man pages section 3: Library Interfaces and Headers

Sun Microsystems, Inc.
4150 Network Circle
Santa Clara, CA 95054
U.S.A.

Part No: 816–0218–10
May 2002

020313@3332

Contents

Preface

Both novice users and those familar with the SunOS operating system can use online man pages to obtain information about the system and its features. A man page is intended to answer concisely the question “What does it do?” The man pages in general comprise a reference manual. They are not intended to be a tutorial.

Overview

The following contains a brief description of each man page section and the information it references:

- Section 1 describes, in alphabetical order, commands available with the operating system.
- Section 1M describes, in alphabetical order, commands that are used chiefly for system maintenance and administration purposes.
- Section 2 describes all of the system calls. Most of these calls have one or more error returns. An error condition is indicated by an otherwise impossible returned value.
- Section 3 describes functions found in various libraries, other than those functions that directly invoke UNIX system primitives, which are described in Section 2.
- Section 4 outlines the formats of various files. The C structure declarations for the file formats are given where applicable.
- Section 5 contains miscellaneous documentation such as character-set tables.
- Section 6 contains available games and demos.
- Section 7 describes various special files that refer to specific hardware peripherals and device drivers. STREAMS software drivers, modules and the STREAMS-generic set of system calls are also described.

- Section 9 provides reference information needed to write device drivers in the kernel environment. It describes two device driver interface specifications: the Device Driver Interface (DDI) and the Driver/Kernel Interface (DKI).
- Section 9E describes the DDI/DKI, DDI-only, and DKI-only entry-point routines a developer can include in a device driver.
- Section 9F describes the kernel functions available for use by device drivers.
- Section 9S describes the data structures used by drivers to share information between the driver and the kernel.

Below is a generic format for man pages. The man pages of each manual section generally follow this order, but include only needed headings. For example, if there are no bugs to report, there is no BUGS section. See the `intro` pages for more information and detail about each section, and `man`(1) for more information about man pages in general.

NAME	This section gives the names of the commands or functions documented, followed by a brief description of what they do.
SYNOPSIS	This section shows the syntax of commands or functions. When a command or file does not exist in the standard path, its full path name is shown. Options and arguments are alphabetized, with single letter arguments first, and options with arguments next, unless a different argument order is required. The following special characters are used in this section: [ ] Brackets. The option or argument enclosed in these brackets is optional. If the brackets are omitted, the argument must be specified. . . . Ellipses. Several values can be provided for the previous argument, or the previous argument can be specified multiple times, for example, "`filename` . . ." . | Separator. Only one of the arguments separated by this character can be specified at a time. { } Braces. The options and/or arguments enclosed within braces are interdependent, such that everything enclosed must be treated as a unit.

PROTOCOL	This section occurs only in subsection 3R to indicate the protocol description file.
DESCRIPTION	This section defines the functionality and behavior of the service. Thus it describes concisely what the command does. It does not discuss OPTIONS or cite EXAMPLES. Interactive commands, subcommands, requests, macros, and functions are described under USAGE.
IOCTL	This section appears on pages in Section 7 only. Only the device class that supplies appropriate parameters to the `ioctl`(2) system call is called `ioctl` and generates its own heading. `ioctl` calls for a specific device are listed alphabetically (on the man page for that specific device). `ioctl` calls are used for a particular class of devices all of which have an `io` ending, such as `mtio`(7I).
OPTIONS	This secton lists the command options with a concise summary of what each option does. The options are listed literally and in the order they appear in the SYNOPSIS section. Possible arguments to options are discussed under the option, and where appropriate, default values are supplied.
OPERANDS	This section lists the command operands and describes how they affect the actions of the command.
OUTPUT	This section describes the output – standard output, standard error, or output files – generated by the command.
RETURN VALUES	If the man page documents functions that return values, this section lists these values and describes the conditions under which they are returned. If a function can return only constant values, such as 0 or –1, these values are listed in tagged paragraphs. Otherwise, a single paragraph describes the return values of each function. Functions declared void do not return values, so they are not discussed in RETURN VALUES.
ERRORS	On failure, most functions place an error code in the global variable `errno` indicating why they failed. This section lists alphabetically all error codes a function can generate and describes the conditions that cause each error. When more than

	one condition can cause the same error, each condition is described in a separate paragraph under the error code.
USAGE	This section lists special rules, features, and commands that require in-depth explanations. The subsections listed here are used to explain built-in functionality: Commands Modifiers Variables Expressions Input Grammar
EXAMPLES	This section provides examples of usage or of how to use a command or function. Wherever possible a complete example including command-line entry and machine response is shown. Whenever an example is given, the prompt is shown as `example%`, or if the user must be superuser, `example#`. Examples are followed by explanations, variable substitution rules, or returned values. Most examples illustrate concepts from the SYNOPSIS, DESCRIPTION, OPTIONS, and USAGE sections.
ENVIRONMENT VARIABLES	This section lists any environment variables that the command or function affects, followed by a brief description of the effect.
EXIT STATUS	This section lists the values the command returns to the calling program or shell and the conditions that cause these values to be returned. Usually, zero is returned for successful completion, and values other than zero for various error conditions.
FILES	This section lists all file names referred to by the man page, files of interest, and files created or required by commands. Each is followed by a descriptive summary or explanation.
ATTRIBUTES	This section lists characteristics of commands, utilities, and device drivers by defining the attribute type and its corresponding value. See `attributes`(5) for more information.
SEE ALSO	This section lists references to other man pages, in-house documentation, and outside publications.

DIAGNOSTICS	This section lists diagnostic messages with a brief explanation of the condition causing the error.
WARNINGS	This section lists warnings about special conditions which could seriously affect your working conditions. This is not a list of diagnostics.
NOTES	This section lists additional information that does not belong anywhere else on the page. It takes the form of an aside to the user, covering points of special interest. Critical information is never covered here.
BUGS	This section describes known bugs and, wherever possible, suggests workarounds.

Introduction

NAME

Intro – introduction to functions and libraries

DESCRIPTION

This section describes functions found in various Solaris libraries, other than those functions described in Section 2 of this manual that directly invoke UNIX system primitives. Function declarations can be obtained from the `#include` files indicated on each page. Pages are grouped by library and are identified by the library name (or an abbreviation of the library name) after the section number. Collections of related libraries are grouped into five volumes as described below. A sixth volume (listed first) contains pages describing the contents of each shared library and each header used by the functions, macros, and external variables described in the remaining five volumes.

Library Interfaces and Headers

This volume describes the contents of each shared library and each header used by functions, macros, and external variables described in the remaining five volumes.

(3LIB) The libraries described in this section are implemented as shared objects.

Descriptions of shared objects may include a definition of the global symbols that define the shared objects' public interface, for example `SUNW_1.1`. Other interfaces may exist within the shared object, for example `SUNW_private.1.1`. The public interface provides a stable, committed set of symbols for application development. The private interfaces are for internal use only, and may change at any time.

For many shared objects, an archive library is provided for backward compatibility on 32–bit systems only. Use of these libraries may restrict an applications ability to migrate between different Solaris releases. As dynamic linking is the preferred compilation method on Solaris, the use of these libraries is discouraged.

(3LIBUCB) The SunOS/BSD Compatibility libraries described in this section are implemented as a shared object. See (3LIB) above.

(3HEAD) The headers described in this section are used by functions, macros, and external variables. Headers contain function prototypes, definitions of symbolic constants, common structures, preprocessor macros, and defined types. Each function described in the remaining five volumes specifies the headers that an application must include in order to use that function. In most cases only one header is required. These headers are present on an application development system; they do have to be present on the target execution system.

Basic Library Functions

The functions described in this volume are the core C library functions that are basic to application development.

(3C) These functions, together with those of Section 2, constitute the standard C library, `libc`, which is automatically linked by the C

compilation system. The standard C library is implemented as a shared object, `libc.so`, and as an archive, `libc.a`. C programs are linked with the shared object version of the standard C library by default. Specify `-Bstatic` or `-dn` on the `cc` command line to link with the archive version. See `libc`(3LIB), `cc`(1B) for other overrides, and the "C Compilation System" chapter of the *ANSI C Programmer's Guide* for a discussion. Some functions behave differently in standard-conforming environments. This behavior is noted on the individual manual pages. See `standards`(5).

(3DL) These functions constitute the dynamic linking library, `libdl`. This library is implemented as a shared object, `libdl.so`, but is not automatically linked by the C compilation system. Specify `-ldl` on the `cc` command line to link with this library. See `libdl`(3LIB).

(3MALLOC) These functions constitute the various memory allocation libraries: `libmalloc`, `libbsdmalloc`, `libmapmalloc`, and `libmtmalloc`. Each of these libraries is implemented as a shared object (`libmalloc.so`, `libbsdmalloc.so`, `libmapmalloc.so`, and `libmtmalloc.so`) and all except `libmtmalloc` are implemented as archives (`libmalloc.a`, `libbsdmalloc.a`, `libmapmalloc.a`). These libraries are not automatically linked by the C compilation system. Specify `-lmalloc`, `-lbsdmalloc`, `-lmapmalloc`, and `-lmtmalloc` to link with, respectively, `libmalloc`, `libbsdmalloc`, `libmapmalloc`, and `libmtmalloc`. See `libmalloc`(3LIB), `libbsdmalloc`(3LIB), `libmapmalloc`(3LIB), and `libmtmalloc`(3LIB).

(3UCB) These functions constitute the Source Compatibility (with BSD functions) library. It is implemented as a shared object, `libucb.so`, and as an archive, `libucb.a`, but is not automatically linked by the C compilation system. Specify `-lucb` on the `cc` command line to link with this library, which is located in the `/usr/ucb` subdirectory. Headers for this library are located within `/usr/ucbinclude`. See `libucb`(3LIBUCB).

Networking Library Functions

The functions described in this volume comprise the various networking libraries.

(3GSS) The functions in this library are the routines that comprise the Generic Security Services API library. This library is implemented as a shared object, `libgss.so.1`, but it is not automatically linked by the C compilation system. Specify `-lgss` on the `cc` command line to link with this library. See `libgss`(3LIB).

(3LDAP) These functions constitute the Lightweight Directory Access Protocol library, `libldap`. This library is implemented as a shared object, `libldap.so`, but is not automatically linked by the C compilation system. Specify `-lldap` on the `cc` command line to link with this library. See `ldap`(3LDAP).

(3NSL) These functions constitute the Network Service Library, `libnsl`. This library is implemented as a shared object, `libnsl.so`, and as an archive, `libnsl.a`, but is not automatically linked by the C compilation system. Specify `-lnsl` on the `cc` command line to link with this library. See `libnsl`(3LIB).

Many base networking functions are also available in the X/Open Networking Interfaces library, `libxnet`. See section (3XNET) below for more information on the `libxnet` interfaces.

(3RAC) These functions constitute the remote asynchronous calls library, `librac`. This library is implemented as a shared object, `librac.so`, and as an archive, `librac.a`, but is not automatically linked by the C compilation system. Specify `-lrac` on the `cc` command line to link with this library. See `librac`(3LIB).

(3RESOLV) These functions constitute the resolver library, `libresolv`. This library is implemented as a shared object, `libresolv.so`, and as an archive, `libresolv.a`, but is not automatically linked by the C compilation system. Specify `-lresolv` on the `cc` command line to link with this library. See `libresolv`(3LIB).

(3RPC) These functions constitute the remote procedure call libraries, `librpcsvc` and `librpcsoc`. The latter is provided for compatibility only; new applications should not link to it. Both libraries are implemented as shared objects, `librpcsvc.so` and `librpcsoc.so`, respectively, and `librpcsvc` is implemented as an archive, `librpcsvc.a`. Neither library is automatically linked by the C compilation system. Specify `-lrpcsvc` or `-lrpcsoc` on the `cc` command line to link with these libraries. See `librpcsvc`(3LIB) and `librpcsoc`(3LIBUCB).

(3SLP) These functions constitute the Service Location Protocol library, `libslp`. This library is implemented as a shared object, `libslp.so.1`, but it is not automatically linked by the C compilation system. See `libslp`(3LIB)

(3SOCKET) These functions constitute the sockets library, `libsocket`. This library is implemented as a shared object, `libsocket.so`, and as an archive, `libsocket.a`, but is not automatically linked by the C compilation system. Specify `-lsocket` on the `cc` command line to link with this library. See `libsocket`(3LIB).

(3XFN) These functions constitute the X/Open Federated Naming library, `libxfn`. This library is implemented as a shared object, `libxfn.so`, but is not automatically linked by the C compilation system. Specify `-lxfn` on the `cc` command line to link with this library. See `libxfn`(3LIB), `xfn`(3XFN), `fns`(5), and `standards`(5).

(3XNET) These functions constitute X/Open networking interfaces which comply with the X/Open CAE Specification, Networking Services, Issue 4 (September, 1994). This library is implemented as a shared object, `libxnet.so`, but is not automatically linked by the C compilation system. Specify `-lxnet` on the `cc` command line to link with this library. See `libxnet`(3LIB) and `standards`(5) for compilation information.

Under all circumstances, the use of the Sockets API is recommended over the XTI and TLI APIs. If portability to other XPGV4v2 (see `standards`(5)) systems is a requirement, the application must use the `libxnet` interfaces. If portability is not required, the sockets interfaces in `libsocket` and `libnsl` are recommended over those in `libxnet`. Between the XTI and TLI APIs, the XTI interfaces (available with `libxnet`) are recommended over the TLI interfaces (available with `libnsl`).

Curses Library Functions

The functions described in this volume comprise the libraries that provide graphics and character screen updating capabilities.

(3CURSES) The functions constitute the following libraries:

libcurses — These functions constitute the curses library, `libcurses`. This library is implemented as a shared object, `libcurses.so`, and as an archive, `libcurses.a`, but is not automatically linked by the C compilation system. Specify `-lcurses` on the `cc` command line to link with this library. See `libcurses`(3LIB).

libform — These functions constitute the forms library, `libform`. This library is implemented as a shared object, `libform.so`, and as an archive, `libforms.a`, but is not automatically linked by the C compilation system. Specify `-lform` on the `cc` command line to link with this library. See `libform`(3LIB).

libmenu — These functions constitute the menus library, `libmenu`. This library is implemented as a shared object, `libmenu.so`, and as an archive, `libmenu.a`, but is not automatically linked by the C compilation system. Specify `-lmenu` on the `cc` command line to link with this library. See `libmenu`(3LIB).

libpanel — These functions constitute the panels library, `libpanel`. This library is implemented as a shared object, `libpanel.so`, and as an archive, `libpanel.a`, but is not automatically linked by the C compilation system. Specify

`-lpanel` on the `cc` command line to link with this library. See `libpanel`(3LIB).

(3PLOT) These functions constitute the grapnics library, `libplot`. This library is implemented as a shared object, `libplot.so`, and as an archive, `libplot.a`, but is not automatically linked by the C compilation system. Specify `-lplot` on the `cc` command line to link with this library. See `libplot`(3LIB).

(3XCURSES) These functions constitute the X/Open Curses library, located in `/usr/xpg4/lib/libcurses.so.1`. This library provides a set of internationalized functions and macros for creating and modifying input and output to a terminal screen. Included in this library are functions for creating windows, highlighting text, writing to the screen, reading from user input, and moving the cursor. X/Open Curses is designed to optimize screen update activities. The X/Open Curses library conforms fully with Issue 4 of the X/Open Extended Curses specification.

Threads and Realtime Library Functions

The functions described in this volume constitute the threads and realtime libraries.

(3AIO) These functions constitute the asynchronous I/O library, `liaio`. This library is implemented as a shared object, `libaio.so`, but is not automatically linked by the C compilation system. Specify `-laio` on the `cc` command line to link with this library. See `libaio`(3LIB).

(3DOOR) These functions constitute the doors library, `libdoor`. This library is implemented as a shared object, `libdoor.so`, but is not automatically linked by the C compilation system. Specify `-ldoor` on the `cc` command line to link with this library.

(3RT) These functions constitute the POSIX.4 Realtime library, `librt`. It is implemented as a shared object, `librt.so`, but is not automatically linked by the C compilation system. Specify `-lrt` on the `cc` command line to link with this library. Note that the former name for this library, `libposix4`, is maintained for backward compatibility but should be avoided. See `librt`(3LIB).

(3SCHED) These functions constitute the LWP scheduling library, `libsched`. This library is implemented as a shared object, `libsched.so`, but is not automatically linked by the C compilation system. Specify `-lsched` on the `cc` command line to link with this library. .

(3THR) These functions constitute the threads libraries, `libpthread`, `libthread`, and `libthread_db`. The `libpthread` and `libthread` libraries are used for building multithreaded applications: `libpthread` implements the POSIX (see `standards`(5)) threads interface, whereas `libthread`

implements the Solaris threads interface. The `libthread_db` library is useful for building debuggers for multithreaded applications.

Both POSIX threads and Solaris threads can be used within the same application. Their implementations are completely compatible with each other; however, only POSIX threads guarantee portability to other POSIX-conforming environments.

When POSIX and Solaris threads are used in the same application, if there are calls with the same name but different semantics, the POSIX semantic supersedes the Solaris threads semantic. For example, the call to `fork()` will imply the `fork1()` semantic in a program linked with the POSIX threads library, whether or not it is also linked with `-lthread` (Solaris threads).

The `libpthread`, `libthread`, and `libthread_db` libraries are implemented as shared objects, `libpthread.so`, `libthread_db.so`, and `libthread.so`, respectively. These libraries are not automatically linked by the C compilation system. Specify `-lpthread`, `-lthread`, or `-lthread_db` on the `cc` command line to link with these libraries. See `libpthread`(3LIB), `libthread`(3LIB), and `libthread_db`(3LIB).

Extended Library Functions

The functions described in this volume comprise various specialized libraries that are not limited to the following:

(3BSM) These functions constitute the basic security library, `libbsm`. This library is implemented as a shared object, `libbsm.so`, and as an archive, `libbsm.a`, but is not automatically linked by the C compilation system. Specify `-lbsm` on the `cc` command line to link with this library. See `libbsm`(3LIB).

(3CFGADM) These functions constitute the configuration administration library, `libcfgadm`. This library is implemented as a shared object, `libcfgadm.so`, but is not automatically linked by the C compilation system. Specify `-lcfgadm` on the `cc` command line to link with this library. See `libcfgadm`(3LIB).

(3CPC) These functions constitute the CPU performance counter library, `libcpc`, and the process context library, `libpctx`. These libraries are implemented as shared objects, `libcpc.so` and `libpctx.so`, respectively, but are not automatically linked by the C compilation system. Specify `-lcpc` or `-lpctx` on the `cc` command line to link with these libraries. See `libcpc`(3LIB) and `libpctx`(3LIB).

(3DEVID) These functions constitute the device ID library, `libdevid`. This library is implemented as a shared object, `libdevid.so`, but is

	not automatically linked by the C compilation system. Specify `-ldevid` on the `cc` command line to link with this library. See `libdevid`(3LIB).
(3DEVINFO)	These functions constitute the device information library, `libdevinfo`. This library is implemented as a shared object, `libdevinfo.so`, but is not automatically linked by the C compilation system. Specify `-ldevinfo` on the `cc` command line to link with this library. See `libdevinfo`(3LIB).
(3DMI)	These functions constitute the DMI libraries, `libdmi`, `libdmici`, and `libdmimi`. These libraries are implemented as shared objects, `libdmi.so`, `libdmici.so`, and `libdmimi.so`, respectively, but are not automatically linked by the C compilation system. Specify `-ldmi`, `-ldmici`, or `-ldmimi` on the `cc` command line to link with these libraries. See `libdmi`(3LIB), `libdmici`(3LIB), and `libdmimi`(3LIB).
(3ELF)	These functions constitute the ELF access library, `libelf`, (Extensible Linking Format). This library provides the interface for the creation and analyses of "elf" files; executables, objects, and shared objects. `libelf` is implemented as a shared object, `libelf.so`, and as an archive, `libelf.a`, but is not automatically linked by the C compilation system. Specify `-lelf` on the `cc` command line to link with this library. See `libelf`(3LIB).
(3EXACCT)	These functions constitute the extended accounting access library, `libexacct`, and the project database access library, `libproject`. These libraries are implemented as shared objects, `libexacct.so` and `libproject.so`, respectively, but are not automatically linked by the C compilation system. Specify `-lexacct` or `-lproject` on the `cc` command line to link with these libraries. See `libexacct`(3LIB) and `libproject`(3LIB).
(3GEN)	These functions constitute the string pattern-matching and pathname manipulation library, `libgen`. This library is implemented as a shared object, `libgen.so`, and as an archive, `libgen.a`, but is not automatically linked by the C compilation system. Specify `-lgen` on the `cc` command line to link with this library. See `libgen`(3LIB).
(3KSTAT)	These functions constitute the kernel statistics library, which is implemented as a shared object, `libkstat.so`, and as an archive, `libkstat.a`, but is not automatically linked by the C compilation system. Specify `-lkstat` on the `cc` command line to link with this library. See `libkstat`(3LIB).
(3KVM)	These functions allow access to the kernel's virtual memory library, which is implemented as a shared object, `libkvm.so`, and

as an archive, `libkvm.a`, but is not automatically linked by the C compilation system. Specify `-lkvm` on the `cc` command line to link with this library. See `libkvm`(3LIB).

(3LAYOUT) These functions constitute the layout service library, which is implemented as a shared object, `liblayout.so`, but is not automatically linked by the C compilation system. Specify `-llayout` on the `cc` command line to link with this library. See `liblayout`(3LIB).

(3M) These functions constitute the mathematical library, `libm`. This library is implemented as a shared object, `libm.so`, and as an archive, `libm.a`, but is not automatically linked by the C compilation system. Specify `-lm` on the `cc` command line to link with this library.

(3MAIL) These functions constitute the user mailbox management library, `libmail`. This library is implemented as a shared object, `libmail.so`, and as an archive, `libmail.a`, but is not automatically linked by the C compilation system. Specify `-lmail` on the `cc` command line to link with this library.

(3MP) These functions constitute the integer mathematical library, `libmp`. This library is implemented as a shared object, `libmp.so`, and as an archive, `libmp.a`, but is not automatically linked by the C compilation system. Specify `-lmp` on the `cc` command line to link with this library. See `libmp`(3LIB).

(3NVPAIR) These functions constitute the name–value pair library, `libnvpair`. This library is implemented as a shared object, `libnvpair.so`, but is not automatically linked by the C compilation system. Specify `-lnvpair` on the `cc` command line to link with this library. See `libnvpair`(3LIB).

(3PAM) These functions constitute the Pluggable Authentication Module (PAM) library, `libpam`. This library is implemented as a shared object, `libpam.so`, and as an archive, `libpam.a`, but is not automatically linked by the C compilation system. Specify `-lpam` on the `cc` command line to link with this library. See `libpam`(3LIB).

(3PICL) These functions constitute the PICL library, `libpicl`. This library is implemented as a shared object, `libpicl.so`, but is not automatically linked by the C compilation system. Specify `-lpicl` on the `cc` command line to link with this library. See `libpicl`(3LIB) and `libpicl`(3PICL).

