CONVEX POSIX Conformance

Third Edition

CONVEX COMPUTER CORPORATION

CONVEX Computer Corporation
3000 Waterview Parkway
P.O. Box 833851
Richardson, TX 75083-3851
United States of America
(214)497-4000

CONVEX POSIX Conformance

Order No. DSW-311

Third Edition
July 1992

CONVEX Press
Richardson, Texas
United States of America

CONVEX
POSIX Conformance

Order No. DSW-311

Printed in the United States of America

Revision Information for

CONVEX POSIX Conformance

Edition	Document No.	Description
Third	710-002030-202	Released with ConvexOS V10.1 in July 1992 as part of the NIST POSIX certification process. (ConvexOS 10.1 has been certified to conform to FIPS 151-1.)
Second	710-002030-201	Released with ConvexOS V10.0 in November 1991. Updated for ISO/IEC 9945-1:1990, IEEE Std 1003.1-1990.
First	710-002030-200	Initially released with ConvexOS V8.0 in December 1989 for IEEE Std 1003.1-1988.

Contents

2 Terminology and General Requirements

3 Process Primitives

4 Process Environment

5 Files and Directories

6 Input and Output Primitives

Tables

Preface

Purpose and audience

This document defines how CONVEX Computer Corporation interprets the Portable Operating System Interface for Computing Environments as detailed in IEEE Std 1003.1-1990 (POSIX.1). ConvexOS is an implementation of the Berkeley BSD operating system containing POSIX.1 functionality with extensions for supercomputer environments.

The *CONVEX POSIX Conformance* book is intended for customers analyzing ConvexOS to determine how it relates to IEEE Std 1003.1-1990 and those individuals evaluating our implementation for FIPS-151-1 certification.

This document is required by anyone claiming compliance with the IEEE Std 1003.1-1990 and wanting FIPS-151-1 certification.

CONVEX will keep its operating system current with all versions of POSIX as each POSIX working committee completes and ratifies its group of standards. Interested customers are advised to monitor the progress of POSIX.1 through participation in IEEE or reading the comp.std.unix network newsgroup.

Organization

This book's structure follows that of the POSIX.1 standard, does not contain any information about extended facilities, describes the contents of <limits.h> and <unistd.h>, and discusses all implementation-defined features.

There is a direct correlation in chapter titles and section headings between this document and the POSIX.1 standard. Where no implementation-defined information is needed, sections have been purposely left blank. Users of this document can place it next to the POSIX.1 standard and compare the two, section by section.

The following subsections are common to many of the sections in the POSIX.1 standard and are not listed in this book:

- Synopsis
- Description
- Returns
- Errors
- Cross-References

Notational conventions

The following typefaces have special meaning and are used in this book:

`Courier`	Identifies input and output, including: • Command names • System calls • Data structures and types • Error messages
Italic	Identifies: • User-supplied variables in a command-line example • New and important terms • Titles of documents
KEYCAP	Indicates keyboard keys to be pressed. For example, **RETURN** refers to the carriage return key. Two **KEYCAP** terms separated by a hyphen indicate two keys that you must press simultaneously. For example, **CTRL-d** indicates that you must press the **d** key while holding down the **CTRL** key.

Associated documents

Using ConvexOS software successfully may require information not within the scope of this guide. For more information, consult the *ConvexOS Extensions User's Guide*, DSW-053, "POSIX Concepts" section; it contains details on POSIX.1 functions, ConvexOS-specific functions, and how CONVEX handles on-going compliance with IEEE standards.

Ordering documentation

To order the current edition of this or any other CONVEX document, send requests to:

CONVEX Computer Corporation
Customer Service
P.O. Box 833851
Richardson TX 75083-3851 USA

Include the order number or the exact title, as listed on the front cover.

Technical assistance

If you have questions that are not answered in this book, contact the CONVEX Technical Assistance Center (TAC) at the following locations:

- Within the continental U.S., call 1(800)952-0379.
- From Canada, call 1(800)345-2384
- All other locations, contact local CONVEX office.

The `contact` utility

The CONVEX Technical Assistance Center (TAC) recommends using the `contact` utility to report a hardware, software, or documentation problem. The `contact` utility is an interactive program that helps the TAC track reports and route them to the CONVEX personnel most qualified to fix a problem.

