explains, in the guide *Programming Support Tools*, how to create kits that are compatible with the `setld` utility. Essentially, a `setld` subset is a compressed `tar` file under strict inventory control that can be installed or deleted from a DEC OSF/1 system with exact verification.

For more information on the `setld` utility, see the guide *Programming Support Tools*, the *System Administration* guide, and the `setld`(8) reference page.

13.3 DEC Verifier and Exerciser Tool

The DEC Verifier and Exerciser Tool (DEC VET) is a system verifier and exerciser utility that runs on all Digital-supported platforms and that performs the following functions:

- Verifies the installation of the base operating system and hardware
- Tests system integration
- Troubleshoots installed systems in multiuser mode
- Troubleshoots multiple, heterogeneous systems over a network, controlling all systems from a single node
- Exercises the CPU, memory, tapes, disks, file systems, printers, terminals, and networks

DEC VET supports both a character-cell and windowing interface, and with the optional `DEC VET Development Kit`, allows programmers to write exercisers for their own devices that can run under DEC VET. For more information on DEC VET, see the *DEC Verifier and Exerciser Tool User's Guide.*

13.4 Analysis Tools with Object Modification

The Analysis Tools with Object Modification (ATOM) Advanced Developer's Kit, an optional subset that ships with DEC OSF/1, 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.

13.5 The uerf Command

The `uerf` command formats system reports after translating to ASCII text the information collected in the file `/usr/adm/binary.errlog` by the binary error logger. The `uerf` command can also format system information in their original binary format.

The `uerf` command comes with a wide array of switches to allow quite detailed searches through the binary error log file, including searches for disk errors, memory errors, informational messages, operating system events, and operational messages, such as those printed out at the console at system startup.

For more information on the `uerf` command, see the *System Administration* guide and the `uerf`(8) reference page.

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

DEC OSF/1 Version 3.0 supports the dynamic configuration of loadable device drivers and other, loadable kernel subsystems, as well as the dynamic configuration and tuning of certain kernel attributes.

Instructions on how to write and package loadable device drivers so that they will install and execute on DEC OSF/1 Version 3.0 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 DEC OSF/1 Version 3.0 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, DEC OSF/1 Version 3.0 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 DEC OSF/1 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 on how to manage kernel attributes, see the *System Administration* guide and the *System Tuning and Performance Management* guide.

13.9 Dataless Management Services

DEC OSF/1 Version 3.0 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.

Maximum System Limits A

A.1 Maximum System Limits

This section lists the maximum system limits for the major components of DEC OSF/1 Version 3.0. For hardware information specific to your individual processor, see the *Software Product Description* (SPD) and the *Systems and Options Catalog*. For information on how to tune system parameters, see the *System Tuning and Performance Management* guide and the *System Administration* guide.

- Backup Utility Limits
  - `cpio`

Files per archive:	4 GB
Files per file system:	4 GB
File size:	32 GB
File name size:	256 bytes
`dev_t`:	64 KB
`dev_major`:	Not used
`dev_minor`:	Not used

  - `dd`

Files per archive:	Not used
Files per file system:	Not used
File size:	32 GB
File name size:	Not used
`dev_t`:	Not used
`dev_major`:	Not used
`dev_minor`:	Not used

- `dump`

Files per archive:	4 GB
Files per file system:	4 GB
File size:	32 GB
File name size:	No limit (part of the inode data)
`dev_t`:	4 GB
`dev_major`:	Not used
`dev_minor`:	Not used

- `tar`

Files per archive:	No limit
Files per file system:	No limit
File size:	8 GB
File name size:	256 bytes (with prefix)
`dev_t`:	Not used
`dev_major`:	16 MB
`dev_minor`:	16 MB

- Device Addressing Limits
  - Device Access

    In DEC OSF/1 Version 3.0, there are two types of disk device access: raw or character and block or buffered.

    For raw or character access, the structure field `uio.uio_offset` describes the byte offset within the disk partition. In DEC OSF/1 Version 3.0, the `uio_offset` is an unsigned 64-bit value, allowing an offset up to 2^{64} or 18 Exabytes. This value is converted to a physical block/sector number that is the data transfer start position. The physical block/sector number is limited by the structure field `buf.b_blkno`.