(3PICLTREE) These functions constitute the PICL plug-in library, `libpicltree`. This library is implemented as a shared object, `libpicltree.so`, but is not automatically linked by the C compilation system.

	Specify `-lpicltree` on the `cc` command line to link with this library. See `libpicltree`(3LIB) and `libpicltree`(3PICLTREE).
(3POOL)	These functions constitute the pool configuration manipulation library, `libpool`. This library is implemented as a shared object, `libpool.so`, but is not automatically linked by the C compilation system. Specify `-lpool` on the `cc` command line to link with this library. See `libpool`(3LIB).
(3PROJECT)	These functions constitute the project database access library, `libproject`. This library is implemented as a shared object, `libproject.so`, but is not automatically linked by the C compilation system. Specify `-lproject` on the `cc` command line to link with this library. See `libproject`(3LIB).
(3RSM)	These functions constitute the remote shared memory library, `librsm`. This library is implemented as a shared object, `librsm.so`, but is not automatically linked by the C compilation system. Specify `-lrsm` on the `cc` command line to link with this library. See `librsm`(3LIB).
(3SEC)	These functions constitute the file access control library, `libsec`. This library is implemented as a shared object, `libsec.so`, and as an archive, `libsec.a`, but is not automatically linked by the C compilation system. Specify `-lsec` on the `cc` command line to link with this library. See `libsec`(3LIB).
(3SECDB)	These functions constitute the security attributes database library, `libsecdb`. This library is implemented as a shared object, `libsecdb.so`, but is not automatically linked by the C compilation system. Specify `-lsecdb` on the `cc` command line to link with this library. See `libsecdb`(3LIB).
(3SNMP)	These functions constitute the SNMP libraries, `libdssagent` and `libdssasnmp`. These libraries are implemented as shared objects, `libssagent.so` and `libssasnmp.so`, respectively, but are not automatically linked by the C compilation system. Specify `-lssagent` or `-lssasnmp` on the `cc` command line to link with these libraries. See `libssagent`(3LIB) and `libssasnmp`(3LIB).
(3SYSEVENT)	These functions constitute the system event library, `libsysevent`. This library is implemented as a shared object, `libsysevent.so`, but is not automatically linked by the C compilation system. Specify `-lsysevent` on the `cc` command line to link with this library. See `libsysevent`(3LIB).
(3TNF)	These functions constitute the TNF libraries, `libtnf`, `libtnfctl`, and `libtnfprobe`. These libraries are implemented as shared objects, `libtnf.so`, `libtnfctl.so`, and `libtnfprobe.so`, respectively, but are not automatically linked by the C compilation

system. Specify `-ltnf`, `-ltnfctl`, or `-ltnfprobe` on the `cc` command line to link with these libraries. See `libtnfctl`(3TNF) and `libtnfctl`(3LIB).

(3VOLMGT) These functions constitute the volume management library, `libvolmgt`. This library is implemented as a shared object, `libvolmgt.so`, and as an archive, `libvolmgt.a`, but is not automatically linked by the C compilation system. Specify `-lvolmgt` on the `cc` command line to link with this library. See `libvolmgt`(3LIB).

(3WSREG) These functions constitute the product install registry library, `libwsreg`. This library is implemented as a shared object, `libwsreg.so`, but is not automatically linked by the C compilation system. Specify `-lwsreg` on the `cc` command line to link with this library. See `libwsreg`(3LIB).

DEFINITIONS

A character is any bit pattern able to fit into a byte on the machine. In some international languages, however, a "character" may require more than one byte, and is represented in multi-bytes.

The null character is a character with value 0, conventionally represented in the C language as `\ 0`. A character array is a sequence of characters. A null-terminated character array (a *string*) is a sequence of characters, the last of which is the null character. The null string is a character array containing only the terminating null character. A null pointer is the value that is obtained by casting `0` into a pointer. C guarantees that this value will not match that of any legitimate pointer, so many functions that return pointers return `NULL` to indicate an error. The macro `NULL` is defined in `<stdio.h>`. Types of the form `size_t` are defined in the appropriate headers.

MT-Level of Libraries

See `attributes`(5) for descriptions of library MT-Levels.

FILES

INCDIR	usually `/usr/include`
LIBDIR	usually `/usr/lib` (32–bit) or `/usr/lib/sparcv9`(64–bit)

LIBDIR`/libc.so`

LIBDIR`/libc.a`

LIBDIR`/libgen.a`

LIBDIR`/libm.a`

LIBDIR`/libsfm.sa`

`/usr/lib/libc.so.1`

SEE ALSO

`ar`(1), `cc`(1B), `ld`(1), `fork`(2), `intro`(3), `stdio`(3C), `attributes` (5), `standards`(5)

Linker and Libraries Guide

Profiling Tools

ANSI C Programmer's Guide

DIAGNOSTICS

For functions that return floating-point values, error handling varies according to compilation mode. Under the `-Xt` (default) option to `cc`, these functions return the conventional values `0`, `±HUGE`, or `NaN` when the function is undefined for the given arguments or when the value is not representable. In the `-Xa` and `-Xc` compilation modes, `±HUGE_VAL` is returned instead of `±HUGE`. (`HUGE_VAL` and `HUGE` are defined in `math.h` to be infinity and the largest-magnitude single-precision number, respectively.)

NOTES ON MULTITHREADED APPLICATIONS

When compiling a multithreaded application, either the `_POSIX_C_SOURCE` or `_POSIX_PTHREAD_SEMANTICS` flag or the `-mt` option must be specified on the command line. This enables special definitions for functions only applicable to multithreaded applications. For POSIX.1c-conforming applications, define the `_POSIX_C_SOURCE` flag to be >= 199506L:

```
cc [flag...] file... -D_POSIX_C_SOURCE=199506L -lpthread
```

For POSIX behavior with the Solaris `fork()` and `fork1()` distinction, compile as follows:

```
cc [flag...] file... -D_POSIX_PTHREAD_SEMANTICS -lthread
```

For Solaris threads behavior, compile as follows:

```
cc - mt [ flag... ] file...
```

When building a singlethreaded application, the above *flag* arguments should be undefined.

Unsafe interfaces should be called only from the main thread to ensure the application's safety.

MT-Safe interfaces are denoted in the `ATTRIBUTES` section of the functions and libraries manual pages (see `attributes`(5)). If a manual page does not state explicitly that an interface is MT-Safe, the user should assume that the interface is unsafe.

REALTIME APPLICATIONS

Be sure to have set the environment variable `LD_BIND_NOW` to a non-null value to enable early binding. Refer to the "When Relocations are Processed" chapter in *Linker and Libraries Guide* for additional information.

NOTES

None of the functions, external variables, or macros should be redefined in the user's programs. Any other name may be redefined without affecting the behavior of other library functions, but such redefinition may conflict with a declaration in an included header.

The headers in *INCDIR* provide function prototypes (function declarations including the types of arguments) for most of the functions listed in this manual. Function prototypes allow the compiler to check for correct usage of these functions in the user's program. The `lint` program checker may also be used and will report discrepancies even if the headers are not included with `#include` statements. Definitions for Sections 2, 3C, and 3S are checked automatically. Other definitions can be included by using the `-l` option to `lint`. (For example, `-lm` includes definitions for `libm`.) Use of `lint` is highly recommended. See the `lint` chapter in *Performance Profiling Tools*.

Users should carefully note the difference between STREAMS and *stream*. STREAMS is a set of kernel mechanisms that support the development of network services and data communication drivers. It is composed of utility routines, kernel facilities, and a set of data structures. A *stream* is a file with its associated buffering. It is declared to be a pointer to a type `FILE` defined in `<stdio.h>`.

In detailed definitions of components, it is sometimes necessary to refer to symbolic names that are implementation-specific, but which are not necessarily expected to be accessible to an application program. Many of these symbolic names describe boundary conditions and system limits.

In this section, for readability, these implementation-specific values are given symbolic names. These names always appear enclosed in curly brackets to distinguish them from symbolic names of other implementation-specific constants that are accessible to application programs by headers. These names are not necessarily accessible to an application program through a header, although they may be defined in the documentation for a particular system.

In general, a portable application program should not refer to these symbolic names in its code. For example, an application program would not be expected to test the length of an argument list given to a routine to determine if it was greater than {`ARG_MAX`}.

Library Interfaces and Headers

NAME

acct – per-process accounting file format

SYNOPSIS

```
#include <sys/types.h>
#include <sys/acct.h>
```

DESCRIPTION

Files produced as a result of calling `acct`(2) have records in the form defined by `<sys/acct.h>`, whose contents are:

```
typedef ushort_t  comp_t;   /* pseudo "floating point" representation */
                            /* 3 bit base-8 exponent in the high */
                            /* order bits, and a 13-bit fraction */
                            /* in the low order bits. */

struct    acct
{
    char   ac_flag;    /* Accounting flag */
    char   ac_stat;    /* Exit status */
    uid_t  ac_uid;     /* Accounting user ID */
    gid_t  ac_gid;     /* Accounting group ID */
    dev_t  ac_tty;     /* control tty */
    time_t ac_btime;   /* Beginning time */
    comp_t ac_utime;   /* accounting user time in clock ticks */
    comp_t ac_stime;   /* accounting system time in clock ticks */
    comp_t ac_etime;   /* accounting total elapsed time in clock ticks */
    comp_t ac_mem;     /* memory usage in clicks (pages) */
    comp_t ac_io;      /* chars transferred by read/write */
    comp_t ac_rw;      /* number of block reads/writes */
    char   ac_comm[8]; /* command name */
};

/*
 * Accounting Flags
 */

#define AFORK   01    /* has executed fork, but no exec */
#define ASU     02    /* used super-user privileges */
#define ACCTF   0300  /* record type */
#define AEXPND  040   /* Expanded Record Type – default */
```

In `ac_flag`, the `AFORK` flag is turned on by each `fork` and turned off by an `exec`. The `ac_comm` field is inherited from the parent process and is reset by any `exec`. Each time the system charges the process with a clock tick, it also adds to `ac_mem` the current process size, computed as follows:

(data size) + (text size) / (number of in-core processes using text)

The value of `ac_mem / (ac_stime + ac_utime)` can be viewed as an approximation to the mean process size, as modified by text sharing.

The structure `tacct`, (which resides with the source files of the accounting commands), represents a summary of accounting statistics for the user id `ta_uid`. This structure is used by the accounting commands to report statistics based on user id.

```
/*
 *  total accounting (for acct period), also for day
 */
struct tacct {
   uid_t           ta_uid;       /* user id */
   char            ta_name[8];   /* login name */
   float           ta_cpu[2];    /* cum. cpu time in minutes, */
                                 /* p/np (prime/non-prime time) */
   float           ta_kcore[2];  /* cum. kcore-minutes, p/np */
   float           ta_con[2];    /* cum. connect time in minutes, p/np */
   float           ta_du;        /* cum. disk usage (blocks)*/
   long            ta_pc;        /* count of processes */
   unsigned short  ta_sc;        /* count of login sessions */
   unsigned short  ta_dc;        /* count of disk samples */
   unsigned short  ta_fee;       /* fee for special services */
};
```

The `ta_cpu`, `ta_kcore`, and `ta_con` members contain usage information pertaining to prime time and non-prime time hours. The first element in each array represents the time the resource was used during prime time hours. The second element in each array represents the time the resource was used during non-prime time hours. Prime time and non-prime time hours may be set in the `holidays` file (see `holidays`(4)).

The `ta_kcore` member is a cumulative measure of the amount of memory used over the accounting period by processes owned by the user with uid `ta_uid`. The amount shown represents kilobyte segments of memory used, per minute.

The `ta_con` member represents the amount of time the user was logged in to the system.

FILES

`/etc/acct/holidays` prime/non-prime time table

SEE ALSO

`acctcom`(1), `acct`(1M), `acctcon`(1M), `acctmerg`(1M), `acctprc`(1M), `acctsh`(1M), `prtacct`(1M), `runacct`(1M), `shutacct`(1M), `acct`(2), `exec`(2), `fork`(2)

NOTES

The `ac_mem` value for a short-lived command gives little information about the actual size of the command, because `ac_mem` may be incremented while a different command (for example, the shell) is being executed by the process.

NAME

aio – asynchronous input and output

SYNOPSIS

```
#include <aio.h>
```

DESCRIPTION

The <aio.h> header defines the aiocb structure which includes the following members:

```
int               aio_fildes       file descriptor
off_t             aio_offset       file offset
volatile void*    aio_buf          location of buffer
size_t            aio_nbytes       length of transfer
int               aio_reqprio      request priority offset
struct sigevent   aio_sigevent     signal number and value
int               aio_lio_opcode   operation to be performed
```

This header also includes the following constants:

```
AIO_CANCELED
AIO_NOTCANCELED
AIO_ALLDONE
LIO_WAIT
LIO_NOWAIT
LIO_READ
LIO_WRITE
LIO_NOP
```

SEE ALSO

lseek(2), read(2), write(2), fsync(3C)

NAME

ar – archive file format

SYNOPSIS

```
#include <ar.h>
```

DESCRIPTION

The archive command `ar` is used to combine several files into one. Archives are used mainly as libraries to be searched by the link editor `ld`.

Each archive begins with the archive magic string.

```
#define  ARMAG   "!<arch>\n"    /* magic string */
#define  SARMAG   8             /* length of magic string */
```

Following the archive magic string are the archive file members. Each file member is preceded by a file member header which is of the following format:

```
#define  ARFMAG   "`\n"         /* header trailer string */

struct  ar_hdr                  /* file member header */
{
    char    ar_name[16];        /* '/' terminated file member name */
    char    ar_date[12];        /* file member date */
    char    ar_uid[6]           /* file member user identification */
    char    ar_gid[6]           /* file member group identification */
    char    ar_mode[8]          /* file member mode (octal) */
    char    ar_size[10];        /* file member size */
    char    ar_fmag[2];         /* header trailer string */
};
```

All information in the file member headers is in printable ASCII. The numeric information contained in the headers is stored as decimal numbers (except for *ar_mode* which is in octal). Thus, if the archive contains printable files, the archive itself is printable.

If the file member name fits, the *ar_name* field contains the name directly, and is terminated by a slash (`/`) and padded with blanks on the right. If the member's name does not fit, *ar_name* contains a slash (`/`) followed by a decimal representation of the name's offset in the archive string table described below.

The *ar_date* field is the modification date of the file at the time of its insertion into the archive. Common format archives can be moved from system to system as long as the portable archive command `ar` is used.

Each archive file member begins on an even byte boundary; a newline is inserted between files if necessary. Nevertheless, the size given reflects the actual size of the file exclusive of padding.

Notice there is no provision for empty areas in an archive file.

Each archive that contains object files (see `a.out`(4)) includes an archive symbol table. This symbol table is used by the link editor `ld` to determine which archive members must be loaded during the link edit process. The archive symbol table (if it exists) is always the first file in the archive (but is never listed) and is automatically created and/or updated by `ar`.

The archive symbol table has a zero length name (that is, `ar_name[0]` is `'/'`), `ar_name[1]==' '`, etc.). All ''words'' in this symbol table have four bytes, using the machine-independent encoding shown below. All machines use the encoding described here for the symbol table, even if the machine's ''natural'' byte order is different.

```
                0       1       2       3
0x01020304      01      02      03      04
```

The contents of this file are as follows:

1. The number of symbols. Length: 4 bytes.
2. The array of offsets into the archive file. Length: 4 bytes * ''the number of symbols''.
3. The name string table. Length: *ar_size* – 4 bytes * (''the number of symbols'' + 1).

As an example, the following symbol table defines 4 symbols. The archive member at file offset 114 defines *name*. The archive member at file offset 122 defines *object*. The archive member at file offset 426 defines `function` and the archive member at file offset 434 defines *name2*.

Example Symbol Table

```
Offset     +0   +1   +2   +3
            _____________________
 0         |          4          | 4 offset entries
           |_____________________|
 4         |         114         | name
           |_____________________|
 8         |         122         | object
           |_____________________|
12         |         426         | function
           |_____________________|
16         |         434         | name2
           |_____________________|
20         |  n | a  | m  | e  |
           |____|____|____|____|
24         | \0 | o  | b  | j  |
           |____|____|____|____|
28         |  e | c  | t  | \0 |
           |____|____|____|____|
32         |  f | u  | n  | c  |
           |____|____|____|____|
36         |  t | i  | o  | n  |
           |____|____|____|____|
40         | \0 | n  | a  | m  |
           |____|____|____|____|
44         |  e | 2  | \0 |    |
           |____|____|____|____|
```

The string table contains exactly as many null terminated strings as there are elements in the offsets array. Each offset from the array is associated with the corresponding name from the string table (in order). The names in the string table are all the defined global symbols found in the common object files in the archive. Each offset is the location of the archive header for the associated symbol.

If some archive member's name is more than 15 bytes long, a special archive member contains a table of file names, each followed by a slash and a new-line. This string table member, if present, will precede all "normal" archive members. The special archive symbol table is not a "normal" member, and must be first if it exists. The `ar_name` entry of the string table's member header holds a zero length name `ar_name[0]=='/'`, followed by one trailing slash (`ar_name[1]=='/'`), followed by blanks (`ar_name[2]==' '`, etc.). Offsets into the string table begin at zero. Example *ar_name* values for short and long file names appear below.

```
Offset   +0   +1   +2   +3   +4   +5   +6   +7   +8   +9
          _____________________________________________
 0       | f  | i  | l  | e  | _  | n  | a  | m  | e  | _  |
         |____|____|____|____|____|____|____|____|____|____|
10       | s  | a  | m  | p  | l  | e  | /  | \n | l  | o  |
         |____|____|____|____|____|____|____|____|____|____|
20       | n  | g  | e  | r  | f  | i  | l  | e  | n  | a  |
         |____|____|____|____|____|____|____|____|____|____|
30       | m  | e  | x  | a  | m  | p  | l  | e  | /  | \n |
         |____|____|____|____|____|____|____|____|____|____|
```

Member Name	ar_name	
short-name	short-name/	Not in string table
file_name_sample	/0	Offset 0 in string table
longerfilenamexample	/18	Offset 18 in string table

SEE ALSO `ar`(1), `ld`(1), `strip`(1), `a.out`(4)

NOTES The `strip` utility will remove all archive symbol entries from the header. The archive symbol entries must be restored with the `-ts` options of the `ar` command before the archive can be used with the link editor `ld`.

NAME

dirent – file system independent directory entry

SYNOPSIS

```
#include <dirent.h>
```

DESCRIPTION

Different file system types may have different directory entries. The `dirent` structure defines a file system independent directory entry, which contains information common to directory entries in different file system types. A set of these structures is returned by the `getdents`(2) system call.

The `dirent` structure is defined:

```
struct  dirent {
        ino_t           d_ino;
        off_t           d_off;
        unsigned short  d_reclen;
        char            d_name[1];
};
```

The `d_ino` is a number which is unique for each file in the file system. The `d_off` entry contains a value which is interpretable only by the filesystem that generated it. It may be supplied as an offset to `lseek`(2) to find the entry following the current one in a directory. The field `d_name` is the beginning of the character array giving the name of the directory entry. This name is null terminated and may have at most `MAXNAMLEN` characters. This results in file system independent directory entries being variable length entities. The value of `d_reclen` is the record length of this entry. This length is defined to be the number of bytes between the current entry and the next one, so that the next structure will be suitably aligned.

SEE ALSO

`getdents`(2), `lseek`(2)

NAME

fcntl – file control options

SYNOPSIS

```
#include <fcntl.h>
```

DESCRIPTION

The `<fcntl.h>` header defines the following requests and arguments for use by the functions `fcntl`(2), `open`(2), and `openat`(2).

Values for *cmd* used by `fcntl()` (the following values are unique):

`F_DUPFD`	Duplicate file descriptor.
`F_DUP2FD`	Similar to `F_DUPFD`, but always returns *arg*.
`F_GETFD`	Get file descriptor flags.
`F_SETFD`	Set file descriptor flags.
`F_GETFL`	Get file status flags.
`F_SETFL`	Set file status flags.
`F_GETOWN`	Get process or process group ID to receive `SIGURG` signals.
`F_SETOWN`	Set process or process group ID to receive `SIGURG` signals.
`F_FREESP`	Free storage space associated with a section of the ordinary file *fildes*.
`F_GETLK`	Get record locking information.
`F_GETLK64`	Equivalent to `F_GETLK`, but takes a `struct flock64` argument rather than a `struct flock` argument.
`F_SETLK`	Set record locking information.
`F_SETLK64`	Equivalent to `F_SETLK`, but takes a `struct flock64` argument rather than a `struct flock` argument.
`F_SETLKW`	Set record locking information; wait if blocked.
`F_SETLKW64`	Equivalent to `F_SETLKW`, but takes a `struct flock64` argument rather than a `struct flock` argument.
`F_SHARE`	Set share reservation.
`F_UNSHARE`	Remove share reservation.

File descriptor flags used for `fcntl()`:

`FD_CLOEXEC`	Close the file descriptor upon execution of an `exec` function (see `exec`(2)).

Values for `l_type` used for record locking with `fcntl()` (the following values are unique):

`F_RDLCK`	Shared or read lock.
`F_UNLCK`	Unlock.

F_WRLCK	Exclusive or write lock.

Values for f_access used for share reservations with fcntl() (the following values are unique):

F_RDACC	Read-only share reservation.
F_WRACC	Write-only share reservation.
F_RWACC	Read and write share reservation.

Values for f_deny used for share reservations with fcntl() (the following values are unique):

F_COMPAT	Compatibility mode share reservation.
F_RDDNY	Deny other read access share reservations.
F_WRDNY	Deny other write access share reservations.
F_RWDNY	Deny other read or write access share reservations.
F_NODNY	Do not deny other read or write access share reservations.

File creation and assignment flags are used in the *oflag* argument by open() and openat(). All of these values are bitwise distinct:

O_CREAT	Create file if it does not exist.
O_EXCL	Exclusive use flag.
O_NOCTTY	Do not assign controlling tty.
O_TRUNC	Truncate flag.
O_XATTR	When opening a file, this flag affects the way in which relative paths are resolved by open() and openat(). With this flag set, the *path* argument is resolved as an extended attribute reference on either the current working directory (if open) or of the file referenced by the file descriptor argument of openat().

File status flags used for fcntl(), open(), and open():

O_APPEND	Set append mode.
O_NDELAY	Non-blocking mode.
O_NONBLOCK	Non-blocking mode (POSIX; see standards(5)).
O_DSYNC	Write I/O operations on the file descriptor complete as defined by synchronized I/O data integrity completion.
O_RSYNC	Read I/O operations on the file descriptor complete at the same level of integrity as specified by the the O_DSYNC and O_SYNC flags. If both O_DSYNC and O_RSYNC are set in *oflag*, all I/O operations on the file descriptor complete as defined by

synchronized I/O data integrity completion. If both `O_SYNC` and `O_RSYNC` are set in *oflag*, all I/O operations on the file descriptor complete as defined by synchronized I/O file integrity completion.