After you invoke `contact`, it prompts you for information about the problem. When you finish your report, `contact` mails it to the TAC electronically. The TAC notifies you within 48 hours that your report has been received.

To use `contact` requires:

- UNIX-to-UNIX Communication Protocol (UUCP) connection to the TAC.
- Full path name of the program or utility in question.
- Version number of the program or utility in question.

Refer to the contact(1) man page for complete details.

General 1

1.1 Scope

ConvexOS fully conforms to the Portable Operating System Interface for Computer Environments (POSIX) IEEE Standard 1003.1-1990 ratified on September 28, 1990.

The CONVEX implementation meets all of the following criteria (as stated in the IEEE standard):

1. The system shall support all required interfaces defined within this part of ISO/IEC 9945. These interfaces shall support the functional behavior described herein.
2. The system may provide additional functions or facilities not required by this part of ISO/IEC 9945. Nonstandard extensions should be identified as such in the system documentation. Nonstandard extensions, when used, may change the behavior of functions or facilities defined by this part of ISO/IEC 9945. The conformance document shall define an environment in which an application can be run with the behavior specified by the standard. In no case shall such an environment require modification of a Strictly Conforming POSIX.1 Application.

This manual provides information specified by Section 1.3.1.2 in POSIX.1 requiring that a document be created "for an implementation claiming conformance."

1.2 Normative References

1.3 Conformance

1.3.1 Implementation Conformance

1.3.1.1 Requirements

1.3.1.2 Documentation

1.3.1.3 Conforming Implementation Options

1.3.2 Application Conformance

1.3.2.1 Strictly Conforming POSIX.1 Application

1.3.2.2 Conforming POSIX.1 Application

1.3.2.2.1 ISO/IEC Conforming POSIX.1 Application

1.3.2.2.2 <National Body> Conforming POSIX.1 Application

1.3.2.3 Conforming POSIX.1 Application Using Extensions

1.3.3 Language-Dependent Services for the C Programming Language

1.3.3.1 Types of Conformance

1.3.3.2 C Standard Language-Dependent System Support

1.3.3.3 Common-Usage C Language-Dependent System Support

1.3.4 Other C Language-Related Specifications

1.3.5 Other Language-Related Specifications

Terminology and General Requirements

2

2.1 Conventions

The conventions used in this document are the same as in the POSIX.1 standard with the exception of the use of typefaces. The CONVEX notational conventions are explained in this book's Preface.

2.2 Definitions

2.2.1 Terminology

2.2.1.1 conformance document:

2.2.1.2 implementation defined:

2.2.1.3 may:

2.2.1.4 obsolescent:

2.2.1.5 shall:

2.2.1.6 should:

2.2.1.7 supported:

2.2.1.8 system documentation:

2.2.1.9 undefined:

2.2.1.10 unspecified

2.2.2 General Terms

For ConvexOS, implementation-defined general terms are defined in the following way:

2.2.2.1 absolute pathname:

2.2.2.2 access mode:

2.2.2.3 address space:

2.2.2.4 appropriate privileges:

Indicates running as root (UID == 0).

2.2.2.5 background process:

2.2.2.6 background process group:

2.2.2.7 block special file:

2.2.2.8 character:

2.2.2.9 character special file:

2.2.2.10 child process:

2.2.2.11 clock tick:

2.2.2.12 controlling process:

2.2.2.13 controlling terminal:

2.2.2.14 current working directory:

2.2.2.15 device:

2.2.2.16 directory:

2.2.2.17 directory entry (link):

2.2.2.18 dot:

2.2.2.19 dot-dot:

2.2.2.20 effective group ID:

2.2.2.21 effective user ID:

2.2.2.22 empty directory:

2.2.2.23 empty string (null string):

2.2.2.24 Epoch:

2.2.2.25 feature test macro:

2.2.2.26 FIFO special file (FIFO):

2.2.2.27 file:

2.2.2.28 file description:

2.2.2.29 file descriptor:

2.2.2.30 file group class:

2.2.2.31 file mode:

2.2.2.32 filename:

2.2.2.33 file offset:

2.2.2.34 file other class:

2.2.2.35 file owner class:

2.2.2.36 file permission bits:

2.2.2.37 file serial number:

