ULTRIX

digital

POSIX Conformance Document

Order Number: AA-LY25C-TE

The ULTRIX Operating System also meets the requirements of the Federal Information Processing Standard (FIPS 151-1).
## Contents

### About This Manual
- Audience xi
- Organization xi
- Related Documents xii
- Conventions xii

### 1 Scope

### 2 Definitions and General Requirements
- 2.2 Conformance 2–1
- 2.2.3 Language-Dependent Services for the C Programming Language 2–1
  - 2.2.3.3 Common Usage C Language-Dependent System Support 2–1
- 2.3 General Terms 2–1
- 2.4 General Concepts 2–3
- 2.5 Error Numbers 2–3
- 2.6 Primitive System Data Types 2–4
- 2.7 Environment Description 2–5
- 2.8 C Language Definitions 2–5
  - 2.8.2 POSIX Symbols 2–5
- 2.9 Numerical Limits 2–5
- 2.10 Symbolic Constants 2–6

### 3 Process Primitives
- 3.1 Process Creation and Execution 3–1
  - 3.1.1 Process Creation 3–1
    - 3.1.1.2 Description 3–1
3.2  Process Termination ................................................. 3-2
  3.2.1  Wait for Process Termination ......................... 3-2
    3.2.1.2  Description ........................................... 3-2
  3.2.2  Terminate a Process ....................................... 3-2
    3.2.2.2  Description ........................................... 3-2

3.3  Signals ................................................................. 3-2
  3.3.1  Signal Concepts ............................................. 3-2
    3.3.1.1  Signal Names ......................................... 3-2
    3.3.1.2  Signal Generation and Delivery ................... 3-3
    3.3.1.3  Signal Actions ........................................ 3-3
  3.3.2  Send a Signal to a Process ............................... 3-3
    3.3.2.2  Description ........................................... 3-3
  3.3.3  Manipulate Signal Sets .................................... 3-4
    3.3.3.4  Errors ................................................... 3-4
  3.3.5  Examine and Change Blocked Signals ................... 3-4
    3.3.5.2  Description ........................................... 3-4
  3.3.6  Examine Pending Signals ................................. 3-4
    3.3.6.4  Errors ................................................... 3-4

3.4  Timer Operations .................................................... 3-4
  3.4.3  Delay Process Execution ................................. 3-4
    3.4.3.2  Description ........................................... 3-4

4  Process Environment

4.2  User Identification ................................................. 4-1
  4.2.3  Get Supplementary Group IDs ............................ 4-1
    4.2.3.2  Description ........................................... 4-1
  4.2.4  Get User Name .............................................. 4-1
    4.2.4.3  Returns ................................................... 4-1
    4.2.4.4  Errors ................................................... 4-1
4.4 System Identification ................................................................. 4-1
  4.4.1 System Name ......................................................................... 4-1
    4.4.1.2 Description ...................................................................... 4-1
    4.4.1.4 Errors ............................................................................ 4-2
  4.4.1.2 Description ...................................................................... 4-1
  4.4.1.4 Errors ............................................................................ 4-2

4.5 Time ......................................................................................... 4-2
  4.5.1 Get System Time .................................................................... 4-2
    4.5.1.4 Errors ........................................................................... 4-2
  4.5.2 Process Times ........................................................................ 4-2
    4.5.2.3 Returns .......................................................................... 4-2
    4.5.2.4 Errors ........................................................................... 4-2

4.6 Environment Variables ............................................................. 4-2
  4.6.1 Environment Access ............................................................... 4-2
    4.6.1.3 Returns .......................................................................... 4-2
    4.6.1.4 Errors ........................................................................... 4-3

4.7 Terminal Identification ............................................................... 4-3
  4.7.1 Generate Terminal Pathname .................................................. 4-3
    4.7.1.3 Returns .......................................................................... 4-3
    4.7.1.4 Errors ........................................................................... 4-3
  4.7.2 Determine Terminal Device Name ............................................ 4-3
    4.7.2.2 Description .................................................................... 4-3
    4.7.2.3 Errors ........................................................................... 4-3

4.8 Configurable System Variables .................................................. 4-3
  4.8.1 Get Configurable System Variables ....................................... 4-3
    4.8.1.2 Description .................................................................... 4-3

5 Files and Directories .....................................................................
  5.1 Directories ................................................................................ 5-1
    5.1.1 Format of Directory Entries ................................................. 5-1
    5.1.2 Directory Operations .......................................................... 5-1
      5.1.2.2 Description .................................................................. 5-1
      5.1.2.4 Errors ......................................................................... 5-1
  5.2 Working Directory ..................................................................... 5-2
    5.2.2 Working Directory Pathname ................................................. 5-2
      5.2.2.2 Description .................................................................. 5-2
5.3 General File Creation ................................................................. 5-2
  5.3.1 Open a File ............................................................................... 5-2
    5.3.1.2 Description ....................................................................... 5-2
  5.3.3 Set File Creation Mask .............................................................. 5-2
    5.3.3.2 Description ....................................................................... 5-2
  5.3.4 Link to a File ............................................................................ 5-3
    5.3.4.2 Description ....................................................................... 5-3

5.4 Special File Creation ................................................................. 5-3
  5.4.1 Make a Directory ...................................................................... 5-3
    5.4.1.2 Description ....................................................................... 5-3
  5.4.2 Make a FIFO Special File ............................................................ 5-3
    5.4.2.2 Description ....................................................................... 5-3

5.5 File Removal .................................................................................. 5-3
  5.5.2 Remove a Directory .................................................................... 5-3
    5.5.2.2 Description ....................................................................... 5-3
  5.5.3 Rename a File ........................................................................... 5-3
    5.5.3.2 Description ....................................................................... 5-3

5.6 File Characteristics ......................................................................... 5-4
  5.6.1 File Characteristics: Header and Data Structure ......................... 5-4
    5.6.1.2 The <sys/stat.h> File Modes ............................................. 5-4
    5.6.1.3 The <sys/stat.h> Time Entries .......................................... 5-4
  5.6.2 Get File Status .......................................................................... 5-4
    5.6.2.2 Description ....................................................................... 5-4
  5.6.3 File Accessibility ....................................................................... 5-4
    5.6.3.2 Description ....................................................................... 5-4
    5.6.3.4 Errors ............................................................................... 5-4
  5.6.4 Change File Modes .................................................................... 5-4
    5.6.4.2 Description ....................................................................... 5-4
  5.6.5 Change Owner and Group of a File .............................................. 5-4
    5.6.5.2 Description ....................................................................... 5-4
    5.6.5.4 Errors ............................................................................... 5-4
5.7 Configurable Pathname Variables ............................................................... 5-5
  5.7.1 Get Configurable Pathname Variables ............................................... 5-5
    5.7.1.2 Description ......................................................................... 5-5
    5.7.1.4 Errors ................................................................................ 5-5

6 Input and Output Primitives

6.4 Input and Output ...................................................................................... 6-1
  6.4.1 Read from a File ................................................................. 6-1
    6.4.1.2 Description ......................................................................... 6-1
    6.4.1.4 Errors ................................................................................ 6-1
  6.4.2 Write to a File ........................................................................... 6-1
    6.4.2.2 Description ......................................................................... 6-1
    6.4.2.4 Errors ................................................................................ 6-1

6.5 Control Operations on Files ....................................................................... 6-2
  6.5.2 File Control ........................................................................... 6-2
    6.5.2.2 Description ......................................................................... 6-2
    6.5.2.4 Errors ................................................................................ 6-2
  6.5.3 Reposition Read/Write File Offset .................................................. 6-2
    6.5.3.2 Description ......................................................................... 6-2

7 Device- and Class-Specific Functions

7.1 General Terminal Interface ....................................................................... 7-1
  7.1.1 Interface Characteristics .................................................................. 7-1
    7.1.1.3 The Controlling Terminal ..................................................... 7-1
    7.1.1.5 Input Processing and Reading Data ........................................ 7-1
    7.1.1.6 Canonical Mode Input Processing .......................................... 7-1
    7.1.1.7 Non-Canonical Mode Input Processing ..................................... 7-1
    7.1.1.8 Writing Data and Output Processing ....................................... 7-1
    7.1.1.9 Special Characters .............................................................. 7-2
  7.1.2 Settable Parameters ............................................................................ 7-2
    7.1.2.1 The termios Structure .......................................................... 7-2
    7.1.2.2 Input Modes ........................................................................ 7-2
    7.1.2.3 Output Modes ....................................................................... 7-2
    7.1.2.4 Control Modes ...................................................................... 7-3
    7.1.2.5 Local Modes ......................................................................... 7-3
    7.1.2.6 Special Control Characters ................................................... 7-3
    7.1.2.7 Baud Rate Functions .............................................................. 7-4
      7.1.2.7.2 Description ................................................................... 7-4
7.1.2.7.4 Errors ................................................................. 7-4