`O_SYNC` When opening a regular file, this flag affects subsequent writes. If set, each `write`(2) will wait for both the file data and file status to be physically updated. Write I/O operations on the file descriptor complete as defined by synchronized I/O file integrity completion.

Mask for use with file access modes:

`O_ACCMODE` Mask for file access modes.

File access modes used for `fcntl()`, `open()`, and `openat()`:

`O_RDONLY` Open for reading only.

`O_RDWR` Open for reading and writing.

`O_WRONLY` Open for writing only.

The following constants are used by system calls capable of resolving paths relative to a provided open file descriptor:

`AT_FDCWD` Special value to pass in place of a file descriptor to inform the called routine that relative path arguments should be resolved from the current working directory.

`AT_SYMLINK_NOFOLLOW` Flag passed to `fstatat`(2) and `fchownat`(2) to change the bahavior of these functions when they are given a file as an argument that is a symbolic link. In this case the functions operate on the symbolic link file rather than the file the link references.

`AT_REMOVEDIR` Flag passed to `unlinkat`(2) to tell it to assume that its path argument refers to a directory and to attempt to remove this directory.

The `flock` structure describes a file lock. It includes the following members:

```
short   l_type;   /* Type of lock */
short   l_whence; /* Flag for starting offset */
off_t   l_start;  /* Relative offset in bytes */
off_t   l_len;    /* Size; if 0 then until EOF */
long    l_sysid;  /* Returned with F_GETLK */
pid_t   l_pid;    /* Returned with F_GETLK */
```

The structure `fshare` describes a file share reservation. It includes the following members:

```
short   f_access; /* Type of reservation */
short   f_deny;   /* Type of reservations to deny */
long    f_id;     /* Process unique identifier */
```

SEE ALSO `creat`(2), `exec`(2), `fcntl`(2), `open`(2), `fdatasync`(3RT), `fsync`(3C), `fsattr`(5), `standards`(5)

NOTES Data is successfully transferred for a write operation to a regular file when the system ensures that all data written is readable on any subsequent open of the file (even one that follows a system or power failure) in the absence of a failure of the physical storage medium.

Data is successfully transferred for a read operation when an image of the data on the physical storage medium is available to the requesting process.

Synchronized I/O data integrity completion (see `fdatasync`(3RT)):

- For reads, the operation has been completed or diagnosed if unsuccessful. The read is complete only when an image of the data has been successfully transferred to the requesting process. If there were any pending write requests affecting the data to be read at the time that the synchronized read operation was requested, these write requests will be successfully transferred prior to reading the data.
- For writes, the operation has been completed or diagnosed if unsuccessful. The write is complete only when the data specified in the write request is successfully transferred, and all file system information required to retrieve the data is successfully transferred.

File attributes that are not necessary for data retrieval (access time, modification time, status change time) need not be successfully transferred prior to returning to the calling process.

Synchronized I/O file integrity completion (see `fsync`(3C)):

- Identical to a synchronized I/O data integrity completion with the addition that all file attributes relative to the I/O operation (including access time, modification time, status change time) will be successfully transferred prior to returning to the calling process.

NAME

floatingpoint – IEEE floating point definitions

SYNOPSIS

```
#include <floatingpoint.h>
```

DESCRIPTION

This file defines constants, types, and functions used to implement standard floating point according to ANSI/IEEE Std 754-1985. The functions are implemented in `libc`. The included header file `<sys/ieeefp.h>` defines certain types of interest to the kernel.

IEEE Rounding Modes

`fp_direction_type`	The type of the IEEE rounding direction mode. Note: the order of enumeration varies according to hardware.
`fp_precision_type`	The type of the IEEE rounding precision mode, which only applies on systems that support extended precision such as machines based on the Intel 80387 FPU or the 80486. `SIGFPE` handling:
`sigfpe_code_type`	The type of a `SIGFPE` code.
`sigfpe_handler_type`	The type of a user-definable `SIGFPE` exception handler called to handle a particular `SIGFPE` code.
`SIGFPE_DEFAULT`	A macro indicating the default `SIGFPE` exception handling, namely to perform the exception handling specified by the user, if any, and otherwise to dump core using `abort`(3C).
`SIGFPE_IGNORE`	A macro indicating an alternate `SIGFPE` exception handling, namely to ignore and continue execution.
`SIGFPE_ABORT`	A macro indicating an alternate `SIGFPE` exception handling, namely to abort with a core dump.

IEEE Exception Handling

`N_IEEE_EXCEPTION`	The number of distinct IEEE floating-point exceptions.
`fp_exception_type`	The type of the `N_IEEE_EXCEPTION` exceptions. Each exception is given a bit number.
`fp_exception_field_type`	The type intended to hold at least `N_IEEE_EXCEPTION` bits corresponding to the IEEE exceptions numbered by `fp_exception_type`. Thus `fp_inexact` corresponds to the least significant bit and `fp_invalid` to the fifth least significant bit. Note: some operations may set more

		than one exception.
IEEE Formats and Classification	`single`; `extended`; `quadruple`	Definitions of IEEE formats.
	`fp_class_type`	An enumeration of the various classes of IEEE values and symbols.
IEEE Base Conversion	The functions described under `floating_to_decimal`(3C) and `decimal_to_floating`(3C) satisfy not only the IEEE Standard, but also the stricter requirements of correct rounding for all arguments.	
	`DECIMAL_STRING_LENGTH`	The length of a `decimal_string`.
	`decimal_string`	The digit buffer in a `decimal_record`.
	`decimal_record`	The canonical form for representing an unpacked decimal floating-point number.
	`decimal_form`	The type used to specify fixed or floating binary to decimal conversion.
	`decimal_mode`	A struct that contains specifications for conversion between binary and decimal.
	`decimal_string_form`	An enumeration of possible valid character strings representing floating-point numbers, infinities, or NaNs.
FILES	`/usr/include/sys/ieeefp.h`	
SEE ALSO	`abort`(3C), `decimal_to_floating`(3C), `econvert`(3C), `floating_to_decimal`(3C), `sigfpe`(3C), `string_to_decimal`(3C), `strtod`(3C)	

NAME

in – Internet Protocol family

SYNOPSIS

```
#include <netinet/in.h>
```

DESCRIPTION

The <netinet/in.h> header defines the following types through typedef:

in_port_t	An unsigned integral type of exactly 16 bits.
in_addr_t	An unsigned integral type of exactly 32 bits. The <netinet/in.h> header defines the in_addr structure that includes the following member:

The <netinet/in.h> header defines the in_addr structure that includes the following member:

```
in_addr_t       s_addr
```

The <netinet/in.h> header defines the type sa_family_t as described in socket(3HEAD).

The <netinet/in.h> header defines the following macros for use as values of the *level* argument of getsockopt() and setsockopt():

IPPROTO_IP	Dummy for IP
IPPROTO_ICMP	Control message protocol
IPPROTO_TCP	TCP
IPPROTO_UDP	User datagram protocol The <netinet/in.h> header defines the following macros for use as destination addresses for connect(), sendmsg(), and sendto():
INADDR_ANY	Local host address
INADDR_BROADCAST	Broadcast address

The <netinet/in.h> header defines the sockaddr_in structure that is used to store addresses for the Internet protocol family. Values of this type must be cast to struct sockaddr for use with the socket interfaces.

Default

For applications that do not require standard-conforming behavior (those that use the socket interfaces described in section (3SOCKET) of the reference manual; see Intro(3) and standards(5)), the <netinet/in.h> header defines the sockaddr_in structure that includes the following members:

```
sa_family_t     sin_family
in_port_t       sin_port
struct in_addr  sin_addr
char            sin_zero[8]
```

Standard conforming

For applications that require standard-conforming behavior (those that use the socket interfaces described in section (3XNET) of the reference manual; see `Intro`(3) and `standards`(5)), the `<netinet/in.h>` header defines the `sockaddr_in` structure that includes the following members:

```
sa_family_t     sin_family
in_port_t       sin_port
struct in_addr  sin_addr
unsigned char   sin_zero[8]
```

SEE ALSO

`Intro`(3), `connect`(3SOCKET), `connect`(3XNET), `getsockopt`(3SOCKET), `getsockopt`(3XNET), `sendmsg`(3SOCKET), `sendmsg`(3XNET), `sendto`(3SOCKET), `sendto`(3XNET), `setsockopt`(3SOCKET), `setsockopt`(3XNET), `socket`(3HEAD), `standards`(5)

NAME

inet – definitions for internet operations

SYNOPSIS

```
#include <arpa/inet.h>
```

DESCRIPTION

The <arpa/inet.h> header defines the type in_port_t, the type in_addr_t, and the in_addr structure, as described in in(3HEAD).

Inclusion of the <arpa/inet.h> header may also make visible all symbols from in(3HEAD).

The following are declared as functions, and may also be defined as macros:

```
in_addr_t       inet_addr(const char *);
in_addr_t       inet_lnaof(struct in_addr);
struct in_addr  inet_makeaddr(in_addr_t, in_addr_t);
in_addr_t       inet_netof(struct in_addr);
in_addr_t       inet_network(const char *);
char            *inet_ntoa(struct in_addr);
```

Default

For applications that do not require standard-conforming behavior (those that use the socket interfaces described in section 3N of the reference manual; see Intro(3) and standards(5)), the following may be declared as functions, or defined as macros, or both:

```
uint32_t    htonl(uint32_t);
uint16_t    htons(uint16_t);
uint32_t    ntohl(uint32_t);
uint16_t    ntohs(uint16_t);
```

Standard conforming

For applications that require standard-conforming behavior (those that use the socket interfaces described in section 3XN of the reference manual; see Intro(3) and standards(5)), the following may be declared as functions, or defined as macros, or both:

```
in_addr_t    htonl(in_addr_t);
in_port_t    htons(in_port_t);
in_addr_t    ntohl(in_addr_t);
in_port_t    ntohs(in_port_t);
```

SEE ALSO

Intro(3), htonl(3SOCKET), htonl(3XNET), inet_addr(3SOCKET), inet_addr(3XNET), in(3HEAD), standards(5)

NAME

langinfo – language information constants

SYNOPSIS

```
#include <langinfo.h>
```

DESCRIPTION

The `<langinfo.h>` header contains the constants used to identify items of `langinfo` data (see `nl_langinfo`(3C)). The type of the constant, `nl_item`, is defined as described in `<nl_types.h>`.

The following constants are defined. The entries under Category indicate in which `setlocale`(3C) category each item is defined.

Constant	Category	Meaning
CODESET	LC_CTYPE	codeset name
D_T_FMT	LC_TIME	string for formatting date and time
D_FMT	LC_TIME	date format string
T_FMT	LC_TIME	time format string
T_FMT_AMPM	LC_TIME	a.m. or p.m. time format string
AM_STR	LC_TIME	ante-meridiem affix
PM_STR	LC_TIME	post-meridiem affix
DAY_1	LC_TIME	name of the first day of the week (for example, Sunday)
DAY_2	LC_TIME	name of the second day of the week (for example, Monday)
DAY_3	LC_TIME	name of the third day of the week (for example, Tuesday)
DAY_4	LC_TIME	name of the fourth day of the week (for example, Wednesday)
DAY_5	LC_TIME	name of the fifth day of the week (for example, Thursday)
DAY_6	LC_TIME	name of the sixth day of the week (for example, Friday)
DAY_7	LC_TIME	name of the seventh day of the week (for example, Saturday)
ABDAY_1	LC_TIME	abbreviated name of the first day of the week
ABDAY_2	LC_TIME	abbreviated name of the second day of the week

Constant	Category	Meaning
ABDAY_3	LC_TIME	abbreviated name of the third day of the week
ABDAY_4	LC_TIME	abbreviated name of the fourth day of the week
ABDAY_5	LC_TIME	abbreviated name of the fifth day of the week
ABDAY_6	LC_TIME	abbreviated name of the seventh day of the week
ABDAY_7	LC_TIME	abbreviated name of the seventh day of the week
MON_1	LC_TIME	name of the first month of the year
MON_2	LC_TIME	name of the second month
MON_3	LC_TIME	name of the third month
MON_4	LC_TIME	name of the fourth month
MON_5	LC_TIME	name of the fifth month
MON_6	LC_TIME	name of the sixth month
MON_7	LC_TIME	name of the seventh month
MON_8	LC_TIME	name of the eighth month
MON_9	LC_TIME	name of the ninth month
MON_10	LC_TIME	name of the tenth month
MON_11	LC_TIME	name of the eleventh month
MON_12	LC_TIME	name of the twelfth month
ABMON_1	LC_TIME	abbreviated name of the first month
ABMON_2	LC_TIME	abbreviated name of the second month
ABMON_3	LC_TIME	abbreviated name of the third month
ABMON_4	LC_TIME	abbreviated name of the fourth month
ABMON_5	LC_TIME	abbreviated name of the fifth month
ABMON_6	LC_TIME	abbreviated name of the sixth month
ABMON_7	LC_TIME	abbreviated name of the seventh month
ABMON_8	LC_TIME	abbreviated name of the eighth month
ABMON_9	LC_TIME	abbreviated name of the ninth month

Constant	Category	Meaning
ABMON_10	LC_TIME	abbreviated name of the tenth month
ABMON_11	LC_TIME	abbreviated name of the eleventh month
ABMON_12	LC_TIME	abbreviated name of the twelfth month
ERA	LC_TIME	era description segments
ERA_D_FMT	LC_TIME	era date format string
ERA_D_T_FMT	LC_TIME	era date and time format string
ERA_T_FMT	LC_TIME	era time format string
ALT_DIGITS	LC_TIME	alternative symbols for digits
RADIXCHAR	LC_NUMERIC	radix character
THOUSEP	LC_NUMERIC	separator for thousands
YESEXPR	LC_MESSAGES	affirmative response expression
NOEXPR	LC_MESSAGES	negative response expression
YESSTR	LC_MESSAGES	affirmative response for yes/no queries
NOSTR	LC_MESSAGES	negative response ro yes/no queries
CRNCYSTR	LC_MONETARY	local currency symbol, preceded by '-' if the symbol sould appear before the value, '+' if the symbol should appear after the value, or '.' if the symbol should replace the radix character

If the locale's values for `p_cs_precedes` and `n_cs_precedes` do not match, the value of `nl_langinfo(CRNCYSTR)` is unspecified.

The `<langinfo.h>` header declares the following as a function:

```
char *nl_langinfo(nl_item);
```

Inclusion of `<langinfo.h>` header may also make visible all symbols from `<nl_types.h>`.

USAGE

Wherever possible, users are advised to use functions compatible with those in the ISO C standard to access items of `langinfo` data. In particular, the `strftime`(3C) function should be used to access date and time information defined in category `LC_TIME`. The `localeconv`(3C) function should be used to access information corresponding to `RADIXCHAR`, `THOUSEP`, and `CRNCYSTR`.

SEE ALSO

`mkmsgs`(1), `localeconv`(3C), `nl_langinfo`(3C), `nl_types`(3HEAD), `setlocale`(3C), `strftime`(3C), `standards`(5)

NAME

libadm – general administrative library

SYNOPSIS

```
cc [ flag ... ] file ... -ladm [ library ... ]
```

DESCRIPTION

Functions in this library provide device management, VTOC handling, regular expressions, and packaging routines.

INTERFACES

The shared object `libadm.so.1` provides the public interfaces defined below. See `intro`(3) for additional information on shared object interfaces.

`advance`	`circf`
`compile`	`devattr`
`devfree`	`devreserv`
`get_ABI_namelngth`	`getdev`
`getdgrp`	`getvol`
`listdev`	`listdgrp`
`loc1`	`loc2`
`locs`	`nbra`
`pkgdir`	`pkginfo`
`pkgnmchk`	`pkgparam`
`read_vtoc`	`reservdev`
`sed`	`set_ABI_namelngth`
`step`	`write_vtoc`

FILES

`/usr/lib/libadm.a`	archive library
`/usr/lib/libadm.so.1`	shared object
`/usr/lib/64/libadm.so.1`	64-bit shared object

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsl, SUNWarc (32-bit) SUNWcslx (64-bit)
MT-Level	Unsafe

SEE ALSO

`pvs`(1), `intro`(3), `read_vtoc`(3EXT), `attributes`(5), `regexp`(5)

NAME

libaio – asynchronous I/O library

SYNOPSIS

```
cc [ flag . . . ] file . . . -laio [ library . . . ]
```

DESCRIPTION

Functions in this library provide routines for asynchronous I/O.

INTERFACES

The shared object `libaio.so.1` provides the public interfaces defined below. See `intro`(3) for additional information on shared object interfaces.

`aiocancel`	`aioread`	`aiowait`	`aiowrite`
`assfail`	`close`	`fork`	`sigaction`
`sigignore`	`signal`	`sigset`	

The following interfaces are unique to the 32-bit version of this library:

`aioread64`	`aiowrite64`

FILES

`/usr/lib/libaio.so.1`	shared object
`/usr/lib/64/libaio.so.1`	64-bit shared object

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsl (32-bit) SUNWcslx (64-bit)
MT-Level	Safe

SEE ALSO

`pvs`(1), `intro`(2), `intro`(3), `aiocancel`(3AIO), `aioread`(3AIO), `aiowait`(3AIO), `aiowrite`(3AIO), `attributes`(5)

NAME

libbsdmalloc – memory allocator interface library

SYNOPSIS

```
cc [ flag . . . ] file . . . -lbsdmalloc [ library . . . ]
#include <stdlib.h>
```

DESCRIPTION

Functions in this library provide a collection of `malloc` routines that use BSD semantics.

INTERFACES

The shared object `libbsdmalloc.so.1` provides the public interfaces defined below. See `intro`(3) for additional information on shared object interfaces.

```
free                    malloc                  realloc
```

FILES

`/usr/lib/libbsdmalloc.a`
archive library

`/usr/lib/libbsdmalloc.so.1`
shared object

`/usr/lib/64/libbsdmalloc.so.1`
64–bit shared object

ATTRIBUTES

See `attributes`(5) for description of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsl, SUNWarc (32-bit) SUNWcslx (64-bit)
MT Level	Unsafe

SEE ALSO

`pvs`(1), `intro`(3), `bsdmalloc`(3MALLOC), `attributes`(5)

NAME

libbsm – basic security library

SYNOPSIS

`cc [ flag . . . ] file . . . -lbsm [ library . . . ]`

DESCRIPTION

Functions in this library provide basic security, library object reuse, and auditing.

INTERFACES

The shared object `libbsm.so.1` provides the public interfaces defined below. See `intro`(3) for additional information on shared object interfaces.

au_close	au_open
au_preselect	au_to_arg
au_to_arg32	au_to_arg64
au_to_attr	au_to_cmd
au_to_data	au_to_groups
au_to_in_addr	au_to_ipc
au_to_iport	au_to_me
au_to_newgroups	au_to_opaque
au_to_path	au_to_process
au_to_return	au_to_return32
au_to_return64	au_to_socket
au_to_subject	au_to_subject_ex
au_to_text	au_user_mask
au_write	audit
auditon	auditsvc
endac	endauclass
endauevent	endauuser
getacdir	getacflg
getacmin	getacna
getauclassent	getauclassent_r
getauclassnam	getauclassnam_r
getaudit	getaudit_addr
getauditflagsbin	getauditflagschar
getauevent	getauevent_r

getauevnam	getauevnam_r
getauevnonam	getauevnum
getauevnum_r	getauid
getauuserent	getauuserent_r
getauusernam	getauusernam_r
getfauditflags	setac
setauclass	setauclassfile
setaudit	setaudit_addr
setauevent	setaueventfile
setauid	setauuser
setauuserfile	testac

FILES

/usr/lib/libbsm.a	archive library
/usr/lib/libbsm.so.1	shared object
/usr/lib/64/libbsm.so.1	64-bit shared object

ATTRIBUTES

See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsl, SUNWarc (32-bit) SUNWcslx (64-bit)
MT-Level	See individual man page for each function.

SEE ALSO

pvs(1), intro(3), attributes(5)

NAME

libc – C library

SYNOPSIS

`cc [ flag . . . ] file . . . -lc [ library . . . ]`

DESCRIPTION

Functions in this library provide various facilities defined by System V, ANSI C, POSIX, and so on. See `standards`(5). In addition, those facilities previously defined in the internationalization and the wide-character libraries are now defined in this library.

Many features in this library are implemented upon dynamic linking. Some of these features are not implemented in the archive version.

INTERFACES

The shared object `libc.so.1` provides the public interfaces defined below. See `intro`(3) for additional information on shared object interfaces.