2.2.2.38 file system:

2.2.2.39 foreground process:

2.2.2.40 foreground process group:

2.2.2.41 foreground process group ID:

2.2.2.42 group ID:

2.2.2.43 job control:

2.2.2.44 link:

2.2.2.45 link count:

2.2.2.46 login:

2.2.2.47 login name:

2.2.2.48 mode:

2.2.2.49 null string:

2.2.2.50 open file:

2.2.2.51 open file description:

2.2.2.52 orphaned process group:

2.2.2.53 parent directory:

2.2.2.54 parent process:

2.2.2.55 parent process ID:

When the creator of a process no longer exists, the parent process ID becomes the initialization process (PID = 1). This applies for `wait`, `_exit`, and all other places referring to assignment of a process whose parent has exited to "an implementation-defined system process."

2.2.2.56 path prefix:

2.2.2.57 pathname:

2.2.2.58 pathname component:

2.2.2.59 pipe:

2.2.2.60 portable filename character set:

2.2.2.61 privilege:

2.2.2.62 process:

2.2.2.63 process group:

2.2.2.64 process group ID:

2.2.2.65 process group leader:

2.2.2.66 process group lifetime:

2.2.2.67 process ID:

2.2.2.68 process lifetime:

2.2.2.69 read-only file system:

Indicates mounting a file system as read only with the `mount()` utility.

2.2.2.70 real group ID:

2.2.2.71 real user ID:

2.2.2.72 regular file:

2.2.2.73 relative pathname:

2.2.2.74 root directory:

2.2.2.75 saved set-group-ID:

2.2.2.76 saved set-user-ID:

2.2.2.77 seconds since the Epoch:

2.2.2.78 session:

2.2.2.79 session leader:

2.2.2.80 session lifetime:

2.2.2.81 signal:

2.2.2.82 slash:

2.2.2.83 supplementary group ID:

2.2.2.84 system:

2.2.2.85 system process:

2.2.2.86 terminal (terminal device):

2.2.2.87 user ID:

2.2.2.88 user name:

2.2.2.89 working directory (current working directory):

2.2.3 Abbreviations

2.2.3.1 C Standard:

2.2.3.2 IRV:

2.2.3.3 POSIX.1:

2.3 General Concepts

2.3.1 extended security controls:

2.3.2 file access permissions:

2.3.3 file hierarchy:

2.3.4 filename portability:

2.3.5 file times update:

2.3.6 pathname resolution:

2.4 Error Numbers

For ConvexOS, EFAULT is supported. See <errno.h> for the ConvexOS-specific values of `errno`.

2.5 Primitive System Data Types

For ConvexOS, data types, in addition to those required by POSIX.1, are defined in <sys/types.h>.

2.6 Environment Description

2.7 C Language Definitions

2.7.1 Symbols from the C Standard

2.7.2 POSIX.1 Symbols

2.7.2.1 C Standard Language-Dependent Support

2.7.2.2 Common Usage-Dependent Support

2.7.3 Headers and Function Prototypes

2.8 Numerical Limits

2.8.1 C Language Limits

2.8.2 Minimum Values

2.8.3 Run-Time Increasable Values

For ConvexOS, Table 1 contains the contents of <limits.h>, which includes variable names, values, and descriptions.

Table 1 Contents of <limits.h>

Name	Value	Description
ARG_MAX	12288	Length of the arguments for one of the exec functions in bytes, including environment data.
POSIX_MAX_CANON	255	Number of bytes in a terminal canonical input queue.
POSIX_MAX_INPUT	255	Number of bytes for which space will be available in a terminal input queue.
POSIX_NAME_MAX	14	Number of bytes in a filename component.
NGROUPS_MAX	16	Number of simultaneous supplementary group IDs per process.
OPEN_MAX	256	Number of files that one process can have open at one time.
PATH_MAX	1023	Number of bytes in a pathname.
PIPE_BUF	4096	Number of bytes that can be written atomically when writing to a pipe.

The following variables do not appear in <limits.h> because they are not compile-time variables. Their values can depend on file pathnames or `sysgen()` parameters.

CHILD_MAX — The maximum number of simultaneous processes per real user ID.

LINK_MAX — The maximum value of a file's link count.