7.2 General Terminal Interface Control Functions ........................................... 7-4
7.2.2 Line Control Functions ................................................................. 7-4
    7.2.2.2 Description ................................................................. 7-4
7.2.3 Get Foreground Process Group ID .................................................... 7-5
    7.2.3.2 Description ................................................................. 7-5
7.2.4 Set Foreground Process Group ID ..................................................... 7-5
    7.2.4.2 Description ................................................................. 7-5

8 Language-Specific Services for the C Programming Language

Conformance ................................................................. 8-1
Implementation Conformance ................................................... 8-1

8.1 Referenced C Language Routines ....................................................... 8-1
    8.1.1 Extensions to Time Functions ............................................. 8-1
    8.1.2 Extensions to the setlocale Function ....................................... 8-1
        8.1.2.2 Description ............................................................ 8-1

8.2 FILE-Type C Language Functions ....................................................... 8-2
    8.2.1 Map a Stream Pointer to a File Descriptor .................................. 8-2
        8.2.1.4 Errors ................................................................. 8-2
    8.2.2 Open a Stream on a File Descriptor ......................................... 8-2
        8.2.2.4 Errors ................................................................. 8-2

8.3 Other C Language Functions ........................................................... 8-2
    8.3.2 Set Time Zone ................................................................. 8-3
        8.3.2.2 Description ............................................................ 8-3

9 System Databases

9.1 System Databases ................................................................. 9-1
9.2 Database Access ................................................................. 9-1
    9.2.1 Group Database Access ..................................................... 9-1
        9.2.1.2 Description ............................................................ 9-1
9.2.1.3 Returns ................................................................. 9-1
9.2.1.4 Errors ................................................................. 9-2
9.2.2 User Database Access .............................................. 9-2
  9.2.2.2 Description ....................................................... 9-2
  9.2.2.3 Returns ............................................................. 9-2
  9.2.2.4 Errors ............................................................... 9-2

10 Data Interchange Format

10.1 Archive/Interchange File Format .................................... 10-1
  10.1.1 Extended tar Format ........................................... 10-1
  10.1.2 Extended cpio Format ....................................... 10-1
    10.1.2.1 Header ......................................................... 10-1
    10.1.2.2 File Name ..................................................... 10-2
    10.1.2.4 Special Entries ............................................ 10-2
    10.1.2.5 The cpio Values .......................................... 10-2
  10.1.3 Multiple Volumes ............................................... 10-2

A POSIX FIPS Additional Requirements

The ULTRIX Operating System also meets the requirements of the Federal Information Processing Standard (FIPS 151-1).

**Audience**

The *POSIX Conformance Document* is written for people who are evaluating operating systems for POSIX conformance and those who want to learn how ULTRIX Version 4.0 implements various POSIX features. The reader should be familiar with IEEE Std 1003.1-1988 or ISO DIS 9945-1 (1989).

**Organization**

This document consists of chapters whose sections briefly explain how ULTRIX Version 4.0 conforms to the implementation-defined features described in POSIX.1.
Related Documents

This manual should be used in conjunction with the *ULTRIX Reference Pages Sections 1-8*.


The IEEE Std 1003.1-1988 can be obtained from the following address:

Publication Sales
IEEE Service Center
P.O. Box 1331
445 Hoes Lane
Piscataway, NJ 08855-1331
(201) 981-0060

The ISO document can be obtained from the following address:

ISO Central Secretariat,
1 rue de Varembi,
CH-1211 Geneva 20,
Switzerland
+41 22 7341240

Relevant information is also contained in the POSIX-related Federal Information Processing Standards (FIPS) documentation. The FIPS documentation can be obtained from the following address:

National Technical Information Service
US Department of Commerce
5285 Port Royal Road
Springfield, VA 22161
(703) 487-4650

Conventions

**system output** This typeface is used in interactive examples to indicate system output and also in code examples and other screen displays. In text, this typeface is used to indicate the exact name of a command, option, partition, pathname, directory, or file.

**UPPERCASE**

**lowercase** The ULTRIX system differentiates between lowercase and uppercase characters. Literal strings that appear in text, examples, syntax descriptions, and function definitions must be typed exactly as shown.

**filename** In examples, syntax descriptions, and function definitions, italics are used to indicate variable values; and in text, to give references to other documents.

This *POSIX Conformance Document* has the same structure as POSIX.1 and is intended to be used in conjunction with that standard.

Sections appearing in this document provide implementation-specific characteristics of the ULTRIX Operating System.

There are two implementations of the ULTRIX Operating System, Version 4.0, one for RISC processors and one for VAX processors. The behavior on both processors is identical with regard to POSIX conformance issues, except where explicitly noted.
Definitions and General Requirements

This chapter provides terminology, concepts, definitions, and general requirements used in this document. For additional information on achieving a POSIX.1 conformant environment, see Chapter 8 of this document.

2.2 Conformance


The ULTRIX POSIX environment supports the functional behavior described in POSIX.1. For information on programming in a POSIX environment, refer to the *Guide to Languages and Programming*.

The ULTRIX Operating System also meets the requirements of the Federal Information Processing Standard (FIPS 151-1).

2.2.3 Language-Dependent Services for the C Programming Language

2.2.3.3 Common Usage C Language-Dependent System Support — The library functions present in ULTRIX conform to the ANSI C Standard (X3.159-1989). For details, see Chapter 8 of this document.

2.3 General Terms

The following paragraphs provide brief descriptions of the implementation-defined features for ULTRIX systems.

appropriate privileges

On ULTRIX systems, all appropriate privileges are associated with the superuser.
character special file
On ULTRIX systems, types of character special files include terminal device files, raw storage devices (for example, tapes or disks), and pseudodevices (for example, /dev/kmem).

file
In addition to the file types defined in POSIX.1, ULTRIX systems also have symbolic links and sockets.

file group class
There are no other members of the file group class beyond those defined in POSIX.1.

parent process ID
A new process is created by a currently active process. The parent process ID of a process is the process ID of its creator for the lifetime of the creator. After the creator's lifetime has ended, the parent process ID is the process ID of the initialization process (process ID = 1).

pathname
On ULTRIX systems, multiple successive slashes (/), including two or more leading slashes, are interpreted as a single slash.

read-only file system
On ULTRIX systems, a read-only file system is a file system that has been mounted in one of the following ways:
• Using the mount system call with the rwflag argument set to 1
• Using the mount utility with the -r option set
• Using the option string ro from the /etc/fstab file

Mounting a read-only file system disables writing to the device in which the file system resides. Operations that require creating, writing, or updating are not permitted and result in ULTRIX systems issuing the error [EROFS]. ULTRIX systems do not update the file access time on read requests for files on a read-only file system. Files and directories that reside on a read-only file system can only be read.

supplementary group ID
On ULTRIX systems, the effective group ID of a process is included in its list of supplementary group IDs.
2.4 General Concepts

The following paragraphs provide brief descriptions of the implementation-defined concepts for ULTRIX systems.

extended security controls

ULTRIX systems do not provide different access control or privilege mechanisms beyond those described in POSIX.1. Additional security controls provided by ULTRIX systems do not alter or override the defined semantics of any of the functions in POSIX.1. For further information, refer to the Security Guide for Administrators.

file access permissions

ULTRIX systems do not provide additional or alternate file access control mechanisms.

file times update

Fields marked for update are updated periodically.

2.5 Error Numbers

The error condition [EFAULT] denotes an invalid address in the argument of a call, as detected by the applicable hardware. When this error is encountered on an ULTRIX system, the error number (errno) for [EFAULT] is returned. The system reliably detects [EFAULT], but it is only detected for system calls. Library routines might get either the signal SIGSEGV or SIGBUS. See the ULTRIX Reference Pages to determine which routines are considered library routines.

In addition to those errors specified in POSIX.1, the following errors are defined in <errno.h>. For a more complete description of these errors, see errno(2) in the ULTRIX Reference Pages.