`____loc1`	`___errno`
`__assert`	`__builtin_alloca`
`__ctype`	`__fbufsize`
`__filbuf`	`__flbf`
`__flsbuf`	`__flt_rounds`
`__fpending`	`__fpurge`
`__freadable`	`__freading`
`__fsetlocking`	`__fwritable`
`__fwriting`	`__huge_val`
`__iob`	`__loc1`
`__major`	`__makedev`
`__minor`	`__nsw_extended_action`
`__nsw_freeconfig`	`__nsw_getconfig`
`__posix_asctime_r`	`__posix_ctime_r`
`__posix_getgrgid_r`	`__posix_getgrnam_r`
`__posix_getlogin_r`	`__posix_getpwnam_r`
`__posix_getpwuid_r`	`__posix_sigwait`
`__posix_ttyname_r`	`__priocntl`
`__priocntlset`	`__pthread_cleanup_pop`
`__pthread_cleanup_push`	`__sysconf_xpg5`
`__xpg4`	`__xpg4_putmsg`

__xpg4_putpmsg	_access
_acct	_addseverity
_alarm	_altzone
_assert	_catclose
_catgets	_catopen
_cfgetispeed	_cfgetospeed
_cfsetispeed	_cfsetospeed
_chdir	_chmod
_chown	_chroot
_cleanup	_close
_closedir	_closefrom
_creat	_crypt
_ctermid	_ctype
_cuserid	_daylight
_dup	_dup2
_encrypt	_environ
_execl	_execle
_execlp	_execv
_execve	_execvp
_exit	_exithandle
_fattach	_fchdir
_fchmod	_fchown
_fcntl	_fdetach
_fdopen	_fdwalk
_filbuf	_fileno
_flsbuf	_flushlbf
_fmtmsg	_fork
_fpathconf	_fstat
_fstatvfs	_fsync
_ftok	_getacct

_getcontext	_getcwd
_getdate	_getdate_err
_getdate_err_addr	_getegid
_geteuid	_getexecname
_getgid	_getgrgid
_getgrnam	_getgroups
_getitimer	_getlogin
_getmsg	_getopt
_getpass	_getpgid
_getpgrp	_getpid
_getpmsg	_getppid
_getprojid	_getpwnam
_getpwuid	_getrlimit
_getsid	_getsubopt
_gettaskid	_gettimeofday
_gettxt	_getuid
_getw	_grantpt
_hcreate	_hdestroy
_hsearch	_initgroups
_insque	_iob
_ioctl	_isascii
_isastream	_isatty
_isnan	_isnand
_kill	_lchown
_lfind	_link
_lockf	_longjmp
_lsearch	_lseek
_lstat	_lwp_cond_broadcast
_lwp_cond_reltimedwait	_lwp_cond_signal
_lwp_cond_timedwait	_lwp_cond_wait

_lwp_continue	_lwp_create
_lwp_detach	_lwp_exit
_lwp_getprivate	_lwp_info
_lwp_kill	_lwp_makecontext
_lwp_mutex_lock	_lwp_mutex_trylock
_lwp_mutex_unlock	_lwp_self
_lwp_sema_init	_lwp_sema_post
_lwp_sema_trywait	_lwp_sema_wait
_lwp_setprivate	_lwp_suspend
_lwp_suspend2	_lwp_wait
_makecontext	_memccpy
_memcntl	_mkdir
_mkfifo	_mknod
_mkstemp	_mktemp
_mlock	_mmap
_modf	_monitor
_mount	_mprotect
_msgctl	_msgget
_msgids	_msgrcv
_msgsnap	_msgsnd
_msync	_munlock
_munmap	_mutex_held
_mutex_lock	_nextafter
_nftw	_nice
_nl_langinfo	_nsc_trydoorcall
_nss_XbyY_buf_alloc	_nss_XbyY_buf_free
_nss_netdb_aliases	_ntp_adjtime
_ntp_gettime	_numeric
_open	_opendir
_pathconf	_pause

_pclose	_pipe
_poll	_popen
_profil	_ptrace
_ptsname	_putacct
_putenv	_putmsg
_putpmsg	_putw
_read	_readdir
_readlink	_readv
_remque	_rename
_resolvepath	_rewinddir
_rmdir	_rw_read_held
_rw_write_held	_rwlock_destroy
_sbrk	_scalb
_seekdir	_sema_destroy
_sema_held	_semctl
_semget	_semids
_semop	_semtimedop
_setcontext	_setgid
_setgroups	_setitimer
_setjmp	_setkey
_setpgid	_setpgrp
_setrlimit	_setsid
_settaskid	_setuid
_shmat	_shmctl
_shmdt	_shmget
_shmids	_sibuf
_sigaction	_sigaddset
_sigaltstack	_sigdelset
_sigemptyset	_sigfillset
_sighold	_sigignore

_sigismember	_siglongjmp
_sigpause	_sigpending
_sigprocmask	_sigrelse
_sigsend	_sigsendset
_sigset	_sigsetjmp
_sigsuspend	_sleep
_sobuf	_stat
_statvfs	_stime
_strdup	_swab
_swapcontext	_symlink
_sync	_sys_buslist
_sys_cldlist	_sys_fpelist
_sys_illlist	_sys_segvlist
_sys_siginfolistp	_sys_siglist
_sys_siglistn	_sys_siglistp
_sys_traplist	_syscall
_sysconf	_sysinfo
_syslog	_tcdrain
_tcflow	_tcflush
_tcgetattr	_tcgetpgrp
_tcgetsid	_tcsendbreak
_tcsetattr	_tcsetpgrp
_tdelete	_tell
_telldir	_tempnam
_tfind	_time
_times	_timezone
_toascii	_tolower
_toupper	_tsearch
_ttyname	_twalk
_tzname	_tzset

_ulimit	_umask
_umount	_umount2
_uname	_unlink
_unlockpt	_utime
_wait	_waitid
_waitpid	_wracct
_write	_writev
_xftw	a64l
abort	abs
access	acct
acl	addsev
addseverity	adjtime
alarm	altzone
ascftime	asctime
asctime_r	atexit
atof	atoi
atol	atoll
attropen	basename
bcmp	bcopy
bindtextdomain	bind_textdomain_codeset
brk	bsd_signal
bsearch	btowc
bzero	calloc
catclose	catgets
catopen	cfgetispeed
cfgetospeed	cfree
cfsetispeed	cfsetospeed
cftime	chdir
chmod	chown
chroot	clearerr

clock	close
closedir	closefrom
closelog	cond_broadcast
cond_destroy	cond_init
cond_signal	cond_timedwait
cond_wait	confstr
creat	crypt
csetcol	csetlen
ctermid	ctermid_r
ctime	ctime_r
cuserid	daylight
dcgettext	dcngettext
dbm_clearerr	dbm_close
dbm_delete	dbm_error
dbm_fetch	dbm_firstkey
dbm_nextkey	dbm_open
dbm_store	dcgettext
decimal_to_double	decimal_to_extended
decimal_to_quadruple	decimal_to_single
dgettext	difftime
directio	dirname
div	double_to_decimal
drand48	dup
dup2	econvert
ecvt	encrypt
endgrent	endnetgrent
endpwent	endspent
endusershell	endutent
endutxent	environ
erand48	errno

euccol	euclen
eucscol	execl
execle	execlp
execv	execve
execvp	exit
exportfs	extended_to_decimal
facl	fattach
fchdir	fchmod
fchown	fchownat
fchroot	fclose
fcntl	fconvert
fcvt	fdetach
fdopen	fdopendir
fdwalk	feof
ferror	fflush
ffs	fgetc
fgetgrent	fgetgrent_r
fgetpos	fgetpwent
fgetpwent_r	fgets
fgetspent	fgetspent_r
fgetwc	fgetws
file_to_decimal	fileno
finite	flockfile
fmtmsg	fnmatch
fopen	fork
fork1	fpathconf
fpclass	fpgetmask
fpgetround	fpgetsticky
fprintf	fpsetmask
fpsetround	fpsetsticky

fputc	fputs
fputwc	fputws
fread	free
freopen	frexp
fscanf	fseek
fseeko	fsetpos
fstat	fstatat
fstatfs	fstatvfs
fsync	ftell
ftello	ftime
ftok	ftruncate
ftrylockfile	ftw
func_to_decimal	funlockfile
futimesat	fwide
fwprintf	fwrite
fwscanf	gconvert
gcvt	getacct
getc	getc_unlocked
getchar	getchar_unlocked
getcontext	getcpuid
getcwd	getdate
getdate_err	getdents
getdtablesize	getegid
getenv	geteuid
getexecname	getextmntent
getgid	getgrent
getgrent_r	getgrgid
getgrgid_r	getgrnam
getgrnam_r	getgroups
gethomelgroup	gethostid

gethostname	gethrtime
gethrvtime	getitimer
getloadavg	getlogin
getlogin_r	getmntany
getmntent	getmsg
getnetgrent	getnetgrent_r
getopt	getpagesize
getpagesizes	getpass
getpassphrase	getpgid
getpgrp	getpid
getpmsg	getppid
getpriority	getprojid
getpw	getpwent
getpwent_r	getpwnam
getpwnam_r	getpwuid
getpwuid_r	getrctl
getrlimit	getrusage
gets	getsid
getspent	getspent_r
getspnam	getspnam_r
getsubopt	gettaskid
gettext	gettimeofday
gettxt	getuid
getusershell	getutent
getutid	getutline
getutmp	getutmpx
getutxent	getutxid
getutxline	getvfsany
getvfsent	getvfsfile
getvfsspec	getw

getwc	getwchar
getwd	getwidth
getws	glob
globfree	gmtime
gmtime_r	grantpt
gsignal	hasmntopt
hcreate	hdestroy
hsearch	iconv
iconv_close	iconv_open
index	initgroups
initstate	innetgr
insque	ioctl
isaexec	isalnum
isalpha	isascii
isastream	isatty
iscntrl	isdigit
isenglish	isgraph
isideogram	islower
isnan	isnand
isnanf	isnumber
isphonogram	isprint
ispunct	issetugid
isspace	isspecial
isupper	iswalnum
iswalpha	iswcntrl
iswctype	iswdigit
iswgraph	iswlower
iswprint	iswpunct
iswspace	iswupper
iswxdigit	isxdigit

jrand48	kill
killpg	l64a
labs	ladd
lchown	lckpwdf
lcong48	ldexp
ldivide	lexp10
lfind	lfmt
link	llabs
lldiv	llog10
llseek	lltostr
localeconv	localtime
localtime_r	lockf
logb	lone
longjmp	lrand48
lsearch	lseek
lshiftl	lstat
lsub	lten
lzero	madvise
makecontext	makeutx
malloc	mblen
mbrlen	mbrtowc
mbsinit	mbsrtowcs
mbstowcs	mbtowc
memalign	memccpy
memchr	memcmp
memcntl	memcpy
meminfo	memmove
memset	mincore
mkdir	mkfifo
mknod	mkstemp

mktemp	mktime
mlock	mlockall
mmap	modctl
modf	modff
modutx	monitor
mount	mprotect
mrand48	msgctl
msgget	msgids
msgrcv	msgsnap
msgsnd	msync
munlock	munlockall
munmap	mutex_destroy
mutex_init	mutex_lock
mutex_trylock	mutex_unlock
nextafter	nfs_getfh
nftw	ngettext
nice	nl_langinfo
nrand48	nss_default_finders
nss_delete	nss_endent
nss_getent	nss_search
nss_setent	ntp_adjtime
ntp_gettime	open
openat	opendir
openlog	optarg
opterr	optind
optopt	p_online
pathconf	pause
pclose	pcsample
perror	pfmt
pipe	plock

poll	popen
pread	printf
printstack	processor_bind
processor_info	profil
pset_assign	pset_bind
pset_create	pset_destroy
pset_getattr	pset_getloadavg
pset_info	pset_list
pset_setattr	psiginfo
psignal	pthread_atfork
pthread_attr_destroy	pthread_attr_getdetachstate
pthread_attr_getguardsize	pthread_attr_getinheritsched
pthread_attr_getschedparam	pthread_attr_getschedpolicy
pthread_attr_getscope	pthread_attr_getstackaddr
pthread_attr_getstacksize	pthread_attr_init
pthread_attr_setdetachstate	pthread_attr_setguardsize
pthread_attr_setinheritsched	pthread_attr_setschedparam
pthread_attr_setschedpolicy	pthread_attr_setscope
pthread_attr_setstackaddr	pthread_attr_setstacksize
pthread_cancel	pthread_cond_broadcast
pthread_cond_destroy	pthread_cond_init
pthread_cond_reltimedwait_np	pthread_cond_signal
pthread_cond_timedwait	pthread_cond_wait
pthread_condattr_destroy	pthread_condattr_getpshared
pthread_condattr_init	pthread_condattr_setpshared
pthread_create	pthread_detach
pthread_equal	pthread_exit
pthread_getconcurrency	pthread_getschedparam
pthread_getspecific	pthread_join
pthread_key_create	pthread_key_delete

pthread_kill	pthread_mutex_destroy
pthread_mutex_getprioceiling	pthread_mutex_init
pthread_mutex_lock	pthread_mutex_setprioceiling
pthread_mutex_trylock	pthread_mutex_unlock
pthread_mutexattr_destroy	pthread_mutexattr_ getprioceiling
pthread_mutexattr_getprotocol	pthread_mutexattr_getpshared
pthread_mutexattr_gettype	pthread_mutexattr_init
pthread_mutexattr_setprioceiling	pthread_mutexattr_setprotocol
pthread_mutexattr_setpshared	pthread_mutexattr_settype
pthread_once	pthread_rwlock_destroy
pthread_rwlock_init	pthread_rwlock_rdlock
pthread_rwlock_tryrdlock	pthread_rwlock_trywrlock
pthread_rwlock_unlock	pthread_rwlock_wrlock
pthread_rwlockattr_destroy	pthread_rwlockattr_getpshared
pthread_rwlockattr_init	pthread_rwlockattr_setpshared
pthread_self	pthread_setcancelstate
pthread_setcanceltype	pthread_setconcurrency
pthread_setschedparam	pthread_setspecific
pthread_sigmask	pthread_testcancel
ptrace	ptsname
putacct	putc
putc_unlocked	putchar
putchar_unlocked	putenv
putmsg	putpmsg
putpwent	puts
putspent	pututline
pututxline	putw
putwc	putwchar
putws	pwrite
qeconvert	qecvt

qfconvert	qfcvt
qgconvert	qgcvt
qsort	quadruple_to_decimal
raise	rand
rand_r	random
rctl_walk	rctlblk_get_enforced_value
rctlblk_get_firing_time	rctlblk_get_global_action
rctlblk_get_global_flags	rctlblk_get_local_action
rctlblk_get_local_flags	rctlblk_get_privilege
rctlblk_get_recipient_pid	rctlblk_get_value
rctlblk_set_local_action	rctlblk_set_local_flags
rctlblk_set_privilege	rctlblk_set_value
rctlblk_set_value	re_comp
re_exec	read
readdir	readdir_r
readlink	readv
realloc	realpath
reboot	regcmp
regcomp	regerror
regex	regexec
regfree	remove
remque	rename
renameat	resetmnttab
resolvepath	rewind
rewinddir	rindex
rmdir	rw_rdlock
rw_read_held	rw_tryrdlock
rw_trywrlock	rw_unlock
rw_write_held	rw_wrlock
rwlock_destroy	rwlock_init

sbrk	scalb
scanf	seconvert
seed48	seekdir
select	sema_destroy
sema_held	sema_init
sema_post	sema_trywait
sema_wait	semctl
semget	semids
semop	semtimedop
setbuf	setbuffer
setcat	setcontext
setegid	seteuid
setgid	setgrent
setgroups	sethostname
setitimer	setjmp
setkey	setlabel
setlinebuf	setlocale
setlogmask	setnetgrent
setpgid	setpgrp
setpriority	setpwent
setrctl	setregid
setreuid	setrlimit
setsid	setspent
setstate	settaskid
settimeofday	setuid
setusershell	setutent
setutxent	setvbuf
sfconvert	sgconvert
shmat	shmctl
shmdt	shmget

shmids	sig2str
sigaction	sigaddset
sigaltstack	sigdelset
sigemptyset	sigfillset
sigfpe	sighold
sigignore	siginterrupt
sigismember	siglongjmp
signal	sigpause
sigpending	sigprocmask
sigrelse	sigsend
sigsendset	sigset
sigsetjmp	sigstack
sigsuspend	sigwait
single_to_decimal	sleep
snprintf	sprintf
srand	srand48
srandom	sscanf
ssignal	stat
statfs	statvfs
stime	str2sig
strcasecmp	strcat
strchr	strcmp
strcoll	strcpy
strcspn	strdup
strerror	strfmon
strftime	string_to_decimal
strlcat	strlcpy
strlen	strncasecmp
strncat	strncmp
strncpy	strpbrk

strptime	strrchr
strsignal	strspn
strstr	strtod
strtok	strtok_r
strtol	strtoll
strtoul	strtoull
strtows	strxfrm
swab	swapcontext
swapctl	swprintf
swscanf	symlink
sync	sync_instruction_memory
syscall	sysconf
sysfs	sysinfo
syslog	system
tcdrain	tcflow
tcflush	tcgetattr
tcgetpgrp	tcgetsid
tcsendbreak	tcsetattr
tcsetpgrp	tdelete
tell	telldir
tempnam	textdomain
tfind	thr_continue
thr_create	thr_exit
thr_getconcurrency	thr_getprio
thr_getspecific	thr_join
thr_keycreate	thr_kill
thr_main	thr_min_stack
thr_self	thr_setconcurrency
thr_setprio	thr_setspecific
thr_sigsetmask	thr_stksegment

thr_suspend	thr_yield
time	times
timezone	tmpfile
tmpnam	tmpnam_r
toascii	tolower
toupper	towctrans
towlower	towupper
truncate	tsearch
ttyname	ttyname_r
ttyslot	twalk
tzname	tzset
uadmin	ualarm
ulckpwdf	ulimit
ulltostr	umask
umount	umount2
uname	ungetc
ungetwc	unlink
unlinkat	unlockpt
unordered	updwtmp
updwtmpx	usleep
ustat	utime
utimes	utmpname
utmpxname	valloc
vfork	vfprintf
vfscanf	vfwprintf
vfwscanf	vhangup
vlfmt	vpfmt
vprintf	vscanf
vsnprintf	vsprintf
vsscanf	vswprintf

vswscanf	vsyslog
vwprintf	vwscanf
wait	wait3
wait4	waitid
waitpid	walkcontext
watoll	wcrtomb
wcscat	wcschr
wcscmp	wcscoll
wcscpy	wcscspn
wcsftime	wcslen
wcsncat	wcsncmp
wcsncpy	wcspbrk
wcsrchr	wcsrtombs
wcsspn	wcsstr
wcstod	wcstok
wcstol	wcstombs
wcstoul	wcswcs
wcswidth	wcsxfrm
wctob	wctomb
wctrans	wctype
wcwidth	wmemchr
wmemcmp	wmemcpy
wmemmove	wmemset
wordexp	wordfree
wprintf	wracct
write	writev
wscanf	wscasecmp
wscat	wschr
wscmp	wscol
wscoll	wscpy

wscspn	wsdup
wslen	wsncasecmp
wsncat	wsncmp
wsncpy	wspbrk
wsprintf	wsrchr
wsscanf	wsspn
wstod	wstok
wstol	wstoll
wstostr	wsxfrm
yield	

The following interfaces are unique to the 32-bit version of this library:

__div64	__mul64
__posix_readdir_r	__rem64
__udiv64	__urem64
_bufendtab	_creat64
_fstat64	_fstatvfs64
_ftruncate64	_ftw64
_getdents64	_getrlimit64
_lastbuf	_lockf64
_lseek64	_lstat64
_mkstemp64	_mmap64
_nftw64	_open64
_pread64	_pwrite64
_readdir64	_readdir64_r
_s_fcntl	_setrlimit64
_stat64	_statvfs64
_sys_nsig	_tell64
_truncate64	_xftw64
creat64	fgetpos64

fopen64	freopen64
fseeko64	fsetpos64
fstat64	fstatvfs64
ftello64	ftruncate64
ftw64	getdents64
getrlimit64	lockf64
lseek64	lstat64
mkstemp64	mmap64
nftw64	open64
pread64	pwrite64
readdir64	readdir64_r
s_fcntl	s_ioctl
select_large_fdset	setrlimit64
stat64	statvfs64
sys_errlist	sys_nerr
tell64	tmpfile64
truncate64	

The following interfaces are unique to the 32-bit SPARC version of this library:

.div	.mul
.rem	.stret1
.stret2	.stret4
.stret8	.udiv
.umul	.urem
_Q_add	_Q_cmp
_Q_cmpe	_Q_div
_Q_dtoq	_Q_feq
_Q_fge	_Q_fgt
_Q_fle	_Q_flt
_Q_fne	_Q_itoq

_Q_lltoq	_Q_mul
_Q_neg	_Q_qtod
_Q_qtoi	_Q_qtoll
_Q_qtos	_Q_qtou
_Q_qtoull	_Q_sqrt
_Q_stoq	_Q_sub
_Q_ulltoq	_Q_utoq
__dtoll	__dtou
__dtoull	__ftoll
__ftou	__ftoull
__umul64	

The following interfaces are unique to the 32-bit Intel version of this library:

__fpstart	_fp_hw
_fpstart	_fxstat
_lxstat	_nuname
_thr_errno_addr	_xmknod
_xstat	nuname

The following interfaces are unique to the 64-bit SPARC version of this library:

_Qp_add	_Qp_cmp
_Qp_cmpe	_Qp_div
_Qp_dtoq	_Qp_feq
_Qp_fge	_Qp_fgt
_Qp_fle	_Qp_flt
_Qp_fne	_Qp_itoq
_Qp_mul	_Qp_neg
_Qp_qtod	_Qp_qtoi
_Qp_qtos	_Qp_qtoui

_Qp_qtoux	_Qp_qtox
_Qp_sqrt	_Qp_stoq
_Qp_sub	_Qp_uitoq
_Qp_uxtoq	_Qp_xtoq
__align_cpy_1	__align_cpy_16
__align_cpy_2	__align_cpy_4
__align_cpy_8	__dtoul
__ftoul	__sparc_utrap_install

FILES

/usr/lib/libc.a	archive library
/usr/lib/libc.so.1	shared object
/usr/lib/64/libc.so.1	64–bit shared object

ATTRIBUTES

See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsl, SUNWarc (32-bit) SUNWcslx (64-bit)
MT-Level	Safe

SEE ALSO

pvs(1), intro(2), intro(3), attributes(5), lf64(5), standards(5)

NAME

libcfgadm – configuration adminstartion library

SYNOPSIS

```
cc [ flag . . . ] file . . . -lcfgadm -ldevinfo -ldl [ library . . . ]
#include <config_admin.h>
```

DESCRIPTION

Functions in this library provide services for configuration administration.

INTERFACES

The shared object `libcfgadm.so.1` provides the public interfaces defined below. See `intro`(3) for additional information on shared object interfaces.

`config_ap_id_cmp`	`config_change_state`
`config_help`	`config_list`
`config_list_ext`	`config_private_func`
`config_stat`	`config_strerror`
`config_test`	`config_unload_libs`

FILES

`/usr/lib/libcfgadm.so.1`
shared object

`/usr/lib/64/libcfgadm.so.1`
64–bit shared object

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsl (32-bit) SUNWcslx (64-bit)
MT Level	Mt-Safe

SEE ALSO

`pvs`(1), `cfgadm`(1M), `intro`(3), `config_admin`(3CFGADM), `attributes`(5)

NAME

libcpc – CPU performance counter library

SYNOPSIS

```
cc [ flag . . . ] file . . . -lcpc [ library . . . ]
```

DESCRIPTION

Functions in this library provide access to CPU performance counters on platforms that contain the appropriate hardware.

INTERFACES

The shared object `libcpc.so.1` provides the public interfaces defined below. See `intro`(3) for additional information on shared object interfaces.

`cpc_access`	`cpc_bind_event`
`cpc_count_sys_events`	`cpc_count_usr_events`
`cpc_event_accum`	`cpc_event_diff`
`cpc_eventtostr`	`cpc_getcciname`
`cpc_getcpuref`	`cpc_getcpuver`
`cpc_getnpic`	`cpc_getusage`
`cpc_pctx_bind_event`	`cpc_pctx_invalidate`
`cpc_pctx_rele`	`cpc_pctx_take_sample`
`cpc_rele`	`cpc_seterrfn`
`cpc_shared_bind_event`	`cpc_shared_close`
`cpc_shared_open`	`cpc_shared_rele`
`cpc_shared_take_sample`	`cpc_strtoevent`
`cpc_take_sample`	`cpc_version`
`cpc_walk_names`	

FILES

`/usr/lib/libcpc.so.1`	shared object
`/usr/lib/64/libcpc.so.1`	64-bit shared object

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcpcu (32-bit) SUNWcpcux (64-bit)
MT-Level	Safe

SEE ALSO

`cputrack`(1), `cpustat`(1M), `intro`(3), `cpc`(3CPC), `attributes`(5)

NAME

libcrypt – encryption/decryption library

SYNOPSIS

```
cc [ flag . . . ] file . . . -lcrypt [ library . . . ]
```

DESCRIPTION

Functions in this library provide encoding and decoding handling routines.