2.8.4 Run-Time Invariant Values (Possibly Indeterminate)

Refer to Table 1.

2.8.5 Pathname Variable Values

Refer to Table 1.

2.9 Symbolic Constants

For ConvexOS, Table 2 contains the contents of <unistd.h>, which includes symbolic names, values, and descriptions.

Table 2 Contents of <unistd.h>

Name	Value	Description
F_OK	0	Tests for existence of file.
X_OK	1	Tests for "execute" permission.
W_OK	2	Tests for "write" permission.
R_OK	4	Tests for "read" permission.
SEEK_SET	0	Sets file offset to offset.
SEEK_CUR	1	Sets file offset to current plus offset.
SEEK_END	2	Sets file offset to EOF plus offset.
_POSIX_JOB_CONTROL	1	Supports job control.
_POSIX_SAVED_IDS	1	Supports both saved set-user-IDs and set-group-IDs.
_POSIX_VERSION	19*xxxx*L	Contains the most current version number of the POSIX standard.
_POSIX_VDISABLE	-1	Disables terminal special characters.

The following constants do not appear in <unistd.h> because their values vary depending on the file to which it is applied. See either the `pathconf()` or `fpathconf()` function.

_POSIX_NO_TRUNC	Generates an error if pathname is longer than NAME_MAX.
_POSIX_CHOWN_RESTRICTED	Restricts the use of `chown()`.

2.9.1 Symbolic Constants for the `access()` Function

Refer to Table 2.

2.9.2 Symbolic Constants for the `lseek()` Function

Refer to Table 2.

2.9.3 Compile-Time Symbolic Constants for Portability Specifications

2.9.4 Execution-Time Symbolic Constants for Portability Specifications

Process Primitives 3

3.1 Process Creation and Execution

3.1.1 Process Creation

For ConvexOS, each open directory stream in the child process does share directory stream positioning with the parent.

3.1.2 Execute a File

For ConvexOS, if the environment variable PATH is not defined, the current directory, then /bin, and, finally, /usr/bin are searched. {ARG_MAX} does not include the argument pointers themselves nor is any internal alignment required.

3.2 Process Termination

3.2.1 Wait for Process Termination

For ConvexOS, which supports SIGCHLD, a child process whose parent terminates is assigned the new parent process initialization (PID = 1).

3.2.2 Terminate a Process

For ConvexOS, a child process whose parent terminates is assigned the new parent process initialization (PID = 1).

3.3 Signals

3.3.1 Signal Concepts

3.3.1.1 Signal Names

3.3.1.2 Signal Generation and Delivery

For ConvexOS, once a signal is pending, subsequent occurrences of that signal are discarded until the signal is handled.

The SIGSYS signal is generated when a sysc instruction is executed with an invalid argument in register s0. This should only occur if an executable image is run on a previous version of the operating system.

The SIGTRAP signal is generated when the single step bit in the Process Status Word (PSW) is set or when the process executes a breakpoint instruction. This signal is used by debuggers for controlling program execution.

Both SIGSYS and SIGTRAP cause a core dump if not caught.

The signals SIGXCPU and SIGXFSZ indicate that the process exceeded limits set by `setrlimit()`. See the setrlimit(2) man page for details. These signals are ignored by default.

The SIGVTALRM, SIGPROF, and SIGWINCH signals may occur when certain ConvexOS-specific functions are used. See the signal(3) man page for details.

3.3.1.3 Signal Actions

3.3.1.4 Signal Effects on Other Functions

3.3.2 Send a Signal to a Process

For ConvexOS, `kill` (0, *n)* sends signal *n* to all processes in the caller's process group. There are no exceptions.

3.3.3 Manipulate Signal Sets

3.3.4 Examine and Change Signal Action

3.3.5 Examine and Change Blocked Signals

3.3.6 Examine Pending Signals

3.3.7 Wait for a Signal

3.4 Timer Operations

3.4.1 Schedule Alarm

3.4.2 Suspend Process Execution

3.4.3 Delay Process Execution

Process Environment 4

4.3.3 Set Process Group ID for Job Control

4.4 System Identification

4.4.1 Get System Name

For ConvexOS, Table 3 contains the fields and values of the `uname()` structure.