<table>
<thead>
<tr>
<th>Error</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EADDRINUSE</td>
<td>Address already in use</td>
</tr>
<tr>
<td>EADDRNOTAVAIL</td>
<td>Can't assign requested address</td>
</tr>
<tr>
<td>EAFNOSUPPORT</td>
<td>Address family not supported by protocol family</td>
</tr>
<tr>
<td>EALIGN</td>
<td>Alignment error</td>
</tr>
<tr>
<td>EALREADY</td>
<td>Operation already in progress</td>
</tr>
<tr>
<td>ECONNABORTED</td>
<td>Software caused connection abort</td>
</tr>
<tr>
<td>ECONNREFUSED</td>
<td>Connection refused</td>
</tr>
<tr>
<td>ECONNRESET</td>
<td>Connection reset by peer</td>
</tr>
<tr>
<td>EDESTADDRREQ</td>
<td>Destination address required</td>
</tr>
<tr>
<td>EDQUOT</td>
<td>Disc quota exceeded</td>
</tr>
<tr>
<td>EFBIG</td>
<td>Maximum file size is $2^{64}-1$</td>
</tr>
<tr>
<td>EHOSTDOWN</td>
<td>Host is down</td>
</tr>
<tr>
<td>EHOSTUNREACH</td>
<td>No route to host</td>
</tr>
<tr>
<td>Error</td>
<td>Description</td>
</tr>
<tr>
<td>------------</td>
<td>----------------------------------</td>
</tr>
<tr>
<td>EIDRM</td>
<td>Identifier removed</td>
</tr>
<tr>
<td>EINVALPROGRESS</td>
<td>Operation now in progress</td>
</tr>
<tr>
<td>EISCONN</td>
<td>Socket is already connected</td>
</tr>
<tr>
<td>ELOOP</td>
<td>Too many levels of symbolic links</td>
</tr>
<tr>
<td>EMSGSIZE</td>
<td>Message too long</td>
</tr>
<tr>
<td>ENETDOWN</td>
<td>Network is down</td>
</tr>
<tr>
<td>ENETRESET</td>
<td>Network dropped connection on reset</td>
</tr>
<tr>
<td>ENETUNREACH</td>
<td>Network is unreachable</td>
</tr>
<tr>
<td>ENOBUFS</td>
<td>No buffer space available</td>
</tr>
<tr>
<td>ENOMSG</td>
<td>No message of desired type</td>
</tr>
<tr>
<td>ENOPROTOOPT</td>
<td>Protocol not available</td>
</tr>
<tr>
<td>ENOTBLK</td>
<td>Block device required</td>
</tr>
<tr>
<td>ENOTCONN</td>
<td>Socket is not connected</td>
</tr>
<tr>
<td>ENOTSOCK</td>
<td>Socket operation on non-socket</td>
</tr>
<tr>
<td>EOPNOTSUPP</td>
<td>Operation not supported on socket</td>
</tr>
<tr>
<td>EPFNOSUPPORT</td>
<td>Protocol family not supported</td>
</tr>
<tr>
<td>EPROCLIM</td>
<td>Too many processes</td>
</tr>
<tr>
<td>EPROTONOSUPPORT</td>
<td>Protocol not supported</td>
</tr>
<tr>
<td>EPROTOTYPE</td>
<td>Protocol wrong type for socket</td>
</tr>
<tr>
<td>EREMOTE</td>
<td>Too many levels of remote in path</td>
</tr>
<tr>
<td>ESHUTDOWN</td>
<td>Can't send after socket shutdown</td>
</tr>
<tr>
<td>ESOCKTNOSUPPORT</td>
<td>Socket type not supported</td>
</tr>
<tr>
<td>ESTALE</td>
<td>Stale NFS file handle</td>
</tr>
<tr>
<td>ETIMEDOUT</td>
<td>Connection timed out</td>
</tr>
<tr>
<td>ETOOMANYREFS</td>
<td>Too many references: can't splice</td>
</tr>
<tr>
<td>ETXTBSY</td>
<td>Text file busy</td>
</tr>
<tr>
<td>EUSER</td>
<td>Too many users</td>
</tr>
<tr>
<td>EWOULDBLOCK</td>
<td>Resource deadlock would occur</td>
</tr>
</tbody>
</table>

### 2.6 Primitive System Data Types

In addition to the data types listed in Table 2-1 of POSIX.1, ULTRIX systems define the following system data type in `<sys/types.h>`:

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>time_t</td>
<td>As defined in ANSI C</td>
</tr>
</tbody>
</table>

For additional information, refer to `types(5)` in the ULTRIX Reference Pages.
2.7 Environment Description

Environment variable names can contain any 8-bit character except an equal sign (=) or a NUL ('\0'). However, built-in shell commands can only operate on environment variable names that contain only underscores, digits, and letters (A-Z and a-z) and that do not begin with a digit.

2.8 C Language Definitions

2.8.2 POSIX Symbols

In addition to the _POSIX_SOURCE feature test macro, the _XOPEN_SOURCE macro is also available on ULTRIX systems. This macro enables those features specified by the X/Open Portability Guide, Issue 3.

2.9 Numerical Limits

The <limits.h> header contains the following POSIX-related definitions. Note that these values cannot be changed at runtime.

```c
NGROUPS_MAX 32 /* max # of groups */
NAME_MAX 255 /* max # of characters in a file name */
MAX_INPUT 256 /* max # of bytes in terminal input queue */
MAX_CANON 256 /* max # of bytes in term canon input line */
OPEN_MAX 64 /* max # of files a process can have open */
PATH_MAX 1024 /* max # of characters in a pathname */
LINK_MAX 32766 /* max # of links to a single file */
PIPE_BUF 4096 /* max # bytes atomic in write to pipe */
```

On the VAX architecture, <limits.h> also contains the following definition:

```c
ARG_MAX 10240 /* max length of arguments to exec */
```

On the RISC architecture, <limits.h> also contains the following definition:

```c
ARG_MAX 20480 /* max length of arguments to exec */
```

The symbol CHILD_MAX is not defined in <limits.h>; its value is set at system configuration time and can be retrieved by the sysconf function.
2.10 Symbolic Constants

The `<unistd.h>` header contains the following POSIX-related definitions. These do not vary.

```
R_OK  4  /* Test for "Read" Permission */
W_OK  2  /* Test for "Write" Permission */
X_OK  1  /* Test for "Execute" (Search) Permission */
F_OK  0  /* Test for existence of file */
SEEK_SET 0  /* Set file offset to offset */
SEEK_CUR 1  /* Set file offset to current plus offset */
SEEK_END 2  /* Set file offset to EOF plus offset */
```

```
/* POSIX options */
_POSIX_JOB_CONTROL 1  /*Job Control Present */
_POSIX_SAVED_IDS 1  /*Support saved-set-ids feature */
_POSIX_VERSION 198808L /*POSIX version */
_POSIX_CHOWN_RESTRICTED 1  /*chown() restricted to superuser */
_POSIX_NO_TRUNC 1  /*Pathname longer than NAME_MAX err */
_POSIX_VDISABLE 0  /*termio(s) special character disable*/
```
This chapter describes the system services provided by ULTRIX systems. These system services deal with processes, interprocess signals, and timers.

3.1 Process Creation and Execution

3.1.1 Process Creation

3.1.1.2 Description — On ULTRIX systems, after a for k () call, a new process (child process) inherits process characteristics from the process that created it, including shared memory allocation. Each open directory stream in the child process shares the directory stream positioning with the corresponding directory stream of the parent.

3.1.1.4 Errors — For the for k () function, ULTRIX systems detect the conditions and return the corresponding errno value for [ENOMEM].

3.1.2 Execute a File

3.1.2.2 Description — On ULTRIX systems, NULL terminators are included in the argument byte count. If the argument file does not contain a slash (/) character and the environment variable PATH is not set, the system searches the current directory for the file.

In addition to the attributes inherited by a new image as listed in POSIX.1, on ULTRIX systems, a new process image inherits the following attributes from the calling process image:

- Resource usages (see getrusage(2))
- Interval timers (see getitimer(2))
- Resource limits (see getrlimit(2))

ARG_MAX is the maximum allowed length of the argument list, including NULL terminators and environment data.

On ULTRIX systems, if the exec () function fails but is able to locate the process image file, the st_atime field is not marked for update.

3.1.2.4 Errors — For the exec () functions, ULTRIX systems detect the conditions and return the corresponding errno value for [ENOMEM].
3.2 Process Termination

3.2.1 Wait for Process Termination

3.2.1.2 Description – On ULTRIX systems, a process that is stopped by the process tracing mechanism, ptrace(2), returns the octal value 177 to either wait() or waitpid(). Children of a terminated process are assigned the init process (process id \( pid = 1 \)) as their new parent process.

3.2.2 Terminate a Process

3.2.2.2 Description – On ULTRIX systems, children of a terminated process are assigned the init() process (process id \( pid = 1 \)) as their new parent process.

The sending of a SIGCHLD signal to the parent process is supported.

On ULTRIX systems, _POSIX_JOB_CONTROL is supported. When the exit of a process causes a process group to become orphaned and any member of the newly orphaned process group is stopped, a SIGHUP signal, followed by a SIGCONT signal, is sent to each process in the newly-orphaned process group.

3.3 Signals

3.3.1 Signal Concepts

3.3.1.1 Signal Names – The following additional signals occur in ULTRIX systems. For a complete description of the default actions, refer to POSIX.1.