INTERFACES

The shared object `libcrypt.so.1` provides the public interfaces defined below. See `intro`(3) for additional information on shared object interfaces.

```
crypt                   encrypt                 setkey
```

FILES

`/usr/lib/libcrypt.a`
archive library

`/usr/lib/libcrypt.so.1`
shared object

`/usr/lib/64/libcrypt.so.1`
64-bit shared object

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
MT-Level	Unsafe

SEE ALSO

`crypt`(1), `intro`(3), `encrypt`(3C), `setkey`(3C), `attributes`(5)

NAME

libcurses, libtermcap, libtermlib – screen handling and optimization library

SYNOPSIS

```
cc [ flag . . . ] file . . . -lcurses [ library . . . ]
```

DESCRIPTION

Functions in the `libcurses` library provide a terminal-independent method of updating character screens with reasonable optimization. The `libtermcap` and `libtermlib` libraries are identical to `libcurses` and are maintained for backward compatibility.

See `libcurses`(3XCURSES) for information about the curses library that conforms to X/Open Curses, Issue 4, Version 2.

INTERFACES

The shared objects `libcurses.so.1`, `libtermcap.so.1`, and `libtermlib.so.1` provide the public interfaces defined below. See `intro`(3) for additional information on shared object interfaces.

`_getsyx`	`_meta`
`_ring`	`_setecho`
`_setnonl`	`_setqiflush`
`addch`	`addchnstr`
`addchstr`	`addnstr`
`addnwstr`	`addstr`
`addwch`	`addwchnstr`
`addwchstr`	`addwstr`
`attroff`	`attron`
`attrset`	`baudrate`
`beep`	`bkgd`
`bkgdset`	`border`
`box`	`can_change_color`
`cbreak`	`clear`
`clearok`	`clrtobot`
`clrtoeol`	`color_content`
`copywin`	`crmode`
`curs_set`	`curserr`
`def_prog_mode`	`def_shell_mode`
`del_curterm`	`delay_output`

delch	deleteln
delkeymap	delscreen
delwin	derwin
doupdate	dupwin
echo	echochar
echowchar	endwin
erase	erasechar
filter	flash
flushinp	getbmap
getch	getmouse
getnwstr	getstr
getwch	getwin
getwstr	halfdelay
has_colors	has_ic
has_il	idcok
idlok	immedok
inch	inchnstr
inchstr	init_color
init_pair	initscr
innstr	innwstr
insch	insdelln
insertln	insnstr
insnwstr	insstr
instr	inswch
inswstr	intrflush
inwch	inwchnstr
inwchstr	inwstr
is_linetouched	is_wintouched
isendwin	keyname
keypad	killchar

leaveok	longname
m_addch	m_addstr
m_clear	m_erase
m_initscr	m_move
m_newterm	m_refresh
map_button	meta
mouse_off	mouse_on
mouse_set	move
mvaddch	mvaddchnstr
mvaddchstr	mvaddnstr
mvaddnwstr	mvaddstr
mvaddwch	mvaddwchnstr
mvaddwchstr	mvaddwstr
mvcur	mvdelch
mvderwin	mvgetch
mvgetnwstr	mvgetstr
mvgetwch	mvgetwstr
mvinch	mvinchnstr
mvinchstr	mvinnstr
mvinnwstr	mvinsch
mvinsnstr	mvinsnwstr
mvinsstr	mvinstr
mvinswch	mvinswstr
mvinwch	mvinwchnstr
mvinwchstr	mvinwstr
mvprintw	mvscanw
mvwaddch	mvwaddchnstr
mvwaddchstr	mvwaddnstr
mvwaddnwstr	mvwaddstr
mvwaddwch	mvwaddwchnstr

mvwaddwchstr	mvwaddwstr
mvwdelch	mvwgetch
mvwgetnwstr	mvwgetstr
mvwgetwch	mvwgetwstr
mvwin	mvwinch
mvwinchnstr	mvwinchstr
mvwinnstr	mvwinnwstr
mvwinsch	mvwinsnstr
mvwinsnwstr	mvwinsstr
mvwinstr	mvwinswch
mvwinswstr	mvwinwch
mvwinwchnstr	mvwinwchstr
mvwinwstr	mvwprintw
mvwscanw	napms
newkey	newpad
newscreen	newterm
newwin	nl
nocbreak	nocrmode
nodelay	noecho
nonl	noqiflush
noraw	notimeout
overlay	overwrite
pair_content	pechochar
pechowchar	pnoutrefresh
prefresh	printw
putp	putwin
qiflush	raw
redrawwin	refresh
request_mouse_pos	reset_prog_mode
reset_shell_mode	resetty

restartterm	ripoffline
savetty	scanw
scr_dump	scr_init
scr_restore	scr_set
scrl	scroll
scrollok	set_term
setcurscreen	setscrreg
setsyx	setterm
setupterm	slk_attroff
slk_attron	slk_attrset
slk_clear	slk_init
slk_label	slk_noutrefresh
slk_refresh	slk_restore
slk_set	slk_start
slk_touch	standend
standout	start_color
subpad	subwin
syncok	termattrs
termname	tgetent
tgetflag	tgetnum
tgetstr	tgoto
tigetflag	tigetnum
tigetstr	timeout
touchline	touchwin
tparm	tputs
traceoff	traceon
typeahead	unctrl
ungetch	ungetwch
untouchwin	vidattr
vidputs	vidupdate

vwprintw	vwscanw
waddch	waddchnstr
waddchstr	waddnstr
waddnwstr	waddstr
waddwch	waddwchnstr
waddwchstr	waddwstr
wadjcurspos	wattroff
wattron	wattrset
wbkgd	wbkgdset
wborder	wclear
wclrtobot	wclrtoeol
wcursyncup	wdelch
wdeleteln	wechochar
wechowchar	werase
wgetch	wgetnstr
wgetnwstr	wgetstr
wgetwch	wgetwstr
whline	winch
winchnstr	winchstr
winnstr	winnwstr
winsch	winsdelln
winsertln	winsnstr
winsnwstr	winsstr
winstr	winswch
winswstr	winwch
winwchnstr	winwchstr
winwstr	wmouse_position
wmove	wmovenextch
wmoveprevch	wnoutrefresh
wprintw	wredrawln

wrefresh	wscanw
wscrl	wsetscrreg
wstandend	wstandout
wsyncdown	wsyncup
wtimeout	wtouchln
wvline	

FILES

/usr/lib/libcurses.a
archive library

/usr/lib/libcurses.so.1
shared object

/usr/lib/64/libcurses.so.1
64-bit shared object

/usr/lib/libtermcap.a
archive library (hard link to /usr/lib/libcurses.a)

/usr/lib/libtermcap.so.1
shared object (symbolic link to /usr/lib/libcurses.so.1)

/usr/lib/64/libtermcap.so.1
64-bit shared object (symbolic link to /usr/lib/64/libcurses.so.1)

/usr/lib/libtermlib.a
archive library (hard link to /usr/lib/libcurses.a)

/usr/lib/libtermlib.so.1
shared object (symbolic link to /usr/lib/libcurses.so.1)

/usr/lib/64/libtermlib.so.1
64-bit shared object (symbolic link to /usr/lib/64/libcurses.so.1)

ATTRIBUTES

See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsl, SUNWarc (32-bit) SUNWcslx (64-bit)
MT-Level	Unsafe

SEE ALSO

intro(3), curses(3CURSES), libcurses(3LIBUCB), libcurses(3XCURSES), attributes(5)

NAME

libcurses – SunOS/BSD-compatible screen handling and optimization library

SYNOPSIS

```
cc [ flag . . . ] -I /usr/ucbinclude file . . . -L /usr/libucb \
-R /usr/libucb -lcurses [ library . . . ]
```

DESCRIPTION

Functions in this library provide a terminal-independent method of updating character screens with reasonable optimization, compatible with SunOS/BSD.

INTERFACES

The shared object `libcurses.so.1` provides the public interfaces defined below. See `intro`(3) for additional information on shared object interfaces.

AL	AL_PARM	AM	BC
BS	BT	CA	CD
CE	CL	CM	COLS
CR	CS	DA	DB
DC	DL	DL_PARM	DM
DO	DOWN_PARM	Def_term	ED
EI	EO	GT	HC
HO	HZ	IC	IM
IN	IP	K0	K1
K2	K3	K4	K5
K6	K7	K8	K9
KD	KE	KH	KL
KR	KS	KU	LEFT_PARM
LINES	LL	MA	MI
MS	My_term	NC	ND
NL	NONL	NS	OS
PC	RC	RIGHT_PARM	SC
SE	SF	SO	SR
TA	TE	TI	UC
UE	UL	UP	UPPERCASE
UP_PARM	US	VB	VE
VS	XB	XN	XS
XT	XX	_echoit	_endwin

_pfast	_rawmode	_res_flg	_tty
_tty_ch	_unctrl	box	curscr
delwin	endwin	getcap	gettmode
idlok	initscr	longname	mvcur
mvprintw	mvscanw	mvwin	mvwprintw
mvwscanw	newwin	normtty	overlay
overwrite	printw	scanw	scroll
setterm	stdscr	subwin	touchline
touchwin	ttytype	waddch	waddstr
wclear	wclrtobot	wclrtoeol	wdelch
wdeleteln	werase	wgetch	wgetstr
winsch	winsertln	wmove	wprintw
wrefresh	wscanw	wstandend	wstandout

FILES

`/usr/libucb/libcurses.a`
archive library

`/usr/libucb/libcurses.so.1`
shared object

`/usr/libucb/64/libcurses.so.1`
64-bit shared object

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
MT-Level	Unsafe

SEE ALSO

`intro`(3), `libcurses`(3LIB), `libcurses`(3XCURSES), `attributes`(5)

NAME

libdbm – database subroutines library

SYNOPSIS

```
cc [ flag . . . ] -I /usr/ucbinclude file . . . -L /usr/libucb \
-R /usr/libucb -ldbm [ library . . . ]
```

DESCRIPTION

Functions in this library maintain key/content pairs in a database. The functions will handle very large (a billion blocks) databases and will access a keyed item in one or two file system accesses.

INTERFACES

The shared object `libdbm.so.1` provides the public interfaces defined below. See `intro`(3) for additional information on shared object interfaces.

`bitno`	`blkno`	`calchash`	`dbmclose`	`dbminit`
`dbrdonly`	`delete`	`dirbuf`	`dirf`	`fetch`
`firstkey`	`hashinc`	`hmask`	`makdatum`	`maxbno`
`nextkey`	`pagbuf`	`pagf`	`store`	

FILES

`/usr/libucb/libdbm.a`
archive library

`/usr/libucb/libdbm.so.1`
shared object

`/usr/libucb/64/libdbm.so.1`
64-bit shared object

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
MT-Level	Unsafe

SEE ALSO

`intro`(3), `dbm`(3UCB), `attributes`(5)

NAME

libdevid – device ID library

SYNOPSIS

```
cc [ flag . . . ] file . . . -ldevid [ library . . . ]
#include <devid.h>
```

DESCRIPTION

Functions in this library provide unique device IDs for identifying a device, independent of the device name or device number.

INTERFACES

The shared object `libdevid.so.1` provides the public interfaces defined below. See `intro`(3) for additional information on shared object interfaces.

`devid_compare`	`devid_deviceid_to_nmlist`
`devid_free`	`devid_free_nmlist`
`devid_get`	`devid_get_minor_name`
`devid_sizeof`	`devid_str_decode`
`devid_str_encode`	`devid_str_free`
`devid_valid`	

FILES

`/usr/lib/libdevid.so.1`	shared object.
`/usr/lib/64/libdevid.so.1`	64-bit shared object.

ATTRIBUTES

See `attributes`(5) for description of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsl (32-bit) SUNWcslx (64-bit)
MT-Level	MT–Safe
Interface Stability	Stable

SEE ALSO

`pvs`(1), `intro`(3), `attributes`(5)

NAME

libdevinfo – device information library

SYNOPSIS

```
cc [ flag ... ] file ... -ldevinfo [ library ... ]
```

DESCRIPTION

Functions in this library access information on device configuration.

INTERFACES

The shared object `libdevinfo.so.1` provides the public interfaces defined below. See `intro`(3) for additional information on shared object interfaces.

`di_binding_name`	`di_bus_addr`
`di_child_node`	`di_compatible_names`
`di_devfs_path`	`di_devfs_path_free`
`di_devid`	`di_driver_name`
`di_driver_ops`	`di_drv_first_node`
`di_drv_next_node`	`di_fini`
`di_init`	`di_instance`
`di_minor_class`	`di_minor_devt`
`di_minor_name`	`di_minor_next`
`di_minor_nodetype`	`di_minor_spectype`
`di_minor_type`	`di_node_name`
`di_nodeid`	`di_parent_node`
`di_prom_fini`	`di_prom_init`
`di_prom_prop_data`	`di_prom_prop_lookup_bytes`
`di_prom_prop_lookup_ints`	`di_prom_prop_lookup_strings`
`di_prom_prop_name`	`di_prom_prop_next`
`di_prop_bytes`	`di_prop_devt`
`di_prop_ints`	`di_prop_lookup_bytes`
`di_prop_lookup_ints`	`di_prop_lookup_strings`
`di_prop_name`	`di_prop_next`
`di_prop_strings`	`di_prop_type`
`di_sibling_node`	`di_state`
`di_walk_minor`	`di_walk_node`

FILES

`usr/lib/libdevinfo.a`
archive library

`/usr/lib/libdevinfo.so.1`
shared object

`/usr/lib/64/libdevinfo.so.1`
64-bit shared object

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsl, SUNWstatl (32-bit) SUNWcslx (64-bit)
MT Level	Safe
Interface Stability	Evolving

SEE ALSO

`pvs`(1), `intro`(3), `libdevinfo`(3DEVINFO), `attributes`(5)

Writing Device Drivers

NAME

libdl – dynamic linking library

SYNOPSIS

```
cc [ flag . . . ] file . . . -ldl [ library  . . . ]
```

DESCRIPTION

Functions in this library provide direct access to the dynamic linking facilities. This library is implemented as a filter on the runtime linker (see `ld.so.1`(1)).

INTERFACES

The shared object `libdl.so.1` provides the public interfaces defined below. See `intro`(3) for additional information on shared object interfaces.

`dladdr`	`dladdr1`	`dlclose`	`dldump`
`dlerror`	`dlinfo`	`dlmopen`	`dlopen`
`dlsym`			

FILES

`/usr/lib/libdl.so.1`	shared object
`/etc/lib/libdl.so.1`	shared object (copy)
`/usr/lib/64/libdl.so.1`	64-bit shared object

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsl (32-bit) SUNWcslx (64-bit)
MT Level	Safe

SEE ALSO

`ld.so.1`(1), `pvs`(1), `intro`(3), `attributes`(5)

NAME

libdmi – Sun Solstice Enterprise Agent DMI library

SYNOPSIS

```
cc [ flag . . . ] file . . .  -ldmi  -lnsl  -lrwtool  [ library  . .  ]
```

DESCRIPTION

The `libdmi` library is a Solstice Enterprise Agent DMI generic library. It supports the DMI service provider, management application, and component instrumentation with data encoding, RPC communication, and other functionalities. This library is linked with management application and component instrumentation programs.

INTERFACES

The shared object `libdmi.so.1` provides the public interfaces defined below. See `intro`(3) for additional information on shared object interfaces.

`dmi_error`	`freeDmiString`
`newDmiAttributeValues`	`newDmiOctetStringFromString`
`newDmiString`	`printDmiDataUnion`
`printDmiString`	

FILES

`/usr/lib/libdmi.so.1`	shared object
`/usr/lib/64/libdmi.so.1`	64-bit shared object

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWsadmi (32-bit) SUNWsadmx (64-bit)
MT-Level	Unsafe

SEE ALSO

`intro`(3), `libdmici`(3LIB), `libdmimi`(3LIB), `attributes`(5)

NAME

libdmici – Sun Solstice Enterprise Agent Component library

SYNOPSIS

```
cc [ flag . . . ] file . . . -ldmici -ldmi -lnsl -lrwtool \
[ library . . ]
```

DESCRIPTION

The `libdmici` library provides Component Interface API functions.

INTERFACES

The shared object `libdmici.so.1` provides the public interfaces defined below. See `intro`(3) for additional information on shared object interfaces.

`ConnectToServer`	`DisconnectToServer`
`DmiOriginateEvent`	`DmiRegisterCi`
`DmiUnregisterCi`	`reg_ci_callback`
`start_svc_run_thread`	

FILES

`/usr/lib/libdmici.so.1`	shared object
`/usr/lib/sparcv9/libdmici.so.1`	64-bit shared object

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWsadmi (32-bit) SUNWsadmx (64-bit)
MT-Level	Unsafe

SEE ALSO

`intro`(3), `libdmi`(3LIB), `attributes`(5)

NAME

libdmimi – Sun Solstice Enterprise Agent Management library

SYNOPSIS

```
cc [ flag . . . ] file . . . -ldmimi  -ldmi   -lnsl   -lrwtool  \
[ library  . .  ]
```

DESCRIPTION

The `libdmimi` library provides Management Interface API functions.

INTERFACES

The shared object `libdmimi.so.1` provides the public interfaces defined below. See `intro`(3) for additional information on shared object interfaces.

`ConnectToServer`	`DisconnectToServer`
`DmiAddComponent`	`DmiAddGroup`
`DmiAddLanguage`	`DmiAddRow`
`DmiDeleteComponent`	`DmiDeleteGroup`
`DmiDeleteLanguage`	`DmiDeleteRow`
`DmiGetAttribute`	`DmiGetConfig`
`DmiGetMultiple`	`DmiGetVersion`
`DmiListAttributes`	`DmiListClassNames`
`DmiListComponents`	`DmiListComponentsByClass`
`DmiListGroups`	`DmiListLanguages`
`DmiRegister`	`DmiSetAttribute`
`DmiSetConfig`	`DmiSetMultiple`
`DmiUnregister`	

FILES

`/usr/lib/libdmimi.so.1`	shared object
`/usr/lib/64/libdmimi.so.1`	64–bit shared object

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWsadmi (32-bit) SUNWsadmx (64-bit)
MT-Level	Unsafe

SEE ALSO

`intro`(3), `libdmi`(3LIB), `attributes`(5)

NAME

libelf – ELF access library

SYNOPSIS

```
cc [ flag . . . ] file . . . -lelf [ library  . . . ]
#include <libelf.h>
```

DESCRIPTION

Functions in this library provide routines to manipulate ELF (Executable and Linking Format) object files, archive files, and archive members. The header provides type and function declarations for all library services.

INTERFACES

The shared object `libelf.so.1` provides the public interfaces defined below. See `intro`(3) for additional information on shared object interfaces.

`elf32_checksum`	`elf32_fsize`
`elf32_getehdr`	`elf32_getphdr`
`elf32_getshdr`	`elf32_newehdr`
`elf32_newphdr`	`elf32_xlatetof`
`elf32_xlatetom`	`elf64_checksum`
`elf64_fsize`	`elf64_getehdr`
`elf64_getphdr`	`elf64_getshdr`
`elf64_newehdr`	`elf64_newphdr`
`elf64_xlatetof`	`elf64_xlatetom`
`elf_begin`	`elf_cntl`
`elf_end`	`elf_errmsg`
`elf_errno`	`elf_fill`
`elf_flagdata`	`elf_flagehdr`
`elf_flagelf`	`elf_flagphdr`
`elf_flagscn`	`elf_flagshdr`
`elf_getarhdr`	`elf_getarsym`
`elf_getbase`	`elf_getdata`
`elf_getident`	`elf_getscn`
`elf_hash`	`elf_kind`
`elf_memory`	`elf_ndxscn`
`elf_newdata`	`elf_newscn`
`elf_next`	`elf_nextscn`
`elf_rand`	`elf_rawdata`

elf_rawfile	elf_strptr
elf_update	elf_version
gelf_checksum	gelf_fsize
gelf_getclass	gelf_getdyn
gelf_getehdr	gelf_getmove
gelf_getphdr	gelf_getrel
gelf_getrela	gelf_getshdr
gelf_getsym	gelf_getsyminfo
gelf_newehdr	gelf_newphdr
gelf_update_dyn	gelf_update_ehdr
gelf_update_move	gelf_update_phdr
gelf_update_rel	gelf_update_rela
gelf_update_shdr	gelf_update_sym
gelf_update_syminfo	gelf_xlatetof
gelf_xlatetom	nlist

FILES

/usr/lib/libelf.a	archive library
/usr/lib/libelf.so.1	shared object
/usr/lib/64/libelf.so.1	64-bit shared object

ATTRIBUTES

See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsl, SUNWarc (32-bit) SUNWcslx (64-bit)
MT-Level	Safe

SEE ALSO

pvs(1), intro(3), elf(3ELF), gelf(3ELF), attributes(5)

NAME

libexacct – extended accounting file access library

SYNOPSIS

```
cc [ flag... ] file... -lexacct [ library... ]
#include <exacct.h>
```

DESCRIPTION

Functions in this library define the interface for reading and writing extended accounting (`exacct`) files. The `<exacct.h>` header provides type and function declarations for all library services, as well as for the characteristics of accounting files generated by the Solaris kernel.

INTERFACES

The shared object `libexacct.so.1` provides the public interfaces defined below. See `intro`(3) for additional information on shared object interfaces.

`ea_alloc`	`ea_attach_to_group`
`ea_attach_to_object`	`ea_close`
`ea_copy_object`	`ea_copy_object_tree`
`ea_error`	`ea_free`
`ea_free_item`	`ea_free_object`
`ea_get_creator`	`ea_get_hostname`
`ea_get_object`	`ea_get_object_tree`
`ea_match_object_catalog`	`ea_next_object`
`ea_open`	`ea_pack_object`
`ea_previous_object`	`ea_set_group`
`ea_set_item`	`ea_strdup`
`ea_strfree`	`ea_unpack_object`
`ea_write_object`	

FILES

`/usr/lib/libexacct.so.1`
shared object

`/usr/lib/64/libexacct.so.1`
64-bit shared object

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsl (32-bit) SUNWcslx (64-bit)

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `acctadm`(1M), `intro`(3), `ea_error`(3EXACCT), `ea_open`(3EXACCT), `ea_pack_object`(3EXACCT), `ea_set_item`(3EXACCT), `attributes`(5)

NOTES The SUNWosdem package provides source code for the `exdump` utility that uses the `libexacct` APIs to dump the contents of extended accounting files. The source code can be compiled in the directory `/usr/demo/libexacct`.

NAME

libform – forms library

SYNOPSIS

```
cc [ flag . . . ] file . . . -lform [ library . . . ]
```

DESCRIPTION

Functions in this library provide forms using `libcurses`(3LIB) routines.

INTERFACES

The shared object `libform.so.1` provides the public interfaces defined below. See `intro`(3) for additional information on shared object interfaces.