    For block or buffered access, the structure field `buf.b_blkno` describes the block/sector offset within the disk partition and is a signed 32-bit value. Since DEC OSF/1 Version 3.0 supports a fixed 512-byte block/sector size defined by `DEV_BSIZE`, the offset is limited to 1 TB.

- Major-Minor Numbers (`dev_t`)

  Devices are described by a major-minor pair of numbers, where the major number describes the device driver and the minor number describes the device. In DEC OSF/1 Version 3.0, these parings are represented by a 32-bit value described by the type `dev_t`. The major number portion of `dev_t` consists of bits 20 to 31 (12 bits). Since each device driver requires 12 bits for its major number, 4096 device drivers may be statically configured into the system.

  The minor number portion of `dev_t` consists of bits 0 to 19 (20 bits). The first 6 bits (0 to 5) of the minor number are device-specific, describing the type of device that is connected to the system. For disks, these bits are known as the partition number, and theoretically they can describe a maximum of 64 partitions. DEC OSF/1 Version 3.0 restricts support to 8 partitions, however.

  Bits 6 to 15 of the minor number (14 bits) represent the unit number, and therefore DEC OSF/1 Version 3.0 supports a theoretical maximum of 16384 units; however, the actual number of supported units is a function of how many buses each individual hardware platform supports. On a DECstation 3000 Model 500, for example, the number of supported units is 14.

- SCSI/CAM Addressing

  Common Access Method (CAM) is an ANSI-proposed standard for a common software interface to Small Computer Systems Interface (SCSI). There are no restrictions or limitations within CAM for disk block addressing; the address is an incoming value.

  For SCSI-2, the Command Descriptor Block (CDB) defines the starting disk block number for the transfer. In DEC OSF/1 Version 3.0, the 10-byte CDB has 4 bytes reserved for the disk block address. This is an unsigned 32-bit value allowing 2^{32} - 1 or 4 Gigasectors of addressing, which corresponds to 2 TB given the 512-byte block/sector size.

  In DEC OSF/1 Version 3.0, the SCSI/CAM driver can address a maximum of 64 buses, with up to 7 device targets per bus, and a maximum of 8 LUNs per device target. As a result, in DEC OSF/1 Version 3.0, SCSI/CAM can address a maximum of 3584 devices.

- Redundant Array of Independent Disks

  DEC OSF/1 Version 3.0 supports three Redundant Array of Independent Disks (RAID) controllers: two for the SCSI bus (HSZ10 and HSZ40) and one for the EISA bus (SWXCR). Each RAID device is seen by the operating system as a single target device (that is, as a single disk) with up to 8 Logical Unit Numbers (LUNs) on the SCSI

controllers and 8 Logical Units (LUs) on the EISA controller, regardless of the number of disks on each RAID device.

The HSZ10 SCSI RAID controller supports a maximum of 35 back-end disks; the HSZ40 SCSI RAID controller, a maximum of 42 back-end disks. As a result of hardware constraints, the maximum number of HSZ10 disks that can be concatenated into a logical volume is 5; the maximum number of HSZ40 disks that can be concatenated into a RAID 0 set for a logical volume is 14 with a total size limit of 32 GB.

The SWXCR EISA RAID controller supports 4 **packs** per controller (a pack is a collection of disks over which data is striped) and 8 RE disk drives per pack, for a total of 32 RE disks per controller. Logical volume sizes are not fixed sizes as compared to other disk devices. The size of a logical volume is configurable based on needs with a total size limit of 32 GB. In addition, the SWXCR controller may have either a one or a three channel SCSI adapter which supports 7 or 21 back-end SCSI disks, respectively.

Although RAID theoretically increases the number of addressable disks significantly, Digital recommends that the maximum number of devices for each system–even with RAID configured–should not exceed the numbers listed in the following section on device limits per processor.

- Disklabel

  The disklabel defines the partitions of a disk and their starting block/sector number. The starting block/sector number of a partition is defined by the structure field `partition.p_offset`, which is an unsigned 32-bit value allowing 2 TB of addressing with a 512-byte block/sector size.