Table 3 `uname()` Structure Members

Name	Value
`sysname`	vmunix
`nodename`	Current host name (refer to the sethostname(2) and gethostname(2) man pages)
`release`	The configuration of the machine; for example, C120, C210, C3210, etc.
`version`	8.0 or greater
`machine`	convex

4.5 Time

4.5.1 Get System Time

For ConvexOS, there are no failure conditions for `time()`.

4.5.2 Get Process Times

For ConvexOS, there are no failure conditions for `times()`.

4.6 Environment Variables

4.6.1 Environment Access

4.7 Terminal Identification

4.7.1 Generate Terminal Pathname

4.7.2 Determine Terminal Device Name

4.8 Configurable System Variables

4.8.1 Get Configurable System Variables

Files and Directories 5

5.1 Directories

5.1.1 Format of Directory Entries

5.1.2 Directory Operations

5.2 Working Directory

5.2.1 Change Current Working Directory

5.2.2 Get Working Directory Pathname

5.3 General File Creation

5.3.1 Open a File

For ConvexOS, files and directories are created with group ownership of the parent directory.

5.3.2 Create a New File or Rewrite an Existing One

For ConvexOS, files and directories are created with group ownership of the parent directory.

5.3.3 Set File Creation Mask

5.3.4 Link to a File

5.4 Special File Creation

5.4.1 Make a Directory

5.4.2 Make a FIFO Special File

For ConvexOS, files and directories are created with group ownership of the parent directory.

Bits other than file permission bits are forced off, regardless of their state in the mode argument to `mkfifo()`. In other words, they are silently ignored.

5.5 File Removal

5.5.1 Remove Directory Entries

5.5.2 Remove a Directory

For ConvexOS, `rmdir()` allows removal of the current directory of a process, including the caller, only if it is an empty directory. When a process' current working directory is removed, that process will be unable to access "." or "..", and `getcwd()` will cease to work for that process until it calls `chdir()` to make its working directory one that exists.

Removing root directories is not allowed. The attempt always fails, returning a -1 and errno EBUSY.

5.5.3 Rename a File

5.6 File Characteristics

5.6.1 File Characteristics: Header and Data Structure

5.6.1.1 <sys/stat.h> File Types

5.6.1.2 <sys/stat.h> File Modes

5.6.1.3 <sys/stat.h> Time Entries

5.6.2 Get File Status

5.6.3 Check File Accessibility

5.6.4 Change File Modes

For ConvexOS, no additional failure conditions exist for `chmod()`.

5.6.5 Change Owner and Group of a File

For ConvexOS, S_ISUID and S_ISGID are cleared unless the caller is superuser.

5.6.6 Set File Access and Modification Times

5.7 Configurable Pathname Variables

5.7.1 Get Configurable Pathname Variables

Input and Output Primitives 6

6.1 Pipes

6.1.1 Create an Inter-Process Channel

6.2 File Descriptor Manipulation

6.2.1 Duplicate an Open File Descriptor

6.3 File Descriptor Deassignment

6.3.1 Close a File

6.4 Input and Output

6.4.1 Read from a File

For ConvexOS, EIO is generated on attempts to read from a network pty when no writer is present. It may also occur on a true device error. See the intro(4) man page for device-specific information.

6.4.2 Write to a File

For ConvexOS, EIO occurs on various device hardware errors. See the intro(4) man page for device-specific information.

6.5 Control Operations on Files

6.5.1 Data Definitions for File Control Operations

6.5.2 File Control

For ConvexOS, `errno` is set to EACCES when either EACCES or EAGAIN are allowed.

6.5.3 Reposition Read/Write File Offset

For ConvexOS, the `lseek()` function on a character-special device returns a random number without affecting the `errno` value.

Device- and Class-Specific Functions

7

7.1 General Terminal Interface

For ConvexOS, the termios interface supports both synchronous ports and network connections in addition to asynchronous ports.

7.1.1 Interface Characteristics

7.1.1.1 Opening a Terminal Device File

7.1.1.2 Process Groups

For ConvexOS, the operating system supports job control.

7.1.1.3 The Controlling Terminal

For ConvexOS, a session leader that does not have a controlling terminal allocates one by opening a terminal device file that is not already associated with a session without using the O_NOCTTY option to `open()`.