`current_field`	`data_ahead`
`data_behind`	`dup_field`
`dynamic_field_info`	`field_arg`
`field_back`	`field_buffer`
`field_count`	`field_fore`
`field_index`	`field_info`
`field_init`	`field_just`
`field_opts`	`field_opts_off`
`field_opts_on`	`field_pad`
`field_status`	`field_term`
`field_type`	`field_userptr`
`form_driver`	`form_fields`
`form_init`	`form_opts`
`form_opts_off`	`form_opts_on`
`form_page`	`form_sub`
`form_term`	`form_userptr`
`form_win`	`free_field`
`free_fieldtype`	`free_form`
`link_field`	`link_fieldtype`
`move_field`	`new_field`
`new_fieldtype`	`new_form`
`new_page`	`pos_form_cursor`
`post_form`	`scale_form`
`set_current_field`	`set_field_back`

set_field_buffer	set_field_fore
set_field_init	set_field_just
set_field_opts	set_field_pad
set_field_status	set_field_term
set_field_type	set_field_userptr
set_fieldtype_arg	set_fieldtype_choice
set_form_fields	set_form_init
set_form_opts	set_form_page
set_form_sub	set_form_term
set_form_userptr	set_form_win
set_max_field	set_new_page
unpost_form	

FILES

/usr/lib/libform.a	archive library
/usr/lib/libform.so.1	shared object
/usr/lib/64/libform.so.1	64-bit shared object

ATTRIBUTES

See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsl, SUNWarc (32-bit) SUNWcslx (64-bit)
MT-Level	Unsafe

SEE ALSO

intro(3), libcurses(3LIB), attributes(5)

NAME

libgen – string pattern-matching library

SYNOPSIS

```
cc [ flag . . . ] file . . . -lgen [ library . . . ]
```

DESCRIPTION

Functions in this library provide routines for string pattern-matching and pathname manipulation.

INTERFACES

The shared object `libgen.so.1` provides the public interfaces defined below. See `intro`(3) for additional information on shared object interfaces.

`___braelist`	`___braslist`	`___loc1`
`___loc2`	`___locs`	`___nbra`
`___regerrno`	`___reglength`	`advance`
`bgets`	`braelist`	`braslist`
`bufsplit`	`compile`	`copylist`
`eaccess`	`gmatch`	`isencrypt`
`loc1`	`loc2`	`locs`
`mkdirp`	`nbra`	`p2close`
`p2open`	`pathfind`	`regerrno`
`reglength`	`rmdirp`	`step`
`strcadd`	`strccpy`	`streadd`
`strecpy`	`strfind`	`strrspn`
`strtrns`		

The following interface is unique to the 32-bit version of this library:

`copylist64`

FILES

`/usr/lib/libgen.a`	archive library
`/usr/lib/libgen.so.1`	shared object
`/usr/lib/64/libgen.so.1`	64-bit shared object

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsl, SUNWarc (32-bit)

ATTRIBUTE TYPE	ATTRIBUTE VALUE
	SUNWcslx (64-bit)
MT-Level	Safe

SEE ALSO `intro`(3), `attributes`(5)

NAME

libgss – Generic Security Services library

SYNOPSIS

```
cc [ flag . . . ] file . . . -lgss [ library . . . ]
#include <gssapi/gssapi.h>
```

DESCRIPTION

The functions in this library are the routines that comprise the Generic Security Services library.

INTERFACES

The shared object `libgss.so.1` provides the public interfaces defined below. See `intro`(3) for additional information on shared object interfaces.

GSS_C_NT_ANONYMOUS	GSS_C_NT_EXPORT_NAME
GSS_C_NT_HOSTBASED_SERVICE	GSS_C_NT_MACHINE_UID_NAME
GSS_C_NT_STRING_UID_NAME	GSS_C_NT_USER_NAME
gss_accept_sec_context	gss_acquire_cred
gss_add_cred	gss_add_oid_set_member
gss_canonicalize_name	gss_compare_name
gss_context_time	gss_create_empty_oid_set
gss_delete_sec_context	gss_display_name
gss_display_status	gss_duplicate_name
gss_export_name	gss_export_sec_context
gss_get_mic	gss_import_name
gss_import_sec_context	gss_indicate_mechs
gss_init_sec_context	gss_inquire_context
gss_inquire_cred	gss_inquire_cred_by_mech
gss_inquire_mechs_for_name	gss_inquire_names_for_mech
gss_process_context_token	gss_release_buffer
gss_release_cred	gss_release_name
gss_release_oid	gss_release_oid_set
gss_seal	gss_sign
gss_test_oid_set_member	gss_unseal
gss_unwrap	gss_verify
gss_verify_mic	gss_wrap
gss_wrap_size_limit	

FILES

`/usr/lib/libgss.so.1`
shared object

`/usr/lib/64/libgss.so.1`
64-bit shared object file

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWgss (32-bit) SUNWgssx (64-bit)
MT-Level	Safe

SEE ALSO

`pvs`(1), `intro`(2), `intro`(3), `attributes`(5)

GSS-API Programming Guide

NAME

libintl – internationalization library

SYNOPSIS

```
cc [ flag . . . ] file . . -lintl [ library . . . ]
#include <libintl.h>
#include <locale.h> /* needed for dcgettext() only */
```

DESCRIPTION

Historically, functions in this library provided wide character translations. This functionality now resides in `libc`(3LIB).

This library is maintained to provide backward compatibility for both runtime and compilation environments. The shared object version is implemented as a filter on `libintl.so.1`, and the archive version is implemented as a null archive. New application development need not reference either version of `libintl`.

INTERFACES

The shared object `libintl.so.1` provides the public interfaces defined below. See `intro`(3) for additional information on shared object interfaces.

`bindtextdomain`	`dcgettext`
`dgettext`	`gettext`
`textdomain`	

FILES

`/usr/lib/libintl.a`	a link to `/usr/lib/null.a`
`/usr/lib/libintl.so.1`	a filter on `libc.so.1`
`/usr/lib/64/libintl.so.1`	a filter on `64/libc.so.1`

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsl, SUNWarc (32-bit) SUNWcslx (64-bit)
MT-Level	Safe with exceptions

SEE ALSO

`pvs`(1), `intro`(3), `gettext`(3C), `libc`(3LIB), `attributes`(5)

NAME

libkstat – kernel statistics library

SYNOPSIS

```
cc [ flag . . . ] file . . . -lkstat [ library . . . ]
#include <kstat.h>
```

DESCRIPTION

Functions in this library provide a general-purpose mechanism for providing kernel statistics to users.

INTERFACES

The shared object `libkstat.so.1` provides the public interfaces defined below. See `intro`(3) for additional information on shared object interfaces.

`kstat_chain_update`	`kstat_close`
`kstat_data_lookup`	`kstat_lookup`
`kstat_open`	`kstat_read`
`kstat_write`	

FILES

`/usr/lib/libkstat.so.1`	shared object
`/usr/lib/64/libkstat.so.1`	64-bit shared object

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsl (32-bit) SUNWcslx (64-bit)
MT-Level	Unsafe

SEE ALSO

`pvs`(1), `intro`(3), `kstat`(3KSTAT), `attributes`(5)

NAME

libkvm – Kernel Virtual Memory access library

SYNOPSIS

```
cc [ flag . . . ] file . . . -lkvm [ library . . . ]
#include <kvm.h>
```

DESCRIPTION

Functions in this library provide application access to kernel symbols, addresses and values. The individual functions are documented in Section 3KVM of the reference manual.

All `libkvm` functions are Uncommitted, since there is almost nothing that can be put as a symbol in a namelist that has stability from release to release. The syntax of these functions has been stable from release to release, but this is subject to change.

INTERFACES

The shared object `libkvm.so.1` provides the public interfaces defined below. See `intro`(3) for additional information on shared object interfaces.

`kvm_close`	`kvm_getcmd`	`kvm_getproc`
`kvm_getu`	`kvm_kread`	`kvm_kwrite`
`kvm_nextproc`	`kvm_nlist`	`kvm_open`
`kvm_read`	`kvm_setproc`	`kvm_uread`
`kvm_uwrite`	`kvm_write`	

FILES

`/usr/lib/libkvm.so.1`	shared object
`/usr/lib/64/libkvm.so.1`	64-bit shared object

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsl (32-bit) SUNWcsl x(64-bit)
MT-Level	Unsafe

SEE ALSO

`pvs`(1), `intro`(3), `attributes`(5)

NAME

libl – lex library

SYNOPSIS

```
cc [ flag . . . ] file . . -ll [ library . . . ]
```

DESCRIPTION

Functions in this library provide user interfaces to the `lex`(1) library.

INTERFACES

The shared object `libl.so.1` provides the public interfaces defined below. See `intro`(3) for additional information on shared object interfaces.

`allprint`	`allprint_w`	`main`	`sprint`
`sprint_w`	`yyless`	`yyless_e`	`yyless_w`
`yyracc`	`yyreject`	`yyreject_e`	`yyreject_w`
`yywrap`			

FILES

`/usr/lib/libl.a`	archive library
`/usr/lib/libl.so.1`	shared object
`/usr/lib/64/libl.so.1`	64-bit shared object

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsl (32-bit) SUNWcslx (64-bit)
MT-Level	Unsafe

SEE ALSO

`lex`(1), `intro`(3), `attributes`(5)

NAME

liblayout – layout service library

SYNOPSIS

```
cc [ flag . . . ] file . . . -llayout [ library  . . . ]
#include <sys/layout.h>
```

DESCRIPTION

Functions in this library provide various layout service routines.

INTERFACES

The shared object `liblayout.so.1` provides the public inter faces defined below. See `intro`(3) for additional information on shared object interfaces.

`m_create_layout`	`m_destroy_layout`
`m_getvalues_layout`	`m_setvalues_layout`
`m_transform_layout`	`m_wtransform_layout`

FILES

`/usr/lib/liblayout.so.1`	shared object
`/usr/lib/64/liblayout.so.1`	64-bit shared object.

ATTRIBUTES

See `attributes`(5) for description of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWctpls (32-bit) SUNWctplx (64-bit)
MT Level	MT–Safe

SEE ALSO

`intro`(3), `attributes`(5)

NAME

libmail – user mailbox lockfile management library

SYNOPSIS

```
cc [ flag . . . ] file . . . -lmail  [ library . . . ]
#include <maillock.h>
```

DESCRIPTION

Interfaces in this library provide functions for managing user mailbox lockfiles.

INTERFACES

The shared object `libmail.so.1` provides the public interfaces defined below. See `intro`(3) for additional information on shared object interfaces.

```
maillock          mailunlock        touchlock
```

FILES

`/usr/lib/libmail.a`
archive library

`/usr/lib/libmail.so.1`
shared object

`/usr/lib/64/libmail.so.1`
64–bit shared object

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsl (32-bit) SUNWcslx (64-bit)
MT Level	Unsafe

SEE ALSO

`intro`(3), `maillock`(3MAIL), `attributes`(5)

NAME

libmalloc – memory allocation library

SYNOPSIS

```
cc [ flag . . . ] file . . . -lmalloc [ library . . . ]
```

DESCRIPTION

Functions in this library provide routines for memory allocation. These routines are space-efficient but have lower performance. Their usage can result in serious performance degradation.

INTERFACES

The shared object `libmalloc.so.1` provides the public interfaces defined below. See `intro`(3) for additional information on shared object interfaces.

`_cfree`	`_mallinfo`	`_mallopt`	`calloc`
`cfree`	`free`	`mallinfo`	`malloc`
`mallopt`	`realloc`		

FILES

`/usr/lib/libmalloc.a`
archive library

`/usr/lib/libmalloc.so.1`
shared object

`/usr/lib/64/libmalloc.so.1`
64-bit shared object

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsl (32-bit) SUNWcslx (64-bit)
MT-Level	Safe

SEE ALSO

`intro`(3), `malloc`(3MALLOC), `attributes`(5)

NAME

libmapmalloc – alternative memory allocator library

SYNOPSIS

```
cc [ flag . . . ] file . . . -lmapmalloc [ library . . . ]
#include <stdlib.h>
```

DESCRIPTION

Functions in this library provide `malloc` routines that use `mmap`(2) instead of `sbrk`(2) for acquiring heap space.

INTERFACES

The shared object `libmapmalloc.so.1` provides the public interfaces defined below. See `intro`(3) for additional information on shared object interfaces.

`calloc`	`cfree`	`free`	`mallinfo`	`malloc`
`mallopt`	`memalign`	`realloc`	`valloc`	

FILES

`/usr/lib/libmapmalloc.a`
archive library

`/usr/lib/libmapmalloc.so.1`
shared object

`/usr/lib/64/libmapmalloc.so.1`
64-bit shared object

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsl (32-bit) SUNWcslx (64-bit)
MT-Level	Safe

SEE ALSO

`pvs`(1), `mmap`(2), `sbrk`(2), `intro`(3), `malloc`(3C), `malloc`(3MALLOC), `mapmalloc`(3MALLOC), `attributes`(5)

NAME

libmd5 – MD5 hashing library

SYNOPSIS

```
cc [ flag . . . ] file . . . -lmd5 [ library . . . ]
#include <md5.h>
```

DESCRIPTION

Functions in this library provide MD5 hashing routines.

INTERFACES

The shared object `libmd5.so.1` provides the public interfaces defined below. See `intro`(3) for additional information on shared object interfaces.

```
MD5Final          MD5Init           MD5Update         md5_calc
```

FILES

`/usr/lib/libmd5.so.1`	shared object
`/usr/lib/64/libmd5.so.1`	64-bit shared object.

ATTRIBUTES

See `attributes`(5) for description of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsl (32-bit) SUNWcslx (64-bit)
MT Level	MT–Safe

SEE ALSO

`intro`(3), `attributes`(5)

NAME

libmenu – menus library

SYNOPSIS

```
cc [ flag . . . ] file . . . -lmenu [ library . . . ]
```

DESCRIPTION

Functions in this library provide menus using `libcurses`(3LIB) routines.

INTERFACES

The shared object `libmenu.so.1` provides the public interfaces defined below. See `intro`(3) for additional information on shared object interfaces.

`current_item`	`free_item`
`free_menu`	`item_count`
`item_description`	`item_index`
`item_init`	`item_name`
`item_opts`	`item_opts_off`
`item_opts_on`	`item_term`
`item_userptr`	`item_value`
`item_visible`	`menu_back`
`menu_driver`	`menu_fore`
`menu_format`	`menu_grey`
`menu_init`	`menu_items`
`menu_mark`	`menu_opts`
`menu_opts_off`	`menu_opts_on`
`menu_pad`	`menu_pattern`
`menu_sub`	`menu_term`
`menu_userptr`	`menu_win`
`new_item`	`new_menu`
`pos_menu_cursor`	`post_menu`
`scale_menu`	`set_current_item`
`set_item_init`	`set_item_opts`
`set_item_term`	`set_item_userptr`
`set_item_value`	`set_menu_back`
`set_menu_fore`	`set_menu_format`
`set_menu_grey`	`set_menu_init`

set_menu_items	set_menu_mark
set_menu_opts	set_menu_pad
set_menu_pattern	set_menu_sub
set_menu_term	set_menu_userptr
set_menu_win	set_top_row
top_row	unpost_menu

FILES

/usr/lib/libmenu.a	archive library
/usr/lib/libmenu.so.1	shared object
/usr/lib/64/libmenu.so.1	64-bit shared object

ATTRIBUTES

See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsl (32-bit) SUNWcslx (64-bit)
MT-Level	Unsafe

SEE ALSO

intro(3), libcurses(3LIB), attributes(5)

NAME

libmp – multiple precision library

SYNOPSIS

```
cc [ flag . . . ] file . . . -lmp [ library . . . ]
#include <mp.h>
```

DESCRIPTION

Functions in this library provide various multiple precision routines.

INTERFACES

The shared object `libmp.so.2` provides the public interfaces defined below. See `intro`(3) for additional information on shared object interfaces.

`mp_gcd`	`mp_itom`	`mp_madd`	`mp_mcmp`	`mp_mdiv`
`mp_mfree`	`mp_min`	`mp_mout`	`mp_msqrt`	`mp_msub`
`mp_mtox`	`mp_mult`	`mp_pow`	`mp_rpow`	`mp_sdiv`
`mp_xtom`				

FILES

`/usr/lib/libmp.a`	archive library
`/usr/lib/libmp.so.1`	shared object for binary compatibility only
`/usr/lib/libmp.so.2`	shared object
`/usr/lib/64/libmp.so.2`	64-bit shared object

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsl (32-bit) SUNWcslx (64-bit)
MT-Level	Unsafe

SEE ALSO

`pvs`(1), `intro`(3), `exp`(3M), `mp`(3MP), `attributes`(5)

NAME

libmtmalloc – multi-threaded memory allocator library

SYNOPSIS

```
cc [ flag . . . ] file . . . -lmtmalloc [ library . . . ]
#include <mtmalloc.h>
```

DESCRIPTION

Functions in this library provide `malloc` routines that provide concurrent access to heap space.

INTERFACES

The shared object `libmtmalloc.so.1` provides the public interfaces defined below. See `intro`(3) for additional information on shared object interfaces.

```
free            malloc          mallocctl       realloc
```

FILES

`/usr/lib/libmtmalloc.so.1`
shared object

`/usr/lib/64/libmtmalloc.so.1`
64–bit shared object

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsl (32-bit) SUNWcslx (64-bit)
MT-Level	Safe

SEE ALSO

`pvs`(1), `sbrk`(2), `intro`(3), `malloc`(3C), `malloc`(3MALLOC), `mapmalloc`(3MALLOC), `mtmalloc`(3MALLOC), `attributes`(5)

NAME

libnsl – network services library

SYNOPSIS

`cc [` *flag* `. . . ] ` *file* `. . . -lnsl [` *library* `. . . ]`

DESCRIPTION

Functions in this library provide routines that provide a transport-level interface to networking services for applications, facilities for machine-independent data representation, a remote procedure call mechanism, and other networking services useful for application programs.

Many features in this library are implemented upon dynamic linking and will not function correctly if the library is statically linked. Additionally, an application that statically links this library will not be compliant with the System V Application Binary Interface.

Some symbols are not intended to be referenced directly. Rather, they are exposed because they are used elsewhere through a private interface. One such example is the set of symbols beginning with the `_xti` prefix. Those symbols are used in implementing the X/Open Transport Interface (XTI) interfaces documented in `libxnet`. See `libxnet`(3LIB).

INTERFACES

The shared object `libnsl.so.1` provides the public interfaces defined below. See `intro`(3) for additional information on shared object interfaces.

`__rpc_createerr`	`__t_errno`
`_nderror`	`_null_auth`
`_xti_accept`	`_xti_alloc`
`_xti_bind`	`_xti_close`
`_xti_connect`	`_xti_error`
`_xti_free`	`_xti_getinfo`
`_xti_getprotaddr`	`_xti_getstate`
`_xti_listen`	`_xti_look`
`_xti_open`	`_xti_optmgmt`
`_xti_rcv`	`_xti_rcvconnect`
`_xti_rcvdis`	`_xti_rcvrel`
`_xti_rcvreldata`	`_xti_rcvudata`
`_xti_rcvuderr`	`_xti_rcvv`
`_xti_rcvvudata`	`_xti_snd`
`_xti_snddis`	`_xti_sndrel`
`_xti_sndreldata`	`_xti_sndudata`

_xti_sndv	_xti_sndvudata
_xti_strerror	_xti_sync
_xti_sysconf	_xti_unbind
_xti_xns5_accept	_xti_xns5_snd
auth_destroy	authdes_create
authdes_getucred	authdes_lock
authdes_seccreate	authnone_create
authsys_create	authsys_create_default
callrpc	clnt_broadcast
clnt_call	clnt_control
clnt_create	clnt_create_timed
clnt_create_vers	clnt_create_vers_timed
clnt_destroy	clnt_dg_create
clnt_door_create	clnt_freeres
clnt_geterr	clnt_pcreateerror
clnt_perrno	clnt_perror
clnt_raw_create	clnt_spcreateerror
clnt_sperrno	clnt_sperror
clnt_tli_create	clnt_tp_create
clnt_tp_create_timed	clnt_vc_create
clntraw_create	clnttcp_create
clntudp_bufcreate	clntudp_create
dbmclose	dbminit
delete	des_setparity
dial	doconfig
endhostent	endnetconfig
endnetpath	endrpcent
fetch	firstkey
freehostent	freenetconfigent
get_myaddress	gethostbyaddr

gethostbyaddr_r	gethostbyname
gethostbyname_r	gethostent
gethostent_r	getipnodebyaddr
getipnodebyname	getnetconfig
getnetconfigent	getnetname
getnetpath	getpublickey
getrpcbyname	getrpcbyname_r
getrpcbynumber	getrpcbynumber_r
getrpcent	getrpcent_r
getrpcport	getsecretkey
h_errno	host2netname
inet_addr	inet_netof
inet_ntoa	inet_ntoa_r
inet_ntop	inet_pton
key_decryptsession	key_encryptsession
key_gendes	key_secretkey_is_set
key_setsecret	maxbno
nc_perror	nc_sperror
netdir_free	netdir_getbyaddr
netdir_getbyname	netdir_options
netdir_perror	netdir_sperror
netname2host	netname2user
nextkey	nis_add
nis_add_entry	nis_addmember
nis_checkpoint	nis_clone_object
nis_creategroup	nis_data
nis_destroy_object	nis_destroygroup
nis_dir_cmp	nis_domain_of
nis_dump	nis_dumplog
nis_find_item	nis_finddirectory

nis_first_entry	nis_free_request
nis_freenames	nis_freeresult
nis_freeservlist	nis_freetags
nis_get_request	nis_get_static_storage
nis_getnames	nis_getservlist
nis_in_table	nis_insert_item
nis_insert_name	nis_ismember
nis_leaf_of	nis_leaf_of_r
nis_lerror	nis_list
nis_local_directory	nis_local_group
nis_local_host	nis_local_principal
nis_lookup	nis_make_error
nis_make_rpchandle	nis_mkdir
nis_modify	nis_modify_entry
nis_name_of	nis_next_entry
nis_perror	nis_ping
nis_print_directory	nis_print_entry
nis_print_group	nis_print_group_entry
nis_print_link	nis_print_object
nis_print_rights	nis_print_table
nis_read_obj	nis_remove
nis_remove_entry	nis_remove_item
nis_remove_name	nis_removemember
nis_rmdir	nis_servstate
nis_sperrno	nis_sperror
nis_sperror_r	nis_stats
nis_verifygroup	nis_write_obj
pmap_getmaps	pmap_getport
pmap_rmtcall	pmap_set
pmap_unset	registerrpc

rpc_broadcast	rpc_broadcast_exp
rpc_call	rpc_control
rpc_createerr	rpc_gss_get_error
rpc_gss_get_mech_info	rpc_gss_get_mechanisms
rpc_gss_get_principal_name	rpc_gss_get_versions
rpc_gss_getcred	rpc_gss_is_installed
rpc_gss_max_data_length	rpc_gss_mech_to_oid
rpc_gss_qop_to_num	rpc_gss_seccreate
rpc_gss_set_callback	rpc_gss_set_defaults
rpc_gss_set_svc_name	rpc_gss_svc_max_data_length
rpc_reg	rpcb_getaddr
rpcb_getmaps	rpcb_gettime
rpcb_rmtcall	rpcb_set
rpcb_unset	sethostent
setnetconfig	setnetpath
setrpcent	store
svc_auth_reg	svc_control
svc_create	svc_destroy
svc_dg_create	svc_dg_enablecache
svc_done	svc_door_create
svc_exit	svc_fd_create
svc_fdset	svc_freeargs
svc_get_local_cred	svc_getargs
svc_getreq	svc_getreq_common
svc_getreq_poll	svc_getreqset
svc_getrpccaller	svc_max_pollfd
svc_pollfd	svc_raw_create
svc_reg	svc_register
svc_run	svc_sendreply
svc_tli_create	svc_tp_create

svc_unreg	svc_unregister
svc_vc_create	svcerr_auth
svcerr_decode	svcerr_noproc
svcerr_noprog	svcerr_progvers
svcerr_systemerr	svcerr_weakauth
svcfd_create	svcraw_create
svctcp_create	svcudp_bufcreate
svcudp_create	t_accept
t_alloc	t_bind
t_close	t_connect
t_errno	t_error
t_free	t_getinfo
t_getname	t_getstate
t_listen	t_look
t_nerr	t_open
t_optmgmt	t_rcv
t_rcvconnect	t_rcvdis
t_rcvrel	t_rcvudata
t_rcvuderr	t_snd
t_snddis	t_sndrel
t_sndudata	t_strerror
t_sync	t_unbind
taddr2uaddr	uaddr2taddr
undial	user2netname
xdr_accepted_reply	xdr_array
xdr_authsys_parms	xdr_bool
xdr_bytes	xdr_callhdr
xdr_callmsg	xdr_char
xdr_destroy	xdr_double
xdr_enum	xdr_float

xdr_free	xdr_getpos
xdr_hyper	xdr_inline
xdr_int	xdr_int16_t
xdr_int32_t	xdr_int64_t
xdr_int8_t	xdr_long
xdr_longlong_t	xdr_opaque
xdr_opaque_auth	xdr_pointer
xdr_quadruple	xdr_reference
xdr_rejected_reply	xdr_replymsg
xdr_setpos	xdr_short
xdr_sizeof	xdr_string
xdr_u_char	xdr_u_hyper
xdr_u_int	xdr_u_long
xdr_u_longlong_t	xdr_u_short
xdr_uint16_t	xdr_uint32_t
xdr_uint64_t	xdr_uint8_t
xdr_union	xdr_vector
xdr_void	xdr_wrapstring
xdrmem_create	xdrrec_create
xdrrec_endofrecord	xdrrec_eof
xdrrec_readbytes	xdrrec_skiprecord
xdrstdio_create	xprt_register
xprt_unregister	yp_all
yp_bind	yp_first
yp_get_default_domain	yp_master
yp_match	yp_next
yp_order	yp_unbind
yp_update	yperr_string
ypprot_err	