The values for the default actions are as follows:

- 1 = Abnormal termination of the process
- 2 = Ignore the signal

<table>
<thead>
<tr>
<th>Symbolic Constant</th>
<th>Default Action</th>
<th>Generating Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>SIGTRAP</td>
<td>1</td>
<td>Trace trap</td>
</tr>
<tr>
<td>SIGIOT</td>
<td>1</td>
<td>IOT instruction</td>
</tr>
<tr>
<td>SIGEMT</td>
<td>1</td>
<td>EMT instruction</td>
</tr>
<tr>
<td>SIGBUS</td>
<td>1</td>
<td>Bus error</td>
</tr>
<tr>
<td>SIGSYS</td>
<td>1</td>
<td>Bad argument to a system call</td>
</tr>
<tr>
<td>SIGURG</td>
<td>2</td>
<td>Urgent condition present on socket</td>
</tr>
<tr>
<td>SIGIO</td>
<td>2</td>
<td>I/O is possible on a descriptor</td>
</tr>
<tr>
<td>SIGXCPU</td>
<td>1</td>
<td>CPU time limit exceeded</td>
</tr>
<tr>
<td>SIGXFSZ</td>
<td>1</td>
<td>File size limit exceeded</td>
</tr>
<tr>
<td>SIGVTALRM</td>
<td>1</td>
<td>Virtual time alarm</td>
</tr>
<tr>
<td>SIGPROF</td>
<td>1</td>
<td>Profiling timer alarm</td>
</tr>
</tbody>
</table>
### Symbolic Constants and Default Actions

<table>
<thead>
<tr>
<th>Symbolic Constant</th>
<th>Default Action</th>
<th>Generating Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>SIGWINCH</td>
<td>2</td>
<td>Window size change</td>
</tr>
<tr>
<td>SIGLOST</td>
<td>1</td>
<td>Lock not reclaimed after server recovery</td>
</tr>
</tbody>
</table>

#### 3.3.1.2 Signal Generation and Delivery

On ULTRIX systems, a subsequent occurrence of a pending signal does not cause the signal to be delivered more than once.

On ULTRIX systems, each signal has an associated integer value. In the case of multiple simultaneous pending signals, the pending signal with the lowest associated integral value is handled first.

The additional signals supported by ULTRIX systems are generated under the conditions indicated in Section 3.3.1.1 of this document.

#### 3.3.1.3 Signal Actions

On ULTRIX systems, if a process ignores a SIGFPE, SIGILL, or SIGSEGV signal that was not generated by the `kill()` system call or the `raise()` function as defined in the C Standard, the signal is ignored.

If a process sets the action for the SIGCHLD signal to SIG_IGN, the signal is ignored.

On ULTRIX systems, if a process returns normally from a signal-catching function for a SIGFPE, SIGILL, or SIGSEGV signal that was not generated by the `kill()` system call or the `raise()` function as defined by the C Standard, the process continues executing.

If a process establishes a signal-catching function for the SIGCHLD signal while it has a terminated child process for which it has not waited, a SIGCHLD signal is generated to indicate that the child process has terminated.

#### 3.3.2 Send a Signal to a Process

##### 3.3.2.2 Description

On ULTRIX systems, the `kill()` system call takes two arguments, `pid` and `sig`. The `pid` is the process ID of the process that is to receive the signal `sig`.

On ULTRIX systems, system processes are not affected by signals from `kill(-1, sig)` or `kill(0, sig)`. System processes are defined as any process whose parent `pid` is equal to zero. This includes the following processes:

- Initialization (process ID = 1)
- Swapper (process ID = 0)
- Pager (process ID = 2)

If the effective user ID of a receiving process is altered through the use of the S_ISUID mode bit, ULTRIX does not permit the application to receive a signal sent by the parent process or by a process with the same real user ID.

If the `pid` is zero, the `sig` is sent to all processes whose process group ID is equal to the process group ID of the sender and for which the process has permission to send a signal, excluding system processes. On ULTRIX systems, no process that is not a system process can have a `pid` of less than 100.
If the pid is -1 and the user is running as root (user ID = 0), the sig is sent to all processes, except system processes.

3.3.3 Manipulate Signal Sets

3.3.3.4 Errors – For the sigaddset(), sigdelset(), and sigismember() functions, ULTRIX systems detect the conditions and return the corresponding errno value for [EINVAL].

3.3.5 Examine and Change Blocked Signals

3.3.5.2 Description – On ULTRIX systems, if any of the SIGFPE, SIGILL, or SIGSEGV signals are generated while they are blocked, unless the signal was generated by a call to the kill() system call or the raise() function as defined by the C Standard, the signal is blocked.

3.3.6 Examine Pending Signals

3.3.6.4 Errors – For the sigpending() system call, ULTRIX systems detect when the argument to sigpending() points to memory that is not a valid part of the process address space and return the errno value [EFAULT].

3.4 Timer Operations

3.4.3 Delay Process Execution

3.4.3.2 Description – The sleep() function establishes a signal handler for SIGALRM. Therefore, the SIGALRM signal cannot be ignored or blocked during the execution of the sleep() function.

If a SIGALRM signal is generated for the calling process during execution of the sleep() function, except as a result of a prior call to the alarm() function, sleep() returns the amount of unslept time.

If a signal-catching function interrupts the sleep() function and examines or changes the time a SIGALRM is scheduled to be generated, the action associated with the SIGALRM signal, or whether the SIGALRM signal is blocked, sleep() returns the amount of unslept time. All caught signals cause sleep() to terminate.

If a signal-catching function interrupts the sleep() function and calls the siglongjmp() or longjmp() function to restore an environment saved prior to the sleep() call, the action associated with the SIGALRM signal and the time at which a SIGALRM signal is scheduled to be generated are as follows:
• The time at which the SIGALRM signal is scheduled is that time set by the signal catching function.

• The action associated with the SIGALRM signal is the action set by the signal catching function.

The `longjmp()` function restores the process’s signal mask as part of the environment.
This chapter describes the process environment for ULTRIX systems.

4.2 User Identification

4.2.3 Get Supplementary Group IDs

4.2.3.2 Description – On ULTRIX systems, the effective group ID of the calling process is included in the returned list of supplementary group IDs.

4.2.4 Get User Name

4.2.4.3 Returns – On ULTRIX systems, the return value from getlogin() points to static data.

On ULTRIX systems, if cuserid() is called with a NULL pointer, the return value from cuserid() points to static data.

On ULTRIX systems, the cuserid() function does not use the getpwnam() function, but does use getpwuid(). Thus, a call to cuserid() will overwrite the data returned by previous calls to getpwnam().

4.2.4.4 Errors – There are no implementation-defined error conditions for the getlogin() and cuserid() functions.

4.4 System Identification

4.4.1 System Name

4.4.1.2 Description – On ULTRIX systems, the struct utsname returned by uname() contains the following members, formats, and (optional) ranges of values:
<table>
<thead>
<tr>
<th>Member Name</th>
<th>Format</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>sysname</td>
<td>string</td>
<td>&quot;ULTRIX&quot;</td>
</tr>
<tr>
<td>nodename</td>
<td>string</td>
<td>The nodename field is initialized upon the first call to <code>uname()</code> from the hostname field</td>
</tr>
<tr>
<td>release</td>
<td>string</td>
<td>&quot;4.0&quot;</td>
</tr>
<tr>
<td>version</td>
<td>string</td>
<td>&quot;0&quot;</td>
</tr>
<tr>
<td>machine</td>
<td>string</td>
<td>&quot;VAX&quot; or &quot;RISC&quot;</td>
</tr>
</tbody>
</table>

4.4.1.4 Errors — On ULTRIX systems, the `uname()` system call fails if the `name` argument points to an invalid address. In such instances, `errno` is set to [EFAULT] when the call returns.

4.5 Time

4.5.1 Get System Time

4.5.1.4 Errors — On ULTRIX systems, for the `time()` function, no error conditions are detected.

On ULTRIX systems, if the `tloc` argument to the `time()` function is not NULL, the `time()` function tries to use it.

4.5.2 Process Times

4.5.2.3 Returns — On ULTRIX systems, the return value for the `times()` function can overflow the possible range of type `clock_t`.

4.5.2.4 Errors — On ULTRIX systems, no error conditions are detected for the `times()` function.

4.6 Environment Variables

4.6.1 Environment Access

4.6.1.3 Returns — The return value for `getenv()` does not point to static data.
4.6.1.4 Errors — There are no implementation-defined error conditions for the `getenv()` function.