7.1.1.4 Terminal Access Control

7.1.1.5 Input Processing and Reading Data

For ConvexOS, one of two user-selectable options governs the behavior of the system when the input stream overflows. If the first option is selected, the system refuses to accept any further input and echoes a bell. Further input will not be stored, but any input already present in the input stream is not disturbed. If the second option is selected, no bell is echoed and all input present in the input queue is silently discarded if the input stream overflows.

Refer to the termios(4) man page for information on how to select one of the two options.

7.1.1.6 Canonical Mode Input Processing

For ConvexOS, if the number of bytes in a canonical input line is exceeded, the excess input is silently discarded.

7.1.1.7 Noncanonical Mode Input Processing

For ConvexOS, a request for noncanonical mode input with a value of MIN greater than MAX_INPUT will block indefinitely in the read operation until a signal is delivered to the requesting process (causing the read to be interrupted).

7.1.1.7.1 Case A: MIN > 0, TIME > 0

7.1.1.7.2 Case B: MIN > 0, TIME = 0

7.1.1.7.3 Case C: MIN = 0, TIME > 0

7.1.1.7.4 Case D: MIN = 0, TIME = 0

7.1.1.8 Writing Data and Output Processing

For ConvexOS, there is a buffering mechanism. When a call to `write()` completes, all of the bytes written have been scheduled for transmission to the device, but the transmission will not necessarily have completed.

7.1.1.9 Special Characters

For ConvexOS, values of special characters START and STOP can be changed as desired.

As a special case, if values of special characters START and STOP are identical, that special character acts as a toggle to stop output if it is started and start output if it is stopped.

7.1.1.10 Modem Disconnect

7.1.1.11 Closing a Terminal Device File

7.1.2 Parameters That Can Be Set

7.1.2.1 `termios` Structure

For ConvexOS, the size of a `termios` structure is 132 bytes.

7.1.2.2 Input Modes

For ConvexOS, in contexts other than asynchronous serial data transmissions, there is no break condition.

The system will send a STOP character when a character is received and the following conditions are true:

```
if (number-chars-available >= ((clist_size - 30) / 2) &&
    (noncanonical-input-mode || chars-on-canonical-queue > 0) &&
    STOP-char-enabled)
    send STOP char
```

where:

`number-chars-available`	Number of characters in the raw and canonical input queues.
`clist_size`	Maximum size of the clists for the terminal device. This is a tunable parameter. Refer to *Managing ConvexOS: Configuration Guide*, the chapter "Customizing Kernel Boot-Time Parameters," for parameters `tty_iop_size`, `tty_viop_size`, and `tty_pty_size`.
`noncanonical-input-mode`	True if ICANON is not set in the `c_lflag` field of the `termios` structure for this terminal device.
`chars-on-canonical-queue`	Number of characters on the canonical input queue currently available for reading.
`STOP-char-enabled`	True if the STOP character is not disabled for this terminal device.

The system will send a START character when a character is received and the following conditions are true:

```
if (chars-on-raw-queue < ((clist_size - 30) / 5) &&
    START-char-enabled)
    send START char
```

where:

`chars-on-raw-queue`	Number of characters on the raw input queue.
`clist_size`	Maximum size of the clists for the terminal device. This is a tunable parameter. Refer to *Managing ConvexOS: Configuration Guide*, the chapter "Customizing Kernel Boot-Time Parameters," for parameters `tty_iop_size`, `tty_viop_size`, and `tty_pty_size`.
`START-char-enabled`	True if the START character is not disabled for this terminal device.

The initial input control value after `open()` is the inclusive OR of the following masks:

- BRKINT
- ICRNL
- IMAXBEL
- ISTRIP
- IXANY
- IXON

See the termios(4) man page for descriptions of other ConvexOS-specific masks.

7.1.2.3 Output Modes

For ConvexOS, if OPOST is set, certain ConvexOS-specific translations occur on output data. See both the termios(4) and tty(4) man pages for information on these translations.

The initial output control value after `open()` is the inclusive OR of OPOST and other ConvexOS-specific masks. See the termios(4) man page for descriptions of these masks.

7.1.2.4 Control Modes

For ConvexOS, the initial hardware control value after `open()` is the inclusive OR of the following masks:

- CREAD
- CS8
- HUPCL

7.1.2.5 Local Modes

For ConvexOS, if IEXTEN is set, checking input characters against certain ConvexOS-specific special control characters is enabled. When not set, checking is disabled. See the termios (4) man page for descriptions of the ConvexOS-specific control characters.