The following interface is unique to the 32-bit version of this library:

`_new_svc_fdset`

FILES

`/usr/lib/libnsl.a`	archive library
`/usr/lib/libnsl.so.1`	shared object
`/usr/lib/64/libnsl.so.1`	64-bit shared object

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsl, SUNWarc (32-bit) SUNWcslx (64-bit)
MT-Level	Safe with exceptions

SEE ALSO

`pvs`(1), `intro`(2), `intro`(3), `libxnet`(3LIB), `attributes`(5)

NAME

libnvpair – name-value pair library

SYNOPSIS

```
cc [ flag . . . ] file . . . -lnvpair [ library . . . ]
#include <libnvpair.h>
```

DESCRIPTION

Functions in this library provide various name-value pair routines.

INTERFACES

The shared object `libnvpair.so.1` provides the public interfaces defined below. See `intro`(3) for additional information on shared object interfaces.

nvlist_add_boolean	nvlist_add_byte
nvlist_add_byte_array	nvlist_add_int16
nvlist_add_int16_array	nvlist_add_int32
nvlist_add_int32_array	nvlist_add_int64
nvlist_add_int64_array	nvlist_add_string
nvlist_add_string_array	nvlist_add_uint16
nvlist_add_uint16_array	nvlist_add_uint32
nvlist_add_uint32_array	nvlist_add_uint64
nvlist_add_uint64_array	nvlist_alloc
nvlist_dup	nvlist_free
nvlist_lookup_boolean	nvlist_lookup_byte
nvlist_lookup_byte_array	nvlist_lookup_int16
nvlist_lookup_int16_array	nvlist_lookup_int32
nvlist_lookup_int32_array	nvlist_lookup_int64
nvlist_lookup_int64_array	nvlist_lookup_string
nvlist_lookup_string_array	nvlist_lookup_uint16
nvlist_lookup_uint16_array	nvlist_lookup_uint32
nvlist_lookup_uint32_array	nvlist_lookup_uint64
nvlist_lookup_uint64_array	nvlist_next_nvpair
nvlist_pack	nvlist_remove
nvlist_remove_all	nvlist_size
nvlist_unpack	nvpair_name
nvpair_type	nvpair_value_byte
nvpair_value_byte_array	nvpair_value_int16

nvpair_value_int16_array	nvpair_value_int32
nvpair_value_int32_array	nvpair_value_int64
nvpair_value_int64_array	nvpair_value_string
nvpair_value_string_array	nvpair_value_uint16
nvpair_value_uint16_array	nvpair_value_uint32
nvpair_value_uint32_array	nvpair_value_uint64
nvpair_value_uint64_array	

FILES

`/usr/lib/libnvpair.so.1`
shared object

`/usr/lib/64/libnvpair.so.1`
64-bit shared object

ATTRIBUTES

See `attributes`(5) for description of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsl (32-bit) SUNWcslx (64-bit)
MT-Level	MT-Safe

SEE ALSO

`intro`(3), `libnvpair`(3NVPAIR), `attributes`(5)

NAME

libpam – PAM (Pluggable Authentication Module) library

SYNOPSIS

```
cc [ flag . . . ] file . . . -lpam [ library . . . ]
#include <security/pam_appl.h>
```

DESCRIPTION

Functions in this library provide routines for the Pluggable Authentication Module (PAM).

INTERFACES

The shared object `libpam.so.1` provides the public interfaces defined below. See `intro`(3) for additional information on shared object interfaces.

`pam_acct_mgmt`	`pam_authenticate`
`pam_chauthtok`	`pam_close_session`
`pam_end`	`pam_get_data`
`pam_get_item`	`pam_get_user`
`pam_getenv`	`pam_getenvlist`
`pam_open_session`	`pam_putenv`
`pam_set_data`	`pam_set_item`
`pam_setcred`	`pam_start`
`pam_strerror`	

FILES

`/usr/lib/libpam.so.1`
shared object

`/etc/pam.conf`
configuration file

`/usr/lib/security/pam_dial_auth.so.1`
authentication management PAM module for dialups

`/usr/lib/security/pam_rhosts_auth.so.1`
authentication management PAM modules that use `ruserok()`

`/usr/lib/security/pam_sample.so.1`
sample PAM module

`/usr/lib/security/pam_unix.so.1`
authentication, account, session and password management PAM module

ATTRIBUTES

See `attributes`(5) for description of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsl

ATTRIBUTE TYPE	ATTRIBUTE VALUE
MT Level	MT-Safe with exceptions

SEE ALSO

pvs(1), intro(3), pam(3PAM), pam.conf(4), attributes(5), pam_authtok_check(5), pam_authtok_get(5), pam_authtok_store(5), pam_dial_auth(5), pam_dhkeys(5), pam_passwd_auth(5), pam_rhosts_auth(5), pam_sample(5), pam_unix(5), pam_unix_account(5), pam_unix_auth(5), pam_unix_session(5)

NOTES

The functions in libpam are MT-Safe only if each thread within the multithreaded application uses its own PAM handle.

The pam_unix(5) module might not be supported in a future release. Similar functionality is provided by pam_authtok_check(5), pam_authtok_get(5), pam_authtok_store(5), pam_dhkeys(5), pam_passwd_auth(5), pam_unix_account(5), pam_unix_auth(5), and pam_unix_session(5).

NAME

libpanel – panels library

SYNOPSIS

```
cc [ flag . . . ] file . . . -lpanel [ library . . . ]
```

DESCRIPTION

Functions in this library provide panels using `libcurses`(3LIB) routines.

INTERFACES

The shared object `libpanel.so.1` provides the public interfaces defined below. See `intro`(3) for additional information on shared object interfaces.

`bottom_panel`	`del_panel`
`hide_panel`	`move_panel`
`new_panel`	`panel_above`
`panel_below`	`panel_hidden`
`panel_userptr`	`panel_window`
`replace_panel`	`set_panel_userptr`
`show_panel`	`top_panel`
`update_panels`	

FILES

`/usr/lib/libpanel.a`	archive library
`/usr/lib/libpanel.so.1`	shared object
`/usr/lib/64/libpanel.so.1`	64-bit shared object

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsl, SUNWarc (32-bit) SUNWcslx (64-bit)
MT-Level	Unsafe

SEE ALSO

`intro`(3), `libcurses`(3LIB), `attributes`(5)

NAME

libpctx – process context library

SYNOPSIS

```
cc [ flag . . . ] file . . . -lpctx [ library . . . ]
```

DESCRIPTION

Functions in this library provide a simple means to access the underlying facilities of `proc`(4) to allow a controlling process to manipulate the state of a controlled process.

This library is primarily for use in conjunction with the `libcpc`(3LIB) library. Used together, these libraries allow developers to construct tools that can manipulate CPU performance counters in other processes. The `cputrack`(1) utility is an example of such a tool.

INTERFACES

The shared object `libpctx.so.1` provides the public interfaces defined below. See `intro`(3) for additional information on shared object interfaces.

```
pctx_capture                    pctx_create
pctx_release                    pctx_run
pctx_set_events
```

FILES

`/usr/lib/libpctx.so.1`	shared object
`/usr/lib/64/libpctx.so.1`	64-bit shared object

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcpcu (32-bit) SUNWcpcux (64-bit)
MT-Level	Safe

SEE ALSO

`cputrack`(1), `intro`(3), `cpc`(3CPC), `libcpc`(3LIB), `proc`(4), `attributes`(5)

NAME

libpicl – PICL library

SYNOPSIS

```
cc [ flag . . . ] file . . . -lpicl [ library . . . ]
#include <picl.h>
```

DESCRIPTION

Functions in this library are used to interface with the PICL daemon to access information from the PICL tree.

INTERFACES

The shared object `libpicl.so.1` provides the public interfaces defined below. See `intro`(3) for additional information on shared object interfaces.

`picl_get_first_prop`	`picl_get_next_by_col`
`picl_get_next_by_row`	`picl_get_next_prop`
`picl_get_prop_by_name`	`picl_get_propinfo`
`picl_get_propinfo_by_name`	`picl_get_propval`
`picl_get_propval_by_name`	`picl_get_root`
`picl_initialize`	`picl_set_propval`
`picl_set_propval_by_name`	`picl_shutdown`
`picl_strerror`	`picl_wait`
`picl_walk_tree_by_class`	

FILES

`/usr/lib/libpicl.so.1`
shared object

`/usr/lib/64/libpicl.so.1`
64-bit shared object

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWpiclu (32-bit) SUNWpiclx (64-bit)
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO

`pvs`(1), `intro`(3), `libpicl`(3PICL), `attributes`(5)

NAME

libpicltree – PICL plug-in library

SYNOPSIS

```
cc [ flag . . . ] file . . . -lpicltree [ library . . . ]
#include <picltree.h>
```

DESCRIPTION

Functions in this library are used to by PICL plug-in modules to register with the PICL daemon and to publish information in the PICL tree.

INTERFACES

The shared object `libpicltree.so.1` provides the public interfaces defined below. See `intro`(3) for additional information on shared object interfaces.

`picld_plugin_register`	`ptree_add_node`
`ptree_add_prop`	`ptree_add_row_to_table`
`ptree_create_and_add_node`	`ptree_create_and_add_prop`
`ptree_create_node`	`ptree_create_prop`
`ptree_create_table`	`ptree_delete_node`
`ptree_delete_prop`	`ptree_destroy_node`
`ptree_destroy_prop`	`ptree_find_node`
`ptree_get_first_prop`	`ptree_get_next_by_col`
`ptree_get_next_by_row`	`ptree_get_next_prop`
`ptree_get_node_by_path`	`ptree_get_prop_by_name`
`ptree_get_propinfo`	`ptree_get_propval`
`ptree_get_propval_by_name`	`ptree_get_root`
`ptree_init_propinfo`	`ptree_post_event`
`ptree_register_handler`	`ptree_unregister_handler`
`ptree_update_propval`	`ptree_update_propval_by_name`
`ptree_walk_tree_by_class`	

FILES

`/usr/lib/libpicltree.so.1`
shared object

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWpiclu
Interface Stability	Evolving

ATTRIBUTE TYPE	ATTRIBUTE VALUE
MT-Level	MT-Safe

SEE ALSO pvs(1), intro(3), libpicltree(3PICLTREE), attributes(5)

NAME

libplot, lib300, lib300s, lib4014, lib450, libvt0 – graphics interface libraries

SYNOPSIS

```
cc [ flag . . . ] file . . . -lplot [ library  . . . ]
#include <plot.h>
```

DESCRIPTION

Functions in this library generate graphics output.

INTERFACES

The shared object `libplot.so.1` provides the public interfaces defined below. See `intro`(3) for additional information on shared object interfaces.

`arc`	`box`	`circle`	`closepl`	`closevt`
`cont`	`dot`	`erase`	`label`	`line`
`linemod`	`move`	`openpl`	`openvt`	`point`
`space`				

FILES

`/usr/lib/libplot.a`
archive library

`/usr/lib/libplot.so.1`
shared object

`/usr/lib/sparcv9/libplot.so.1`
64-bit shared object

`/usr/lib/lib300.a`
archive library

`/usr/lib/lib300.so.1`
shared object

`/usr/lib/sparcv9/lib300.so.1`
64-bit shared object

`/usr/lib/lib300s.a`
archive library

`/usr/lib/lib300s.so.1`
shared object

`/usr/lib/sparcv9/lib300s.so.1`
64-bit shared object

`/usr/lib/lib4014.a`
archive library

`/usr/lib/lib4014.so.1`
shared object

`/usr/lib/sparcv9/lib4014.so.1`
64-bit shared object

`/usr/lib/lib450.a`
archive library

`/usr/lib/lib450.so.1`
shared object

`/usr/lib/sparcv9/lib450.so.1`
64-bit shared object

`/usr/lib/libvt0.a`
archive library

`/usr/lib/libvt0.so.1`
shared object

`/usr/lib/sparcv9/libvt0.so.1`
64-bit shared object

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsl, SUNWarc (32-bit) SUNWcslx (64-bit)
MT-Level	Unsafe

SEE ALSO

`pvs`(1), `intro`(3), `attributes`(5)

NAME

libpool – pool configuration manipulation library

SYNOPSIS

```
cc [ flag... ] file...  [ library... ]
#include <pool.h>
```

DESCRIPTION

The functions in this library define the interface for reading and writing resource pools configuration files, as well as that for commiting an existing configuration to becoming the running OS configuration (with respect to partitioning subsystems). The `<pool.h>` header provides type and function declarations for all library services.

The resource pools facility brings together process-bindable resources into a common abstraction called a pool. Processor sets and other entities can be configured, grouped, and labelled in a persistent fashion such that workload components can be associated with a subset of a system's total resources. The `libpool` library provides a C language API for accessing this functionality, while `pooladm`(1M), `poolbind`(1M), and `poolcfg`(1M) make this facility available through command invocations from a shell. Each of those manual pages describes aspects of the pools facility; this page describes the properties available to the various entities managed within the pools facility. These entities include the system, pools, and the `pset` resources for processor sets.

Each active entity within the resource pools framework can have an arbitrary collection of named, typed properties associated with it. Properties supported by the pools framework are listed, with descriptions, under each entity below. In general, resource properties may be one of five types: boolean, signed and unsigned integers, floating point, and string values.

All entities and resources support a string property for commenting purposes; this property is available for use by management applications to record descriptions and other administrator oriented data. The comment field is not used by the default pools commands, except when a configuration is initiated by the `poolcfg` utility, in which case an informative message is placed in the `system.comment` property for that configuration.

System

Property name	Type	Description
`system.bind-default`	boolean	If specified pool not found, bind to pool with 'pool.default' property set to true.
`system.comment`	string	User description of system.
`system.version`	int	`libpool` version required to manipulate this configuration.

The `system.bind-default` and `system.comment` properties are writable; the `system.version` property is not.

Pools

Property name	Type	Description
pool.active	boolean	Mark this pool as active, if true.
pool.comment	string	User description of pool.
pool.default	boolean	Mark this pool as the default pool, if true; see system.bind-default property.
pool.importance	int	Relative importance of this pool; for possible resource dispute resolution.
pool.name	string	User name for pool; used by `setproject`(3PROJECT) as value for 'project.pool' project attribute in `project`(4) database.
pool.scheduler	string	Scheduler class to which consumers of this pool will be bound. This property is optional and if not specified, the scheduler bindings for consumers of this pool are not affected.

All of the above listed properties are writable.

Processor Sets

Property name	Type	Description
`pset.comment`	string	User description of resource.
`pset.default`	boolean	Marks default processor set.
`pset.escapable`	boolean	Represents whether `PSET_NOESCAPE` is set for this pset (see `pset_setattr`(2))
`pset.max`	uint	Maximum number of CPUs permitted in this processor set.
`pset.min`	uint	Minimum number of CPUs permitted in this processor set.
`pset.name`	string	User name for resource.
`pset.size`	uint	Current number of CPUs in this processor set.
`pset.sys_id`	int	System-assigned processor set ID.
`pset.type`	string	Names resource type; value for all processor sets is `pset`.
`pset.units`	string	Identifies meaning of size-related properties; value for all processor sets is `population`.

The `pset.comment`, `pset.default`, `pset.escapable`, `pset.max`, `pset.min`, `pset.min`, and `pset.name` properties are writable; the `pset.size`, `pset.sys_id`, `pset.type`, and `pset.units` properties are not.

INTERFACES

The shared object `libpool.so.1` provides the public interfaces defined below. See `intro`(3) for additional information on shared object interfaces.

`pool_associate`	`pool_component_info`
`pool_component_to_elem`	`pool_conf_alloc`
`pool_conf_close`	`pool_conf_commit`
`pool_conf_export`	`pool_conf_free`
`pool_conf_info`	`pool_conf_location`
`pool_conf_open`	`pool_conf_remove`
`pool_conf_rollback`	`pool_conf_status`
`pool_conf_to_elem`	`pool_conf_validate`
`pool_create`	`pool_destroy`
`pool_dissociate`	`pool_dynamic_location`
`pool_error`	`pool_get_binding`
`pool_get_owning_resource`	`pool_get_pool`
`pool_get_property`	`pool_get_resource`
`pool_get_resource_binding`	`pool_info`
`pool_put_property`	`pool_query_components`
`pool_query_pool_resources`	`pool_query_pools`
`pool_query_resource_components`	`pool_query_resources`
`pool_resource_create`	`pool_resource_destroy`
`pool_resource_info`	`pool_resource_to_elem`
`pool_resource_transfer`	`pool_resource_xtransfer`
`pool_rm_property`	`pool_set_binding`
`pool_static_location`	`pool_strerror`
`pool_to_elem`	`pool_value_alloc`
`pool_value_free`	`pool_value_get_bool`
`pool_value_get_double`	`pool_value_get_int64`

pool_value_get_name
pool_value_get_string
pool_value_get_type
pool_value_get_uint64
pool_value_set_bool
pool_value_set_double
pool_value_set_int64
pool_value_set_name
pool_value_set_string
pool_value_set_uint64
pool_version
pool_walk_components
pool_walk_pools
pool_walk_properties
pool_walk_resources

FILES

/usr/lib/libpool.so.1
shared object

/usr/lib/sparcv9/libpool.so.1
64-bit shared object

ATTRIBUTES

See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWpool (32-bit) SUNWpoolx (64-bit)
CSI	Enabled
Interface Stability	Evolving
MT-Level	Unsafe

SEE ALSO

intro(3), pool_component_info(3POOL), pool_conf_open(3POOL), pool_conf_to_elem(3POOL), pool_create(3POOL), pool_error(3POOL), pool_get_binding(3POOL), pool_get_property(3POOL), pool_get_resource(3POOL), pool_resource_create(3POOL), pool_value_alloc(3POOL), pool_walk_pools(3POOL), attributes(5)

NOTES

Functions in libpool are unsafe for use in multithreaded applications where multiple configurations are being simultaneously manipulated by the application, due to shared state in the backing repository facility.

NAME

libproject – project database access library

SYNOPSIS

```
cc [ flag . . . ] file . . . -lproject [ library . . . ]
#include <project.h>
```

DESCRIPTION

Functions in this library provide various interfaces to extract data from the `project`(4) database. The header provides structure and function declarations for all library interfaces.

INTERFACES

The shared object `libproject.so.1` provides the public interfaces defined below. See `intro`(3) for additional information on shared object interfaces.

`endprojent`	`fgetprojent`
`getdefaultproj`	`getprojbyid`
`getprojbyname`	`getprojent`
`getprojidbyname`	`inproj`
`project_walk`	`setproject`
`setprojent`	

FILES

`/usr/lib/libproject.so.1`
shared object

`/usr/lib/64/libproject.so.1`
64-bit shared object

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsl (32-bit) SUNWcslx (64-bit)
Interface Stability	Evolving
MT-Level	Safe

SEE ALSO

`pvs`(1), `intro`(3), `getprojent`(3PROJECT), `project`(4), `attributes`(5), `standards`(5)

NAME

libpthread – POSIX threads library

SYNOPSIS

```
cc -mt [ flag . . . ] file . . . -lpthread [  -lrt  library . . . ]
```

DESCRIPTION

Functions in this library provide routines that provide POSIX threading support. See `standards`(5). This library is implemented as a filter on the threads library (see `libthread`(3LIB)).

INTERFACES

The shared object `libpthread.so.1` provides the public interfaces defined below. See `intro`(3) for additional information on shared object interfaces.