- Device Limits Per Processor
  - SCSI Bus
    - Buses/Adapter – 1 or 2
    - RZ/TZ Devices/Bus – 7
  - Digital Storage Architecture (DSA)
    - CIXCD Controller – 1 to 4 HSCs
    - HSC – Up to the limit supported by each HSC, with a combined maximum of 96 RA/TA devices
    - KDM Controller – 8 RA/TA devices up to 6 controllers
    - CI Star – A maximum of 16 nodes

  Note that DSA supports a maximum of 96 RA/TA devices.

- Processors
  * DEC 2000 series
    + Model 300 – 28 RZ/TZ Devices

      (4 Single SCSI Adapters) = 4 SCSI buses * 7 devices/bus
    + Model 500 – 28 RZ/TZ Devices

      (4 Single SCSI Adapters) = 4 SCSI buses * 7 devices/bus
  * DEC 2100 series – 77 RZ/TZ Devices

    (1 PCI Baseboard Single SCSI) + (3 Single SCSI PCI Adapters) + (1 EISA Baseboard Single SCSI) + (2 Tri-SCSI Adapters) = 11 SCSI buses * 7 devices/bus
  * DEC 3000 series
    + Model 800/800S – 98 RZ/TZ Devices

      (1 Baseboard Dual SCSI) + (6 Dual SCSI TURBOchannel Adapters) = 14 SCSI buses * 7 devices/bus
    + Model 600S – 56 RZ/TZ Devices

      (1 Baseboard Dual SCSI) + (3 Dual SCSI TURBOchannel Adapters) = 8 SCSI buses * 7 devices/bus
    + Model 600 – 42 RZ/TZ Devices

      (1 Baseboard Dual SCSI) + (2 Dual SCSI TURBOchannel Adapters) = 6 SCSI buses * 7 devices/bus
    + Model 500 – 98 RZ/TZ Devices

      (1 Baseboard Dual SCSI) + (6 Dual SCSI TURBOchannel Adapters) = 14 SCSI buses * 7 devices/bus
    + Model 500X – 84 RZ/TZ Devices

      (1 Baseboard Dual SCSI) + (5 Dual SCSI TURBOchannel Adapters) = 12 SCSI buses * 7 devices/bus
    + Model 400S – 56 RZ/TZ Devices

      (1 Baseboard Dual SCSI) + (3 Dual SCSI TURBOchannel Adapters) = 8 SCSI buses * 7 devices/bus
    + Model 400 – 42 RZ/TZ Devices

      (1 Baseboard Dual SCSI) + (2 Dual SCSI TURBOchannel Adapters) = 6 SCSI buses * 7 devices/bus
    + Model 300/300X – 35 RZ/TZ Devices

      (1 Baseboard Single SCSI) + (2 Dual SCSI TURBOchannel Adapters) = 5 SCSI buses * 7 devices/bus

+ Model 300L/300LX – 7 RZ/TZ Devices

  (1 Baseboard Single SCSI) = 1 SCSI bus * 7 devices/bus

* DEC 4000 series

  35 RZ/TZ Devices

  (5 Baseboard Single SCSI) = 5 SCSI buses * 7 devices/bus

* DEC 7000/10000 series

  96 RA/TA Devices (DSA)

  112 RZ/TZ Devices (SCSI)

  (8 Dual SCSI XMI Adapters) = 16 SCSI buses * 7 devices/bus

- CPU Limits Per Processor

DEC 2000 series:	1
DEC 2100 series:	4
DEC 3000 series:	1
DEC 4000 series:	2
DEC 7000/10000 series:	6

- File System Limits
  - Maximum Size
    * UNIX File System

      In DEC OSF/1 Version 3.0, the UNIX File System (UFS) file size is limited by the amount of space that the kernel `buf` structure can address. The structure field `buf.b_blkno`, defined as `daddr_t`, is the block/sector offset within a disk partition and is a 32-bit signed value. The block or sector size, `DEV_BSIZE`, is 512 bytes. As a result, the theoretical maximum file system size that DEC OSF/1 Version 3.0 supports is 1 TB ($2^{31} * 2^{9}$). Note that Digital only supports 32 GB (the maximum logical volume size supported by the Logical Volume Manager) and 128 GB (the maximum logical volume size supported by the Logical Storage Manager).