The initial local control value after `open()` is the inclusive OR of the following masks:

- ECHO
- ECHOCTL
- ECHOE
- ECHOK
- ICANON
- IEXTEN
- ISIG

Refer to the termios(4) man page for descriptions of other ConvexOS-specific masks.

7.1.2.6 Special Control Characters

For ConvexOS, NCCS, number of elements in the `c_cc` array, is defined to be 32.

Table 4 contains relative positions and initial values of each function in the `c_cc` array.

Table 4 c_cc Array Functions

Position	Function	Initial value
0	EOF	CTRL-d
1	EOL	disable
3	ERASE	DELETE
6	KILL	CTRL-u
8	INTR	CTRL-c
9	QUIT	CTRL-\
10	SUSP	CTRL-Z
12	START	CTRL-Q
13	STOP	CTRL-S
16	MIN	1
17	TIME	0

The values of indices 20 through 31 have no function and are initialized to 0xff. Other positions contain values for ConvexOS-specific functions. See the termios(4) man page for descriptions of these.

7.1.3 Baud Rate Functions

For ConvexOS, attempts to set baud rates with `cfsetispeed()`, `cfsetospeed()`, or `tcsetattr()` to values unsupported by the hardware are silently ignored. The baud rate remains unchanged.

Both `cfsetispeed()` and `cfsetospeed()` return -1 to indicate the desired speed was not within the range of known speeds as defined in <termios.h>.

7.2 General Terminal Interface Control Functions

7.2.1 Get and Set State

7.2.2 Line Control Functions

For ConvexOS, if the duration argument to `tcsendbreak()` is non-zero, it is interpreted as the duration of the break in increments of 0.1 second.

7.2.3 Get Foreground Process Group ID

7.2.4 Set Foreground Process Group ID

Language-Specific Services for the C Programming Language

8

8.1 Referenced C Language Routines

8.1.1 Extensions to Time Functions

8.1.2 Extensions to `setlocale()` Function

For ConvexOS, the empty string is equivalent to the "C" locale.

8.2 C Language Input/Output Functions

8.2.1 Map a Stream Pointer to a File Descriptor

8.2.2 Open a Stream on a File Descriptor

For ConvexOS, no additional types are supported.

8.2.3 Interactions of Other FILE-Type C Functions

8.2.3.1 `fopen()`

8.2.3.2 `fclose()`

8.2.3.3 `freopen()`

8.2.3.4 `fflush()`

8.2.3.5 `fgetc()`, `fgets()`, `fread()`, `getc()`, `getchar()`, `gets()`, `scanf()`, `fscanf()`

8.2.3.6 `fputc()`, `fputs()`, `fwrite()`, `putc()`, `putchar()`, `puts()`, `printf()`, `fprintf()`

8.2.3.7 `fseek()`, `rewind()`

8.2.3.8 `perror()`

8.2.3.9 `tmpfile()`

8.2.3.10 `ftell()`

8.2.3.11 Error Reporting

8.2.3.12 `exit()`, `abort()`

8.2.4 Operations on Files—the `remove()` Function

8.3 Other C Language Functions

8.3.1 Nonlocal Jumps

8.3.2 Set Time Zone

System Databases

9

9.1 System Databases

9.2 Database Access

9.2.1 Group Database Access

9.2.2 User Database Access

Data Interchange Format 10

10.1 Archive/ Interchange File Format

10.1.1 Extended `tar` Format

Refer to the tar(1) man page for details.

10.1.2 Extended `cpio` Format

Refer to the cpio(1) man page for details.

10.1.2.1 `cpio` Header

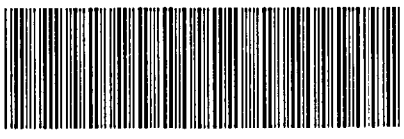
10.1.2.2 `cpio` File Name

10.1.2.3 `cpio` File Data

10.1.2.4 `cpio` Special Entries

10.1.2.5 `cpio` Values

10.1.3 Multiple Volumes



Order Number
DSW-311

Document Number
710-002030-202