4.7 Terminal Identification

4.7.1 Generate Terminal Pathname

4.7.1.3 Returns — If the parameter to `ctermid()` is NULL, the return value is in a static data area.

4.7.1.4 Errors — There are no implementation-defined error conditions for the `ctermid()` function.

4.7.2 Determine Terminal Device Name

4.7.2.2 Description — On ULTRIX systems, the return value of `ttyname()` points to static data.

4.7.2.3 Errors — On ULTRIX systems, there are no implementation-defined error conditions for the `ttyname()` or the `isatty()` function.

4.8 Configurable System Variables

4.8.1 Get Configurable System Variables

4.8.1.2 Description — On ULTRIX systems, additional system configurable variables beyond those listed in Table 4-2 of POSIX.1 are supported. These variables and the values from `<unistd.h>` used as the `name` argument to the `sysconf()` function are shown in the following table:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
<th>name value</th>
</tr>
</thead>
<tbody>
<tr>
<td>{PASS_MAX}</td>
<td>Maximum number of characters in a password (not including terminating null)</td>
<td>_SC_PASS_MAX</td>
</tr>
<tr>
<td>{_XOPEN_VERSION}</td>
<td>Integer value indicating the version of XPG to which the system is compliant</td>
<td>_SC_XOPEN_VERSION</td>
</tr>
</tbody>
</table>
This chapter describes the functions that ULTRIX systems use for file and directory manipulation.

5.1 Directories

5.1.1 Format of Directory Entries

On ULTRIX systems, a directory is represented by a file-like inode. A directory and regular file inode are differentiated by the mode field in the inode. The data portion of a directory points to directory structures. The format of a directory entry is described by the `struct dirent` in the system header file `<sys/dir.h>`. For more information, refer to `dir(5)` in the *ULTRIX Reference Pages*.

The character array `d_name` can contain a maximum of 256 bytes.

5.1.2 Directory Operations

5.1.2.2 Description — On ULTRIX systems, the pointer returned by `readdir()` points to data that is overwritten by a subsequent call to `readdir()` on the same directory stream. Overwriting occurs when the number of entries buffered by `readdir()` is exhausted and the next set of entries is read by the function.

The `readdir()` function can buffer several directory entries for each actual read operation.

On ULTRIX systems, if the `dirp` argument passed to any of the file operation functions does not refer to a currently open directory stream, the function fails and `errno` is set to [EBADF].

On ULTRIX systems, after a call to the `fork()` function, if both the parent and child processes use the `readdir()` or `rewinddir()` function, each effects the other's offset of the directory being operated on through the function calls.

5.1.2.4 Errors — For the `opendir()` function, ULTRIX systems detect the conditions and return the corresponding `errno` values for [EMFILE] and [ENFILE].

For the `readdir()` function, ULTRIX systems detect the conditions and return the corresponding `errno` value for [EBADF].

For the `closedir()` function, ULTRIX systems detect the conditions and return the corresponding `errno` value for [EBADF].
5.2 Working Directory

5.2.2 Working Directory Pathname

5.2.2.2 Description — On ULTRIX systems, if the buf argument to getcwd (buf, siz) is a NULL pointer, the getcwd() directory obtains size bytes of space using malloc(). The pointer returned by getcwd() can be used as the argument to a subsequent call to free().

5.2.2.3 Returns — On ULTRIX systems, after an error on a call to getcwd() is returned, the current contents of buf are undefined.

5.2.2.4 Errors — For the getcwd() function, ULTRIX systems detect the conditions and return the corresponding errno value for [EACCES].

5.3 General File Creation

5.3.1 Open a File

5.3.1.2 Description — On ULTRIX systems, if the O_RDWR flag is set in oflag and open (path, oflag, mode) is called to open a FIFO, the file descriptor returned by open() can be used for reading and writing bytes to and from the FIFO.

On ULTRIX systems, when the O_CREAT flag is set in oflag and when bits in mode other than the file permission bits are set, the file is created with the mode bits that remain after the mode argument is masked against the process’s file creation mask.

On ULTRIX systems, if the O_EXCL bit is set and O_CREAT is not set, the O_EXCL bit is silently ignored and the open() is attempted. If the file does not exist, it will not be created, resulting in open() returning an error.

On ULTRIX systems, the O_NONBLOCK flag in the path parameter is ignored unless the path parameter refers to a block or character special file or the path parameter refers to a FIFO special file and the O_RDONLY or the O_WRONLY flag is also set in oflag.

If the O_TRUNC bit is set in oflag and path resolves to a socket or character or block special file, the result is an access check for write privileges.

On ULTRIX systems, using O_TRUNC with O_RDONLY has the following effect: if the process has write access, the file is truncated to a size of zero.

5.3.3 Set File Creation Mask

5.3.3.2 Description — On ULTRIX systems, if you specify invalid permission bits to the umask() function, it sets the file creation mask of the process to valid file permission bits by ignoring the invalid bits. The valid bits are defined to be those corresponding to the bitwise inclusive OR of S_IRWXU, S_IRWXG, and S_IRWXO.
5.3.4 Link to a File

5.3.4.2 Description – On ULTRIX systems, links across file systems are not valid. If you attempt such an operation, the link system call returns a value of -1 with *errno* set to [EXDEV].

On ULTRIX systems, the linking of directories is not supported through the *link()* function.

The calling process does not require read, write, or execute permission on the file being linked. However, the calling process must have search permission on the parent directory of the file.

5.4 Special File Creation

5.4.1 Make a Directory

5.4.1.2 Description – On ULTRIX systems, the *mode* argument to *mkdir(path, mode)* is masked against the process’s file creation mask set by *umask()*.

This masking results in the directory being created with a valid mode.

5.4.2 Make a FIFO Special File

5.4.2.2 Description – On ULTRIX systems, the *mode* argument to the *mknod()* function is masked against the process’s file creation mask set by *umask()*.

This results in a FIFO special file being created with a valid mode.

5.5 File Removal

5.5.2 Remove a Directory

5.5.2.2 Description – On ULTRIX systems, when the root directory or the current working directory of a process is removed, the process holds a pointer to its current working directory. Because ULTRIX systems do not release the space allocated to a directory until all references are gone, there is no effect for the process when the root or current working directory is removed. Other processes will not be able to access the directory, however.

On ULTRIX systems, a directory can be removed only if it is empty.

5.5.3 Rename a File

5.5.3.2 Description – On ULTRIX systems, in a call to *rename(old, new)*, if the *old* argument points to the pathname of a directory, write access permission is not required for the directory named by *old*; if the directory named by *new* exists, write permission for the directory is not required.
5.6 File Characteristics

5.6.1 File Characteristics: Header and Data Structure

5.6.1.2 The <sys/stat.h> File Modes — On ULTRIX systems, there are no additional bits logically ORed with S_IRWXU, S_IRWXG, and S_IRWXO.

5.6.1.3 The <sys/stat.h> Time Entries — ULTRIX systems do not change the time-related fields of struct stat by any functions other than those described in POSIX.1.

5.6.2 Get File Status

5.6.2.2 Description — If the access control methods described in POSIX.1 are honored, ULTRIX systems cannot deny the existence of the file.

If the file does not exist, errno is set to [ENOENT] when the call returns.

5.6.3 File Accessibility

5.6.3.2 Description — If a process has appropriate privileges, access (path, X_OK) returns a zero whether or not execute permission bits are set for path.

5.6.3.4 Errors — For the access() function, ULTRIX systems detect the conditions and return the errno value for [EINVAL].

5.6.4 Change File Modes

5.6.4.2 Description — On ULTRIX systems, the S_ISUID and S_ISGID bits on file systems that have been mounted with the nosuid option set are ignored.

On ULTRIX systems, changes made through the chmod() function do not effect file descriptions for files that are currently open.

5.6.5 Change Owner and Group of a File

5.6.5.2 Description — On ULTRIX systems, calls to chown() from a process with appropriate privileges do not clear the S_ISUID and S_ISGID bits.

5.6.5.4 Errors — For the chown() function, ULTRIX systems detect the conditions and return the errno value for [EINVAL].
5.7 Configurable Pathname Variables

5.7.1 Get Configurable Pathname Variables

5.7.1.2 Description – ULTRIX systems do not support any additional variables, beyond those listed in Table 5-2 of POSIX.1, that can be queried by the pathconf() and fpathconf() functions.

ULTRIX systems do not need to use the path or fildes arguments. Therefore, the return value is not effected by the type of file the argument refers to.

5.7.1.4 Errors – For the pathconf() function, ULTRIX systems detect the conditions and return the corresponding errno values for [EACCES], [EINVAL], [ENAMETOOLONG], [ENOENT], and [ENOTDIR].

For the fpathconf() function, ULTRIX systems detect the conditions and return the corresponding errno values for [EBADF] and [EINVAL].
This chapter provides a brief explanation of the input and output functions and the control operations of ULTRIX systems.

6.4 Input and Output

6.4.1 Read from a File

6.4.1.2 Description – For a device special file, ULTRIX systems handle subsequent read operations in the same way they handle read operations to a file.