* Network File System

  In DEC OSF/1 Version 3.0, the theoretical maximum size of a file that is accessible through the Network File System (NFS) is as follows:

  + NFS Version 2.0 – 2 GB
  + NFS Version 3.0 – 16 Exabytes ($2^{64} - 1$)

  Digital supports the following maximum file sizes:

  + NFS Version 2.0 – 2 GB
  + NFS Version 3.0 – 128 GB

  Note that an NFS server is always limited by the size of the underlying local file system.

* CD-ROM File Svstem

  The size of the CD-ROM File System (CDFS) files and file systems is limited by the compact read-only optical disk (CD-ROM) media where they reside. Currently, the CD-ROM media supports approximately 0.60 GB. However, DEC OSF/1 is able to support larger CD-ROMs should they become available.

* Advanced File System

  In the Advanced File System (AdvFS), a **volume** is any single logical device which can be a partition on a physical disk or a logical volume. A **domain** is a named set of bound volumes on which filesets are placed. A **fileset** is a named collection of files that is bound to a single domain. An **active fileset** is a fileset that has been mounted, like a UFS file system, for example.

  Although the architectural maximum limit of domains is 2048, in DEC OSF/1 Version 3.0 the AdvFS supports a maximun file size of 128 GB, up to 100 active file domains per system, and a maximum of 250 volumes per domain.

  To decrease the chance of disk errors which may result when a domain is made inaccessible, Digital also recommends that you create no more than 8 volumes per domain.

  Note that while DEC OSF/1 supports an unlimited number of filesets per system, only 256 filesets can be mounted at one time.

  The number of files per fileset is limited by the number of tags available to uniquely identify a file in a fileset. There are approximately 1023 tags per page. With a 31-bit page number, the maximum files per fileset is $1023 * 2^{31}$ or 2196875771904.

Note that over time the actual limit of files per fileset decreases, since a tag can only be used 4096 times due to a sequence number limit.

Although AdvFS can support page sizes larger than 13 bits, the theoretical maximum AdvFS file and fileset size is 16 TB ($2^{13} * 2^{31}$) with a 13-bit page size and 31-bit page number.

* Logical Volume Manager

  In DEC OSF/1 Version 3.0, the Logical Volume Manager (LVM) supports a maximum of 32 physical volumes per volume group, where each physical volume is a disk partition. Up to three volume groups are supported. The supported maximum number of logical volumes per volume group is 256; the supported maximum volume group size is 64 GB; and the supported maximum logical volume size is 32 GB.

  Note that LVM will be retired in a future release, since it has been superseded by the Logical Storage Manager (LSM), which now ships as part of the operating system.

* Logical Storage Manager

  In DEC OSF/1 Version 3.0, the Logical Storage Manager (LSM) supports a maximum of 128 disk groups and 128 disks either in a disk group or across the system.

  In LSM, the term **volume** describes a virtual disk representing an addressable range of disk blocks used by applications such as file systems or databases. DEC OSF/1 Version 3.0 supports a maximum of 128 GB of disk space per system, with a maximum supported volume of 128 GB. The maximum number of supported LSM volumes is 250 for all disk groups in a system.

  In LSM, the term **plex** describes the physical disk or disks that contain a complete copy of a volume's data. So, for example, a mirrored volume would be made up of at least two plexes. In DEC OSF/1 Version 3.0, the maximum number of supported plexes per volume is 8 and the maximum number of supported plexes per system is 256.

  In LSM, the term **subdisk** describes a contiguous portion of a physical disk which can be striped or concatenated together to form a plex. Up to 256 subdisks can be associated with one plex. While up to 1024 subdisks can theoretically be used in one volume, DEC OSF/1 Version 3.0 supports a maximum of 256 subdisks per volume. While as many as 65536 subdisks can theoretically be used on a system, DEC OSF/1 Version 3.0 supports a maximum of 700 subdisks per disk group and 700 subdisks per system.

LSM object names (such as volumes, plexes, subdisks, disk groups), volume attribute names (such as user and group), and `dxlsm view` names are limited to 14 characters.

* Sparse Files

  DEC OSF/1 supports **sparse files** on AdvFS and UFS, which means that the size of a file may exceed the size of the file system where it resides. DEC OSF/1 supports the following maximum sizes for sparse files:

AdvFS	$2^{44} - 512K$
UFS	$2^{44} - 8K$

- Memory Mapped File Limit

  The supported maximum size of a file that can be mapped into memory without segmenting the file is 1 GB.