If a read to any kind of file is interrupted after a data transfer and some data was successfully read before the interrupt, the read() function returns the number of bytes read before the interrupt.

If the value of nbyte is greater than {INT_MAX}, a zero is returned.

If the device is incapable of seeking (for example, a terminal device), the concept of a file offset is not applicable. Subsequent reads will start from the current position in the file.

A read fails with errno set to [EINVAL] if the count is greater than {INT_MAX}.

6.4.1.4 Errors – On ULTRIX systems, an [EIO] error is generated if an I/O error occurs while reading from the file system.

6.4.2 Write to a File

6.4.2.2 Description – On ULTRIX systems, if a write of zero bytes to a file that is not a regular file is attempted, the system ignores the write.

If a write to any kind of file is interrupted after a data transfer and the data was successfully written before the interrupt, the write() function returns the number of bytes written before the interrupt.

A write fails with errno set to [EINVAL] if nbyte is greater than {INT_MAX}.

6.4.2.4 Errors – On ULTRIX systems, the maximum file size is \(2^{64}-1\) bytes. If the size of a file exceeds this limit, write() returns a value of -1 with errno set to [EBIG].
6.5 Control Operations on Files

6.5.2 File Control

6.5.2.2 Description – ULTRIX systems support additional file status flags beyond those listed in POSIX.1 Table 6-5. See fcntl(2) in the ULTRIX Reference Pages for a complete list of the file status flags. If the fcntl() function is called with the cmd argument set to F_SETFL and if any bits are set in arg other than those listed in the fcntl(2) reference page, the invalid file status bits are ignored and the file status flag is set to the valid flags passed through arg.

ULTRIX systems support advisory locking on regular, special device, and socket file types.

ULTRIX systems support a negative l_len member of struct flock. The user must already own the lock. The fcntl() function subtracts the absolute value of l_len from the previous start and sets the length of the lock to the absolute value of l_len.

6.5.2.4 Errors – For the fcntl() function, ULTRIX systems detect the conditions and return the errno value for [EDEADLK].

6.5.3 Reposition Read/Write File Offset

6.5.3.2 Description – On ULTRIX systems, when the lseek() function is performed on a file that is incapable of seeking, three possible events can occur:

- The function returns a file offset that is never used.
- The function returns a value of -1 with errno set to [ESPIPE] if the file descriptor describes a pipe.
- The function returns -1 with errno set to [EBADF] if the file descriptor describes a file in the file system.
This chapter describes the interface characteristics and control functions of ULTRIX systems.

7.1 General Terminal Interface

ULTRIX systems support asynchronous terminals and network terminals, using the General Terminal Interface defined in POSIX.1, through the use of pseudoterminals. Synchronous ports are not supported by the same interface. For a complete description of the termios interface, refer to termios(4) in the ULTRIX Reference Pages.

7.1.1 Interface Characteristics

7.1.1.3 The Controlling Terminal – On ULTRIX systems, if a session leader has no controlling terminal and opens a terminal device file that is not already associated with a session without using the open function's O_NOCTTY option, the terminal becomes the controlling terminal of the session leader.

7.1.1.5 Input Processing and Reading Data – ULTRIX systems impose the limit MAX_INPUT on the number of bytes that can be stored in the input queue. When ICANON is not set by an application and if the character input buffers of the system become full, the system flushes the input buffer without notice. This action causes all the characters in the input queue to be lost. If an application sets ICANON and the character input buffers of the system become full, the driver discards additional characters and echoes a bell (ASCII BEL) to notify the user of the full condition.

7.1.1.6 Canonical Mode Input Processing – When there are MAX_CANON characters in an input line, additional characters are discarded.

7.1.1.7 Non-Canonical Mode Input Processing – On ULTRIX systems, the value of MIN cannot exceed MAX_INPUT because the value is placed in an unsigned character type and MAX_INPUT is equal to UCHAR_MAX.

7.1.1.8 Writing Data and Output Processing – On ULTRIX systems, the terminal interface provides a buffering mechanism. For example, when a call to write() completes, all of the characters written have been scheduled for transmission to the device, but the transmission is not necessarily complete. The characters are transmitted to the device as soon as previously written characters have been output successfully.
7.1.9 **Special Characters** – On ULTRIX systems, the values for the START and STOP characters can be set by the user. There are no multibyte sequences with different meanings from the meaning of the bytes when considered individually.

ULTRIX systems define the following single-byte functions:

<table>
<thead>
<tr>
<th>Function</th>
<th>Character</th>
<th>Effect</th>
</tr>
</thead>
<tbody>
<tr>
<td>VLNEXT</td>
<td>CTRL/V</td>
<td>Next character is literal</td>
</tr>
<tr>
<td>VFLUSH</td>
<td>CTRL/O</td>
<td>Toggle output discarding</td>
</tr>
<tr>
<td>VWERASE</td>
<td>CTRL/W</td>
<td>Erase previous word</td>
</tr>
<tr>
<td>VRPRNT</td>
<td>CTRL/R</td>
<td>Reprint line</td>
</tr>
<tr>
<td>VDSUSP</td>
<td>CTRL/Y</td>
<td>Delayed suspend character</td>
</tr>
</tbody>
</table>

7.1.2 **Settable Parameters**

7.1.2.1 **The termios Structure** – On ULTRIX systems, the `termios` structure is defined in the header `<termios.h>`. The members of this structure are shown as follows:

```c
struct termios {
    tcflag_t c_iflag /* input modes */
    tcflag_t c_oflag /* output modes */
    tcflag_t c_cflag /* control modes */
    tcflag_t c_lflag /* local modes */
    cc_t c_cc[NCCS] /* control char */
    cc_t c_line;    /* line discipline */
};
```

The types `tcflag_t` (unsigned long) and `cc_t` (unsigned char) are defined in the header `<termios.h>`. The value of NCCS is 19 and is defined in `<termios.h>`.

The total size of the `termios` structure is 36 bytes.

7.1.2.2 **Input Modes** – When an application sets ICANON, if the number of characters input exceeds (MAX_INPUT/2) and if the number of characters in the canonical input queue exceeds (MAX_INPUT/3), the system transmits a STOP character. The system transmits a START character when the number of characters in the canonical input queue drops below (MAX_INPUT/5).

If an application does not set ICANON and if the number of input characters exceeds (MAX_INPUT/2), the system transmits a STOP character. The system transmits a START character when the number of characters in the input queue drops below (MAX_INPUT/5).

After an application opens a terminal line, the initial input control value is zero (all settings off).

It is not possible to generate a break condition on network pseudoterminals.

7.1.2.3 **Output Modes** – After an application opens a terminal line, the initial output control value is zero (all settings off).
Additional output control values have been defined. These attributes are used when OPOST is set. For a complete description of output modes, refer to termios(4) in the ULTRIX Reference Pages.

7.1.2.4 Control Modes – The terminal subsystem of ULTRIX systems only supports asynchronous terminals.

On ULTRIX systems, the initial hardware control values after open() are CS8, CREAD, B300, and HUPCL.

7.1.2.5 Local Modes – If ECHOE and ICANON are set and the ERASE character attempts to erase when there is no character to erase, ULTRIX systems do not echo anything.

If ECHOK and ICANON are set, the KILL character echoes a newline (\n) character after the KILL character. If ECHOE and ICANON are turned on and there are no characters to erase, a newline is echoed. If there are characters to erase from the line and ECHOE and ICANON are set, the characters are erased from the display.

The IEXTEN flag enables additional special control characters.

IEXTEN being set interacts with ICANON, ISIG, IXON, or IXOFF.

After an application opens a terminal line, the initial local control value is zero (all settings off). Additional local control values have been defined. For a complete list of local modes and their interactions with other values, refer to termios(4) in the ULTRIX Reference Pages.

7.1.2.6 Special Control Characters – The number of elements in the c_cc array, NCCS, is 19.

On ULTRIX systems, additional special control characters have been defined. These special control characters are used only when the IEXTEN flag is set. For a complete list of special control characters, refer to termios(4) in the ULTRIX Reference Pages.