- Mounts

  * UNIX File System

    The UNIX File System (UFS) supports a total of 512 mounts, which are now allocated dynamically by the system rather than being dependent on statically configured mount tables as they were in previous releases of the operating system.

  * Network File System

    A Network File System (NFS) Version 2.0 or Version 3.0 client can mount a maximum of 2048 files or directories. The `vnodes` necessary to support the NFS-mounts are now allocated dynamically rather than being dependent on a statically configured `vnode` table as they were in previous releases of the operating system.

  * CD-ROM File System

    DEC OSF/1 Version 3.0 supports a maximum of 512 CD-ROM File System (CDFS) mounts.

  * Advanced File System

    The Advanced File System (AdvFS) supports a maximum of 256 mounted filesets. However, each active domain has an invisible mounted fileset associated with it which must be factored into the total number of mounted filesets. So, for example, if you have an active domain with two mounted filesets, the invisible fileset associated with the domain itself brings the total number of mounted filesets to three.

- Open files

  The maximum number of files a process can open is set to 4096 by default in the `OPEN_MAX_SYSTEM` variable in the file `/usr/sys/include/sys/param.h`. This number can be adjusted between 64 and 4096, either in individual programs by using the `setrlimit.2` system call or on a system-wide basis by editing the file `/usr/sys/conf/param.c`, changing the `open_max_soft` and `open_max_hard` variables, and then relinking or rebuilding the kernel. Note that file descriptor entries in the per process file table are dynamically allocated after the initial 64 entries in the `utask` structure are used.

- File Locking Limits

  The DEC OSF/1 file record locking service allows applications to lock any number of bytes in a file in the range of 0 to $2^{63} - 1$ inclusive. File locking is supported by UFS, AdvFS, and both NFS Version 2 and Version 3. Note that since the NFS Version 2 protocol suite only allows ranges to be specified with 32-bit numbers, it supports a file locking range of 0 to $2^{31} - 1$ inclusive.

- Pathname Limits

  AdvFS, UFS, CDFS, and NFS support a maximum pathname component of 255 bytes and a maximum file pathname of 1023 bytes.

- Installation Limits

  This section lists the disk space required for `root`, `/usr`, and `/var` when performing a basic or an advanced installation of DEC OSF/1 Version 3.0.

  - Basic Installation

    `root`: 30 MB
    `/usr`: 148.5 MB
    `/var`: –

  - Advanced Installation (taking all subsets)

    `root`: 40 MB
    `/usr`: 320 MB
    `/var`: 6 MB

- Memory Limits
  - Physical Memory

    The maximum supported memory is different for each individual processor, although the DEC 7000/10000 series—the largest processors—support a total of 14 GB of physical memory. The operating system, however, supports a total of approximately 4 GB of physical memory.

    For more information on supported memory, see the *Systems and Options Catalog* and the SPD.
  - Virtual Memory
    * Per process

      The default virtual memory per process is 2 GB, although available swap space may in many cases be exhausted before this limit is reached.

      This value can be increased to a maximum of 8 TB by defining the `MAXVAS` variable in the system configuration file and relinking or rebuilding the kernel.
    * Page size

      The default page size is 8 KB and is not configurable. The page size is hardware dependent and is set up by the console at boot time.
- Networking Limits
  - Pseudoterminals (`ptys`)

    The maximum number of supported `ptys` is 1400.
  - Network Transfer Rates

    For information on network transfer rates, see Chapter 3.
- Process Limits
  - Per system

    The number of processes per system depends on the value of `MAXUSERS`, which is configurable and set in the configuration file to 32 by default. With `MAXUSERS` at its default value, the number of processes per system is set to 276 in the `NPROC` variable in `/usr/sys/conf/param.c`.

    You can increase this value by either changing the value of `MAXUSERS` in the system configuration file or by adding the `maxproc` variable to the system configuration file and relinking or rebuilding the kernel.

You might increase the value of MAXUSERS to allow more users to log in to your system or to allow applications that run as root to fork more processes than NPROC allows by default. However, increasing the number of processes per system reserves more system memory, so the upper limit of NPROC is dependent upon the system's total memory, the number of actual users on the system, and the requirements of your applications.