ULTRIX systems support setting of the START and STOP characters.
The following table shows the octal default values for the special characters:

<table>
<thead>
<tr>
<th>Special Character</th>
<th>Octal Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>VINTR</td>
<td>177</td>
</tr>
<tr>
<td>VQUIT</td>
<td>034</td>
</tr>
<tr>
<td>VERASE</td>
<td>043</td>
</tr>
<tr>
<td>VKILL</td>
<td>100</td>
</tr>
<tr>
<td>VEOF</td>
<td>004</td>
</tr>
<tr>
<td>VEOL</td>
<td>377</td>
</tr>
<tr>
<td>VMIN</td>
<td>006</td>
</tr>
<tr>
<td>VTIME</td>
<td>001</td>
</tr>
<tr>
<td>VSTART</td>
<td>021</td>
</tr>
<tr>
<td>VSTOP</td>
<td>023</td>
</tr>
<tr>
<td>VSUSP</td>
<td>032</td>
</tr>
</tbody>
</table>

7.1.2.7 Baud Rate Functions

7.1.2.7.2 Description – On ULTRIX systems, attempts to set unsupported baud rates are ignored. The functions cfsetispeed(), cfsetospeed(), and tcsetattr() do not return errors in response to unsupported baud rates. It is possible to set input and output baud rates to different values.

7.1.2.7.4 Errors – For the functions cfsetispeed() and cfsetospeed(), ULTRIX systems do not detect error conditions.

A call to tcgetattr(), then a call to cfgetispeed() or cfgetospeed() should be made to verify that the baud rate was set after a call to tcsetattr().

7.2 General Terminal Interface Control Functions

7.2.2 Line Control Functions

7.2.2.2 Description – ULTRIX systems support the following types of terminal devices:
- Asynchronous
- Pseudoterminal
- LAT (local area transport protocol)

On ULTRIX systems, if the duration passed to the tcsendbreak() function is greater than zero, zero-valued bits will be transmitted for (duration*0.01) seconds.

If the object referred to by fildes is a pseudoterminal, no break is generated.
7.2.3 Get Foreground Process Group ID

7.2.3.2 Description – ULTRIX systems define _POSIX_JOB_CONTROL. The tcgetpgrp() function is supported.

7.2.4 Set Foreground Process Group ID

7.2.4.2 Description – ULTRIX systems define _POSIX_JOB_CONTROL. The tcsetpgrp() function is supported.
This chapter provides C programming language conformance statements and describes language specific services for ULTRIX systems.

Conformance

Implementation Conformance

ULTRIX systems conform to C Language Binding (Common Usage C Language Dependent System Support) on all systems using the default C compiler. In order to select the POSIX versions of the library routines, compilation and linking commands must either specify the -YPOSIX option, or the environment variable PROG_ENV must have the value POSIX with no conflicting -y option. For additional information, refer to the Guide to Languages and Programming.

8.1 Referenced C Language Routines

All functions listed in Section 8.1 are implemented as specified by the C Standard.

8.1.1 Extensions to Time Functions

On ULTRIX systems, the environment variable TZ uses the format in which the first character is a colon (:) and the characters that follow the colon are interpreted as the pathname of a tzfile() format file. This file provides the time conversion information. If the pathname begins with a slash (/), it represents an absolute pathname; otherwise the pathname is relative to the system time conversion information directory /etc/zoneinfo. For further information, see ctime(3) in the ULTRIX Reference Pages.

8.1.2 Extensions to the setlocale Function

8.1.2.2 Description – On ULTRIX systems, the default locale is based on 7-bit US-ASCII with strings in American English and date formats according to the American definition. Called the C locale, it is the system default locale for the following environment variables:
<table>
<thead>
<tr>
<th>Environment Variable</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>LC_TYPE</td>
<td>C</td>
</tr>
<tr>
<td>LC_COLLATE</td>
<td>C</td>
</tr>
<tr>
<td>LC_TIME</td>
<td>C</td>
</tr>
<tr>
<td>LC_NUMERIC</td>
<td>C</td>
</tr>
<tr>
<td>LC_MONETARY</td>
<td>C</td>
</tr>
<tr>
<td>LANG</td>
<td>C</td>
</tr>
</tbody>
</table>

On ULTRIX systems, if the specified *LC_* environment variable is not set or if it is set to the empty string, setlocale () examines the *LANG* environment variable. If *LANG* is set to the name of a valid locale, that value is used at runtime to set the program locale. Otherwise, setlocale () returns a NULL pointer and does not change the locale. For further information, refer to the *Guide to Developing International Software*. For further information on setlocale(3int), refer to the *ULTRIX Reference Pages*.

8.2 FILE-Type C Language Functions

8.2.1 Map a Stream Pointer to a File Descriptor

8.2.1.4 Errors — On ULTRIX systems, there are no implementation-defined error conditions for the fileno () function.

8.2.2 Open a Stream on a File Descriptor

8.2.2.2 Description — On ULTRIX systems, the *type* argument to the fdopen () function contains the following additional values:

- "A" Synonym for "a"
- "A+" Synonym for "a+

8.2.2.4 Errors — There are no implementation-defined error conditions for the fdopen () function.

8.3 Other C Language Functions

8.3.2 Set Time Zone
8.3.2.2 Description — On ULTRIX systems, if the environment variable TZ is absent, the file /etc/zoneinfo/localtine is used to obtain time conversion information. If retrieving the information from this file fails for any reason, the Greenwich Mean Time (GMT) offset as provided by the kernel is used. For further information, see ctime(3) in the ULTRIX Reference Pages.
This chapter provides information on system databases and database access for ULTRIX systems.

9.1 System Databases

On ULTRIX systems, if the home directory field in the user database is null, the root directory (/) is used as the home directory.

9.2 Database Access

9.2.1 Group Database Access

9.2.1.2 Description – The getgrgid() and getgrnam() functions both return a pointer to an object of type struct group that contains an entry from the group database with a matching gid or name. This structure includes the members shown in the following table. In addition to the group database fields specified in Section 9.1 of POSIX.1, the group database on ULTRIX systems contains the gr_passwd field (encrypted password).

<table>
<thead>
<tr>
<th>Member Type</th>
<th>Member Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>char *</td>
<td>gr_name</td>
<td>The name of the group</td>
</tr>
<tr>
<td>char *</td>
<td>gr_passwd</td>
<td>The encrypted password for the</td>
</tr>
<tr>
<td></td>
<td></td>
<td>group</td>
</tr>
<tr>
<td>gid_t</td>
<td>gr_gid</td>
<td>The numerical group ID</td>
</tr>
<tr>
<td>char **</td>
<td>gr_mem</td>
<td>A null-terminated vector of</td>
</tr>
<tr>
<td></td>
<td></td>
<td>pointers to the individual</td>
</tr>
<tr>
<td></td>
<td></td>
<td>member names</td>
</tr>
</tbody>
</table>

9.2.1.3 Returns – The return value of the getgrgid() function points to a static data area and will be overwritten by subsequent calls. This static area is also used by the functions getgrent() and getgrnam(). A call to any one of these functions overwrites any data contained in the static area.
9.2.1.4 **Errors** – There are no implementation-defined error conditions for the `getgrgid()` or the `getgrnam()` function.

9.2.2 **User Database Access**

9.2.2.2 **Description** – On ULTRIX systems, the `getpwuid()` and `getpwnam()` functions both return a pointer to an object of type `struct passwd` containing an entry from the user database with a matching `uid` or `name`.

This structure includes the members shown in the following table. In addition to the user database fields specified in Section 9.1 of POSIX.1, the ULTRIX user database contains three additional fields: `pw_quota`, `pw_comment`, and `pw_gecos`.

<table>
<thead>
<tr>
<th>Member Type</th>
<th>Member Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>char *</td>
<td><code>pw_name</code></td>
<td>User’s login name</td>
</tr>
<tr>
<td>char *</td>
<td><code>pw_passwd</code></td>
<td>User’s encrypted password</td>
</tr>
<tr>
<td><code>uid_t</code></td>
<td><code>pw_uid</code></td>
<td>User ID number</td>
</tr>
<tr>
<td><code>gid_t</code></td>
<td><code>pw_gid</code></td>
<td>Group ID number</td>
</tr>
<tr>
<td>int</td>
<td><code>pw_quota</code></td>
<td>Unused</td>
</tr>
<tr>
<td>char *</td>
<td><code>pw_comment</code></td>
<td>Unused</td>
</tr>
<tr>
<td>char *</td>
<td><code>pw_gecos</code></td>
<td>User description</td>
</tr>
<tr>
<td>char *</td>
<td><code>pw_dir</code></td>
<td>Initial working directory</td>
</tr>
<tr>
<td>char *</td>
<td><code>pw_shell</code></td>
<td>Initial user program</td>
</tr>
</tbody>
</table>

The `cuserid()` function uses the `getpwuid()` function and overwrites data in the static area on subsequent calls.

9.2.2.3 **Returns** – The `getpwuid()` and `getpwnam()` functions return values to the static data area and overwrite data in this area on each subsequent call.

9.2.2.4 **Errors** – There are no implementation-defined error conditions for the `getpwuid()` or `getpwnam()` function.
This chapter contains information on the tar and cpio file formats on ULTRIX systems.

10.1 Archive/Interchange File Format

On ULTRIX systems, the format-reading and format-creating utilities are named tar and cpio. For a description of these utilities and the interfaces to them, refer to tar(1) and cpio(1) in the ULTRIX Reference Pages.