- Per user

  The number of processes that each user can fork is set to 64 by default through the CHILDMAX variable in the file /usr/sys/include/sys/syslimits.h. The number of processes per user can be varied by adding the maxuprc variable to the system configuration file, setting its value to some number of processes, and then relinking or rebuilding the kernel.

  You might increase this value if you had an application that needed more processes than CHILDMAX is set to by default. However, increasing the value of maxuprc reserves more system memory, so the upper limit of maxuprc is dependent upon the system's total memory, the number of actual users on the system, and the requirements of your applications.

Conformance to Internet Host Requirements B

This appendix addresses the conformance of DEC OSF/1 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 DEC OSF/1 to the requirements specified in those RFCs is beyond the scope of this appendix.

The DEC OSF/1 Software Product Description (SPD) contains additional technical information about the networking component of DEC OSF/1 and a complete list of RFCs implemented in DEC OSF/1.

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 DEC OSF/1 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 DEC OSF/1 systems can be configured to conditionally comply to Internet Host Requirements when acting as a host.

B.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 the DEC OSF/1's conformance to **must** and **must not** level requirements only. Although DEC OSF/1 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 1540: *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.

DEC OSF/1 was validated against RFC 1600. RFC 1600 renders RFCs 1540, 1500, 1410, 1360, 1280, 1250, 1200, 1140, 1130, 1100, and 1083 obsolete.

- RFC 1340: *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. DEC OSF/1 was validated against RFC 1340. RFC 1340 renders RFC 1060, 1010, as well as many earlier RFCs, obsolete.

B.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 DEC OSF/1 was validated against and that are referenced in the Host Requirement RFCs. Additionally, Table B-8 and Table B-9 list the total **must/must not** level requirements explicitly stated by the Host Requirements RFCs.

Table B-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 B-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 893: *Trailer Encapsulations*	Discusses the motivation for using ''trailer encapsulations'' on Local Area Networks (LANs) and describes the implementation of such an encapsulation on various media.

Table B-1: (continued)

Referenced RFC[a]	Description
RFC 826: *Address Resolution Protocol (ARP)*	Presents a method for converting protocol addresses (IP addresses) to local network addresses (Ethernet addresses).

Table Notes:

a. DEC OSF/1 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 B-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 B-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.

Table B-2: (continued)

Referenced RFC	Description
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 B-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 B-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 B-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 B-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.

Table B-4: (continued)

Referenced RFC	Description
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.
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.

Table B-4: (continued)

Referenced RFC	Description
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
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 B-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 B-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 B-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 B-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.

Table B-6: (continued)

Referenced RFC	Description
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 B-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 B-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. DEC OSF/1 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 B-8 summarizes the **must/must not** level requirements, per layer, explicitly stated by RFC 1122.

Table B-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 B-9 summarizes the **must/must not** level requirements explicitly stated by RFC 1123.

Table B-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)

B.3 Configuring DEC OSF/1 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 DEC OSF/1 does not comply by default. Along with each item is a discussion about why DEC OSF/1 does not comply, and information about how to configure your system to comply with that item.

B.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).

DEC OSF/1 discards the partially reassembled datagram when the reassembly timeout expires. However, by default, DEC OSF/1 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 DEC OSF/1 be configured to generate ICMP Time Exceeded messages. DEC OSF/1 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
        0 redirects sent
```

[1] DEC OSF/1 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 (VrPF) as "MUST IMPLEMENT". DEC OSF/1 does not implement OSPF.

B.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.

B.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, DEC OSF/1 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.

B.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

P

Q

R

S

T

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

DEC OSF/1
Technical Overview
AA-Q0R1B-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 Tear - Fold Here and Tape ----

digital™

No Postage
Necessary
if Mailed in the
United States

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO.33 MAYNARD MASS.

POSTAGE WILL BE PAID BY ADDRESSEE

DIGITAL EQUIPMENT CORPORATION
UEG PUBLICATIONS MANAGER
ZKO3-3/Y32
110 SPIT BROOK ROAD
NASHUA, NH 03062-9987

--- Do Not Tear - Fold Here ----

Cut Along Dotted Line

digital

Printed in the U.S.A.