10.1.1 Extended tar Format

On ULTRIX systems, the tar utility by default groups twenty 512-byte blocks into a record for physical I/O operations.

On ULTRIX systems, the tar utility by default writes a group of twenty 512-byte blocks in a single write() operation.

In the typeflag field of the header used by the tar utility, the ASCII digit ’7’ is reserved to indicate some high-performance attribute.

On ULTRIX systems, if a file name is found on the medium that would create an invalid file name on the system, the data from the file is not stored on the file hierarchy. The tar utility ignores such files and produces an error message ("Can’t create...") indicating that the file is being ignored.

On ULTRIX systems, the tar utility terminates if mode bits that are not defined in POSIX.1 are encountered in the mode field.

On ULTRIX systems, if the typeflag field is CHARTYPE, BLKTYPE, or FIFOTYPE, the size field is set to zero and is not used.

On ULTRIX systems, all characters are valid for use in file names. For character and block special file types, the devmajor and devminor fields contain the major and minor device numbers as used by the local system.

10.1.2 Extended cpio Format

On ULTRIX systems, the cpio utility writes a series of fixed size blocks of bytes in a POSIX conformant manner.

10.1.2.1 Header – On ULTRIX systems, the cpio header values c_dev, c_ino, and c_rdev for a particular file are obtained from the information returned by the stat system call in the fields st_dev, st_ino, and st_rdev, respectively.

For character or block special files, c_rdev contains the major and minor numbers of the device the file describes.
10.1.2.2 **File Name** – On ULTRIX systems, the `cpio` utility supports the use of the POSIX portable filename character set.

When an ULTRIX system finds a file name on a medium that would create an invalid file name, it issues an error message ("Can't create ..."), skips the file, and processes the next file in the archive.

10.1.2.4 **Special Entries** – On ULTRIX systems, the `c_filesize` parameter is not defined for device special files. For other special files (FIFOs, sockets, directories, and so on), `c_filesize` is set to the size of the archived file.

10.1.2.5 **The `cpio` Values** – On ULTRIX systems, the `cpio` command supports some special file types as defined by POSIX.1. These include sockets (C_ISSOCK), links (C_ISLNK), character special files, and block special files.

POSIX.1 reserves C_ISVTX, C_ISCTG, C_ISLNK, and C_ISSOCK to retain compatibility with some existing implementations.

On ULTRIX system, the `cpio` utility ignores `mode` flags in the archive that are not mentioned in POSIX.1.

10.1.3 **Multiple Volumes**

On ULTRIX systems, the `cpio` command supports archive save sets that span multiple volumes. The user is prompted to mount the next volume of an archive when `cpio` determines the next volume is needed. The next volume must be mounted on the same physical device as the initial volume. That is, if initially reading from `/dev/rmt0h`, all subsequent volumes must be mounted on `/dev/rmt0h`. 
The ULTRIX Version 4.0 Operating System meets the requirements specified by the POSIX Federal Information Processing Standard (FIPS 151-1). This appendix is a reproduction of these requirements. Note that the base document for FIPS 151-1 is IEEE Std 1003.1-1988.

The following modifications to IEEE Std 1003.1-1988, *IEEE Standard Portable Operating System Interface for Computer Environments*, are required for implementations of POSIX that are acquired by Federal agencies:

- Inconsistencies with CLK_TCK exist between the IEEE Std 1003.1-1988 and the referenced ANSI/X3.159-1989 Programming Language C Standard draft 13 May 1988 (X3J11/88-002). This inconsistency shall be resolved in the ratified C Standard. Until the C Standard is ratified, CLK_TCK is to be treated as a POSIX-only symbol.

- The implementation shall support the option \_POSIX_CHOWN_RESTRICTED.

- The implementation shall support the option \{NGROUPS_MAX\} such that the value of \{NGROUPS_MAX\} is greater than or equal to eight (8).

- The implementation shall support the setting of the group-ID of a file (when it is created) to that of its parent directory.

- The implementation shall support the functionaity associated with the feature \_POSIX_SAVED_IDS.

- The implementation shall support the functionality associated with the feature \_POSIX_VDISABLE.

- The implementation shall support the option \_POSIX_JOB_CONTROL.

- The implementation shall support the functionality associated with the feature \_POSIX_NO_TRUNC.

- In section 6.4.1.2, the sentence “If a read() is interrupted by a signal after it has successfully read some data, either it shall return -1 with errno set to [EINTR], or it shall return the number of bytes read.” shall be deleted and replaced with the sentence “If a read() is interrupted by a signal after it has successfully read some data, it shall return the number of bytes the system has read.”

- In section 6.4.2.2, the sentence “If a write() is interrupted by a signal after it successfully writes some data, either it shall return -1 with errno set to [EINTR], or it shall return the number of bytes written.” shall be deleted and replaced with the sentence “If a write() is interrupted by a signal after it successfully writes some data, it shall return the number of bytes the system has written.”

- The environment for the login shell shall contain the environment variables HOME and LOGNAME as defined in section 2.7.
How to Order Additional Documentation

Technical Support
If you need help deciding which documentation best meets your needs, call 800-343-4040 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-baud 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

<table>
<thead>
<tr>
<th>Your Location</th>
<th>Call</th>
<th>Contact</th>
</tr>
</thead>
<tbody>
<tr>
<td>Continental USA,</td>
<td>800-DIGITAL</td>
<td>Digital Equipment Corporation</td>
</tr>
<tr>
<td>Alaska, or Hawaii</td>
<td></td>
<td>P.O. Box CS2008</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Nashua, New Hampshire 03061</td>
</tr>
<tr>
<td>Puerto Rico</td>
<td>809-754-7575</td>
<td>Local Digital Subsidiary</td>
</tr>
<tr>
<td>Canada</td>
<td>800-267-6215</td>
<td>Digital Equipment of Canada</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Attn: DECdirect Operations KAO2/2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>P.O. Box 13000</td>
</tr>
<tr>
<td></td>
<td></td>
<td>100 Herzberg Road</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Kanata, Ontario, Canada K2K 2A6</td>
</tr>
<tr>
<td>International</td>
<td></td>
<td>Local Digital subsidiary or approved distributor</td>
</tr>
<tr>
<td>Internal*</td>
<td></td>
<td>SSB Order Processing - WMO/E15</td>
</tr>
<tr>
<td></td>
<td></td>
<td>or Software Supply Business</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Digital Equipment Corporation</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Westminster, Massachusetts 01473</td>
</tr>
</tbody>
</table>

* For internal orders, you must submit an Internal Software Order Form (EN-01740-07).
**Reader's Comments**

Please use this postage-paid form to comment on this manual. If you require a written reply to a software problem and are eligible to receive one under Software Performance Report (SPR) service, submit your comments on an SPR form.

Thank you for your assistance.

Please rate this manual:

<table>
<thead>
<tr>
<th></th>
<th>Excellent</th>
<th>Good</th>
<th>Fair</th>
<th>Poor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy (software works as manual says)</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td>Completeness (enough information)</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td>Clarity (easy to understand)</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td>Organization (structure of subject matter)</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td>Figures (useful)</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td>Examples (useful)</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td>Index (ability to find topic)</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td>Page layout (easy to find information)</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
</tr>
</tbody>
</table>

What would you like to see more/less of? ____________________________________________

What do you like best about this manual? ____________________________________________

What do you like least about this manual? ____________________________________________

Please list errors you have found in this manual:

<table>
<thead>
<tr>
<th>Page</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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 ____________________
BUSINESS REPLY MAIL
FIRST-CLASS MAIL PERMIT NO. 33 MAYNARD MA

POSTAGE WILL BE PAID BY ADDRESSEE

DIGITAL EQUIPMENT CORPORATION
OPEN SOFTWARE PUBLICATIONS MANAGER
ZK03-2/Z04
110 SPIT BROOK ROAD
NASHUA NH 03062-9987
Please use this postage-paid form to comment on this manual. If you require a written reply to a software problem and are eligible to receive one under Software Performance Report (SPR) service, submit your comments on an SPR form.

Thank you for your assistance.

Please rate this manual:

<table>
<thead>
<tr>
<th></th>
<th>Excellent</th>
<th>Good</th>
<th>Fair</th>
<th>Poor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy (software works as manual says)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Completeness (enough information)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Clarity (easy to understand)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Organization (structure of subject matter)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Figures (useful)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Examples (useful)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Index (ability to find topic)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Page layout (easy to find information)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

What would you like to see more/less of? ______________________________________

What do you like best about this manual? ______________________________________

What do you like least about this manual? ______________________________________

Please list errors you have found in this manual:

<table>
<thead>
<tr>
<th>Page</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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 __________