`__pthread_cleanup_pop`	`__pthread_cleanup_push`
`pthread_attr_destroy`	`pthread_attr_getdetachstate`
`pthread_attr_getguardsize`	`pthread_attr_getinheritsched`
`pthread_attr_getschedparam`	`pthread_attr_getschedpolicy`
`pthread_attr_getscope`	`pthread_attr_getstackaddr`
`pthread_attr_getstacksize`	`pthread_attr_init`
`pthread_attr_setdetachstate`	`pthread_attr_setguardsize`
`pthread_attr_setinheritsched`	`pthread_attr_setschedparam`
`pthread_attr_setschedpolicy`	`pthread_attr_setscope`
`pthread_attr_setstackaddr`	`pthread_attr_setstacksize`
`pthread_cancel`	`pthread_cond_broadcast`
`pthread_cond_destroy`	`pthread_cond_init`
`pthread_cond_reltimedwait_np`	`pthread_cond_signal`
`pthread_cond_timedwait`	`pthread_cond_wait`
`pthread_condattr_destroy`	`pthread_condattr_getpshared`
`pthread_condattr_init`	`pthread_condattr_setpshared`
`pthread_create`	`pthread_detach`
`pthread_equal`	`pthread_exit`
`pthread_getconcurrency`	`pthread_getschedparam`
`pthread_getspecific`	`pthread_join`
`pthread_key_create`	`pthread_key_delete`
`pthread_kill`	`pthread_mutex_consistent_np`
`pthread_mutex_destroy`	`pthread_mutex_getprioceiling`

`pthread_mutex_init`	`pthread_mutex_lock`
`pthread_mutex_setprioceiling`	`pthread_mutex_trylock`
`pthread_mutex_unlock`	`pthread_mutexattr_destroy`
`pthread_mutexattr_getprioceiling`	`pthread_mutexattr_getprotocol`
`pthread_mutexattr_getpshared`	`pthread_mutexattr_getrobust_np`
`pthread_mutexattr_gettype`	`pthread_mutexattr_init`
`pthread_mutexattr_setprioceiling`	`pthread_mutexattr_setprotocol`
`pthread_mutexattr_setpshared`	`pthread_mutexattr_setrobust_np`
`pthread_mutexattr_settype`	`pthread_once`
`pthread_rwlock_destroy`	`pthread_rwlock_init`
`pthread_rwlock_rdlock`	`pthread_rwlock_tryrdlock`
`pthread_rwlock_trywrlock`	`pthread_rwlock_unlock`
`pthread_rwlock_wrlock`	`pthread_rwlockattr_destroy`
`pthread_rwlockattr_getpshared`	`pthread_rwlockattr_init`
`pthread_rwlockattr_setpshared`	`pthread_self`
`pthread_setcancelstate`	`pthread_setcanceltype`
`pthread_setconcurrency`	`pthread_setschedparam`
`pthread_setspecific`	`pthread_sigmask`
`pthread_testcancel`	

FILES

`/usr/lib/libpthread.so.1`
a filter on `libthread.so.1`

`/usr/lib/64/libpthread.so.1`
a filter on the 64-bit version of `libthread.so.1`

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsl (32-bit) SUNWcslx (64-bit)
MT-Level	Safe

SEE ALSO | `pvs`(1), `intro`(2), `intro`(3), `libpthread`(3LIB), `libthread`(3LIB), `libthread_db`(3LIB), `libthread_db`(3THR), `threads`(3THR), `attributes`(5)

NAME

librac – remote asynchronous calls library

SYNOPSIS

```
cc [ flag . . . ] file . . . -lrac  -lnsl  [ library  . . . ]
#include <rpc/rpc.h>
#include <rpc/rac.h>
```

DESCRIPTION

Functions in this library provide a remote asynchronous call interface to the RPC library.

INTERFACES

The shared object `librac.so.1` provides the public interfaces defined below. See `intro`(3) for additional information on shared object interfaces.

`clnt_create`	`clnt_create_vers`
`clnt_dg_create`	`clnt_tli_create`
`clnt_tp_create`	`clnt_vc_create`
`rac_drop`	`rac_poll`
`rac_recv`	`rac_send`
`rac_senderr`	`rpcb_getaddr`
`rpcb_getmaps`	`rpcb_gettime`
`rpcb_rmtcall`	`rpcb_set`
`rpcb_taddr2uaddr`	`rpcb_uaddr2taddr`
`rpcb_unset`	`xdrrec_create`
`xdrrec_endofrecord`	`xdrrec_eof`
`xdrrec_readbytes`	`xdrrec_skiprecord`

FILES

`/usr/lib/librac.a`	archive library
`/usr/lib/librac.so.1`	shared object
`/usr/lib/64/librac.so.1`	64-bit shared object file

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsl, SUNWarc (32-bit) SUNWcslx (64-bit)
MT-Level	Unsafe

SEE ALSO

`pvs`(1), `intro`(3), `rpc_rac`(3RAC), `attributes`(5)

NAME

libresolv – resolver library

SYNOPSIS

```
cc [ flag . . . ] file . . . -lresolv  -lsocket   -lnsl  [ library . . . ]
#include <sys/types.h>
#include <netinet/in.h>
#include <arpa/nameser.h>
#include <resolv.h>
#include <netdb.h>
```

DESCRIPTION

Functions in this library provide for creating, sending, and interpreting packets to the Internet domain name servers.

INTERFACES

The shared object `libresolv.so.2` provides the public interfaces defined below. See `intro`(3) for additional information on shared object interfaces.

`__dn_skipname`	`__fp_query`
`__hostalias`	`__p_cdname`
`__p_class`	`__p_query`
`__p_time`	`__p_type`
`__putlong`	`_getlong`
`_getshort`	`_res`
`dn_comp`	`dn_expand`
`fp_resstat`	`h_errno`
`herror`	`hstrerror`
`res_hostalias`	`res_init`
`res_mkquery`	`res_nclose`
`res_ninit`	`res_nmkquery`
`res_nquery`	`res_nquerydomain`
`res_nsearch`	`res_nsend`
`res_nsendsigned`	`res_query`
`res_querydomain`	`res_search`
`res_send`	`res_update`

Programs are expected to use the aliases defined in `<resolv.h>` rather than calling the "__" prefixed procedures, as indicated in the following table. Use of the routines in the first column is discouraged.

FUNCTION REFERENCED	ALIAS TO USE
__dn_skipname	dn_skipname
__fp_query	fp_query
__putlong	putlong
__p_cdname	p_cdname
__p_class	p_class
__p_time	p_time
__p_type	p_type

FILES

`/usr/lib/libresolv.so.1`
shared object for backward compatibility only

`/usr/lib/64/libresolv.so.1`
64–bit shared object for backward compatibility only

`/usr/lib/libresolv.so.2`
shared object

`/usr/lib/64/libresolv.so.2`
64–bit shared object

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsl (32-bit) SUNWcslx (64-bit)
MT-Level	See `resolver`(3RESOLV)
Interface Stability	BIND 8.2.4

SEE ALSO

`pvs`(1), `intro`(3), `resolver`(3RESOLV), `attributes`(5)

NAME

librpcsoc – obsolete RPC library

SYNOPSIS

```
cc [ flag . . . ] -I /usr/ucbinclude file . . . -L /usr/libucb \
-R /usr/libucb -lrpcsoc [ library . . . ]
#include <rpc/rpc.h>
```

DESCRIPTION

Functions in this library implement socket based RPC calls (using socket calls, not TLI). Applications that require this library should link it before `libnsl`, which implements the same calls over TLI .

This library is provided for compatibility only; new applications should not link in this library.

INTERFACES

The shared object `librpcsoc.so.1` provides the public interfaces defined below. See `intro`(3) for additional information on shared object interfaces.

`clnttcp_create`	`clntudp_bufcreate`
`clntudp_create`	`get_myaddress`
`getrpcport`	`rtime`
`svcfd_create`	`svctcp_create`
`svcudp_bufcreate`	`svcudp_create`
`svcudp_enablecache`	

FILES

`/usr/ucblib/librpcsoc.so.1`
shared object

`/usr/ucblib/64/librpcsoc.so.1`
64–bit shared object

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWscpu (32-bit) SUNWscpux (64-bit)
MT-Level	Unsafe

SEE ALSO

`pvs`(1), `intro`(3), `rpc_soc`(3NSL), `libnsl`(3LIB), `attributes`(5)

NAME

librpcsvc – RPC services library

SYNOPSIS

```
cc [ flag . . . ] file . . . -lrpcsvc [ library . . . ]
#include <rpc/rpc.h>
#include <rpcsvc/rstat.h>
```

DESCRIPTION

Functions in this library provide RPC services. See the manual pages in Section 3RPC for the individual functions.

INTERFACES

The shared object `librpcsvc.so.1` provides the public interfaces defined below. See `intro`(3) for additional information on shared object interfaces.

`havedisk`	`rnusers`
`rstat`	`rusers`
`rwall`	`xdr_statstime`
`xdr_statsvar`	`xdr_utmpidlearr`

FILES

`/usr/lib/librpcsvc.a`
archive library

`/usr/lib/librpcsvc.so.1`
shared object

`/usr/lib/64/librpcsvc.so.1`
64-bit shared object

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsl, SUNWarc (32-bit) SUNWcslx (64-bit)
MT-Level	Safe

SEE ALSO

`pvs`(1), `intro`(3), `rstat`(3RPC), `attributes`(5)

NAME

librsm – remote shared memory interface library

SYNOPSIS

```
cc [ flag . . . ] file . . . -lrsm [ library . . . ]
#include <rsmapi.h>
```

DESCRIPTION

The functions in this library provide an interface for OS bypass messaging for applications over high-speed interconnects, including facilities to set up low-latency, high-bandwidth interprocess communication mechanisms and to perform I/O.

INTERFACES

The shared object `librsm.so.2` provides the public interfaces defined below. See `intro`(3) for additional information on shared object interfaces.

`rsm_create_localmemory_handle`	`rsm_free_interconnect_topology`
`rsm_free_localmemory_handle`	`rsm_get_controller`
`rsm_get_controller_attr`	`rsm_get_interconnect_topology`
`rsm_get_segmentid_range`	`rsm_intr_signal_post`
`rsm_intr_signal_wait`	`rsm_memseg_export_create`
`rsm_memseg_export_destroy`	`rsm_memseg_export_publish`
`rsm_memseg_export_rebind`	`rsm_memseg_export_republish`
`rsm_memseg_export_unpublish`	`rsm_memseg_get_pollfd`
`rsm_memseg_import_close_barrier`	`rsm_memseg_import_connect`
`rsm_memseg_import_destroy_barrier`	`rsm_memseg_import_disconnect`
`rsm_memseg_import_get`	`rsm_memseg_import_get16`
`rsm_memseg_import_get32`	`rsm_memseg_import_get64`
`rsm_memseg_import_get8`	`rsm_memseg_import_get_mode`
`rsm_memseg_import_getv`	`rsm_memseg_import_init_barrier`
`rsm_memseg_import_map`	`rsm_memseg_import_open_barrier`
`rsm_memseg_import_order_barrier`	`rsm_memseg_import_put`
`rsm_memseg_import_put16`	`rsm_memseg_import_put32`
`rsm_memseg_import_put64`	`rsm_memseg_import_put8`
`rsm_memseg_import_putv`	`rsm_memseg_import_set_mode`
`rsm_memseg_import_unmap`	`rsm_memseg_release_pollfd`
`rsm_release_controller`	

FILES

`/usr/lib/librsm.so.2`
shared object

`/usr/lib/64/librsm.so.2`
64-bit shared object

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWrsm (32-bit) SUNWrsmx (64-bit)
Interface Stability	Evolving
MT-Level	Safe

SEE ALSO

`intro`(2), `intro`(3), `attributes`(5)

NAME

librt, libposix4 – POSIX.1b Realtime Extensions library

SYNOPSIS

```
cc [ flag . . . ] file . . . -lrt [ library . . . ]
```

```
cc [ flag . . . ] file . . . -lposix4 [ library . . . ]
```

DESCRIPTION

Functions in this library provide most of the interfaces specified by the POSIX.1b Realtime Extension. See `standards`(5). Specifically, this includes the interfaces defined under the Asynchronous I/O, Message Passing, Process Scheduling, Realtime Signals Extension, Semaphores, Shared Memory Objects, Synchronized I/O, and Timers options. The interfaces defined under the Memory Mapped Files, Process Memory Locking, and Range Memory Locking options are provided in `libc`(3LIB).

See the man pages for the individual interfaces in section 3RT for information on required headers.

The name `libposix4` is maintained for backward compatibility and should be avoided. `librt` is the preferred name for this library.

INTERFACES

The shared objects `librt.so.1` and `libposix4.so.1` provide the public interfaces defined below. See `intro`(3) for additional information on shared object interfaces.

`aio_cancel`	`aio_error`
`aio_fsync`	`aio_read`
`aio_return`	`aio_suspend`
`aio_waitn`	`aio_write`
`clock_getres`	`clock_gettime`
`clock_settime`	`close`
`fdatasync`	`fork`
`lio_listio`	`mq_close`
`mq_getattr`	`mq_notify`
`mq_open`	`mq_receive`
`mq_send`	`mq_setattr`
`mq_unlink`	`nanosleep`
`sched_get_priority_max`	`sched_get_priority_min`
`sched_getparam`	`sched_getscheduler`
`sched_rr_get_interval`	`sched_setparam`
`sched_setscheduler`	`sched_yield`
`sem_close`	`sem_destroy`

sem_getvalue	sem_init
sem_open	sem_post
sem_trywait	sem_unlink
sem_wait	shm_open
shm_unlink	sigqueue
sigtimedwait	sigwaitinfo
timer_create	timer_delete
timer_getoverrun	timer_gettime
timer_settime	

The following interfaces are unique to the 32-bit version of this library:

aio_cancel64	aio_error64
aio_fsync64	aio_read64
aio_return64	aio_suspend64
aio_write64	lio_listio64

FILES

/usr/lib/librt.so.1
shared object

/usr/lib/64/librt.so.1
64-bit shared object file

/usr/lib/libposix4.so.1
shared object

/usr/lib/64/libposix4.so.1
64-bit shared object file

ATTRIBUTES

See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsl (32-bit) SUNWcslx (64-bit)
MT-Level	Safe

SEE ALSO

pvs(1), intro(3), libc(3LIB), attributes(5), standards(5)

NAME

librtld_db – runtime linker debugging library

SYNOPSIS

```
cc [ flag ... ] file ... -lrtld_db [ library ... ]
#include <proc_service.h>
#include <rtld_db.h>
```

DESCRIPTION

Functions is this library are useful for building debuggers for dynamically linked programs. For a full description of these interfaces refer to the *Linker and Libraries Guide*.

INTERFACES

The shared object `librtld_db.so.1` provides the public interfaces defined below. See `intro`(3) for additional information on shared object interfaces.

`rd_delete`	`rd_errstr`
`rd_event_addr`	`rd_event_enable`
`rd_event_getmsg`	`rd_init`
`rd_loadobj_iter`	`rd_log`
`rd_new`	`rd_objpad_enable`
`rd_plt_resolution`	`rd_reset`

FILES

`/usr/lib/librtld_db.so.1`
shared object

`/usr/lib/64/librtld_db.so.1`
64-bit shared object

ATTRIBUTES

See `attributes`(5) for description of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsl (32-bit) SUNWcslx (64-bit)
MT-Level	Safe

SEE ALSO

`ld.so.1`(1), `pvs`(1), `intro`(3), `rtld_db`(3EXT), `attributes`(5)

Linker and Libraries Guide

NAME

libsec – File Access Control List library

SYNOPSIS

```
cc [ flag . . . ] file . . . -lsec [ library . . . ]
#include <sys/acl.h>
```

DESCRIPTION

Functions in this library provide comparison and manipulation of File Access Control Lists.

INTERFACES

The shared object `libsec.so.1` provides the public interfaces defined below. See `intro`(3) for additional information on shared object interfaces.

`aclcheck`	`aclfrommode`	`aclfromtext`
`aclsort`	`acltomode`	`acltotext`

FILES

`/usr/lib/libsec.a`	archive library
`/usr/lib/libsec.so.1`	shared object
`/usr/lib/64/libsec.so.1`	64-bit shared object

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsl, SUNWarc (32-bit) SUNWcslx (64-bit)
Interface Stability	Evolving
MT-Level	Unsafe

SEE ALSO

`pvs`(1), `intro`(3), `attributes`(5)

NAME

libsecdb – security attributes database library

SYNOPSIS

```
cc [ flag . . . ] file . . . -lsecdb [ library . . . ]
#include <secdb.h>
#include <user_attr.h>
#include <prof_attr.h>
#include <exec_attr.h>
#include <auth_attr.h>
```

DESCRIPTION

Functions in this library provide routines for manipulation of security attribute databases.

INTERFACES

The shared object `libsecdb.so.1` provides the public interfaces defined below. See `intro`(3) for additional information on shared object interfaces.

`chkauthattr`	`endauthattr`
`endexecattr`	`endprofattr`
`enduserattr`	`fgetuserattr`
`free_authattr`	`free_execattr`
`free_profattr`	`free_proflist`
`free_userattr`	`getauthattr`
`getauthnam`	`getexecattr`
`getexecprof`	`getexecuser`
`getprofattr`	`getproflist`
`getprofnam`	`getuserattr`
`getusernam`	`getuseruid`
`kva_match`	`match_execattr`
`setauthattr`	`setexecattr`
`setprofattr`	`setuserattr`

FILES

`/usr/lib/libsecdb.so.1`	shared object
`/usr/lib/64/libsecdb.so.1`	64-bit shared object

ATTRIBUTES

See `attributes`(5) for description of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsl (32-bit)

ATTRIBUTE TYPE	ATTRIBUTE VALUE
	SUNWcslx (64-bit)
MT Level	MT-Safe

SEE ALSO `intro`(3), `attributes`(5)

NAME

libsendfile – sendfile library

SYNOPSIS

```
cc [ flag... ] file... -lsendfile [ library... ]
#include <sys/sendfile.h>
```

DESCRIPTION

The functions in this library provide routines that enable files to be sent over sockets, buffers to be sent over sockets, files to be copied to files, and buffers to be copied to files.

INTERFACES

The shared object `libsendfile.so.1` provides the public interfaces defined below. See `intro`(3) for additional information on shared object interfaces.

```
sendfile                        sendfilev
```

The following interfaces are unique to the 32-bit version of this library:

```
sendfile64                      sendfilev64
```

FILES

`/usr/lib/libsendfile.so.1`	shared object
`/usr/lib/64/libsendfile.so.1`	64–bit shared object

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsl (32-bit) SUNWcslx (64-bit)
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO

`pvs`(1), `intro`(3), `sendfile`(3EXT), `sendfilev`(3EXT), `attributes`(5)

NAME

libslp – service location protocol library

SYNOPSIS

```
cc [ flag . . . ] file . . . -lslp [ library . . . ]
```

DESCRIPTION

Functions in this library provide routines that provide the Service Location Protocol C library.

INTERFACES

The shared object `libslp.so.1` provides the public interfaces defined below. See `intro`(3) for additional information on shared object interfaces.

`SLPClose`	`SLPDelAttrs`
`SLPDereg`	`SLPEscape`
`SLPFindAttrs`	`SLPFindScopes`
`SLPFindSrvTypes`	`SLPFindSrvs`
`SLPFree`	`SLPGetProperty`
`SLPGetRefreshInterval`	`SLPOpen`
`SLPParseSrvURL`	`SLPReg`
`SLPSetProperty`	`SLPUnescape`
`slp_strerror`	

FILES

`/usr/lib/libslp.so.1`
shared object

`/usr/lib/64/libslp.so.1`
64-bit shared object file

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWslpu

SEE ALSO

`pvs`(1), `intro`(2), `intro`(3), `attributes`(5)

NAME

libsocket – sockets library

SYNOPSIS

`cc [ flag . . . ] file . . . -lsocket [ library . . . ]`

DESCRIPTION

Functions in this library provide routines that provide the socket internetworking interface, primarily used with the TCP/IP protocol suite.

INTERFACES

The shared object `libsocket.so.1` provides the public interfaces defined below. See `intro`(3) for additional information on shared object interfaces.

`__xnet_bind`	`__xnet_connect`
`__xnet_getsockopt`	`__xnet_listen`
`__xnet_recvmsg`	`__xnet_sendmsg`
`__xnet_sendto`	`__xnet_socket`
`__xnet_socketpair`	`accept`
`bind`	`bindresvport`
`connect`	`endnetent`
`endprotoent`	`endservent`
`ether_aton`	`ether_hostton`
`ether_line`	`ether_ntoa`
`ether_ntohost`	`freeaddrinfo`
`gai_strerror`	`getaddrinfo`
`getnameinfo`	`getnetbyaddr`
`getnetbyaddr_r`	`getnetbyname`
`getnetbyname_r`	`getnetent`
`getnetent_r`	`getpeername`
`getprotobyname`	`getprotobyname_r`
`getprotobynumber`	`getprotobynumber_r`
`getprotoent`	`getprotoent_r`
`getservbyname`	`getservbyname_r`
`getservbyport`	`getservbyport_r`
`getservent`	`getservent_r`
`getsockname`	`getsockopt`
`htonl`	`htons`

`if_freenameindex`	`if_indextoname`
`if_nameindex`	`if_nametoindex`
`in6addr_any`	`in6addr_loopback`
`inet_lnaof`	`inet_makeaddr`
`inet_network`	`listen`
`ntohl`	`ntohs`
`rcmd`	`rcmd_af`
`recv`	`recvfrom`
`recvmsg`	`rexec`
`rexec_af`	`rresvport`
`rresvport_af`	`ruserok`
`send`	`sendmsg`
`sendto`	`setnetent`
`setprotoent`	`setservent`
`setsockopt`	`shutdown`
`socket`	`socketpair`

FILES

`/usr/lib/libsocket.a`
archive library

`/usr/lib/libsocket.so.1`
shared object

`/usr/lib/64/libsocket.so.1`
64-bit shared object

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsl, SUNWarc (32-bit) SUNWcslx (64-bit)
MT-Level	Safe

SEE ALSO

`pvs`(1), `intro`(2), `intro`(3), `attributes`(5)

NAME

libssagent – Sun Solstice Enterprise Agent library

SYNOPSIS

```
cc [ flag . . . ] file . . . -lssagent [ library . .  ]
```

DESCRIPTION

The `libssagent` library is a high level API library that is dependent on `libssasnmp`. This library contains the starting point of the request-driven engine that always runs in the background within the subagent. It receives SNMP requests, evaluates variables, calls the appropriate functions, and sends the correct responses.

INTERFACES

The shared object `libssagent.so.1` provides the public interfaces defined below. See `intro`(3) for additional information on shared object interfaces.

`SSAAgentIsAlive`	`SSAGetTrapPort`
`SSAMain`	`SSARegSubagent`
`SSARegSubtree`	`SSASubagentOpen`
`_SSASendTrap`	`_SSASendTrap2`
`_SSASendTrap3`	`callItem`
`numCallItem`	`numTrapElem`
`trapAnyEnterpriseInfo`	`trapBucket`
`trapEnterpriseInfo`	`trapTableMap`

FILES

`/usr/lib/libssagent.so.1`	shared object
`/usr/lib/64/libssagent.so.1`	64-bit shared object

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWsasnm
MT-Level	Unsafe

SEE ALSO

`intro`(3), `libssasnmp`(3LIB), `attributes`(5)

NAME

libssasnmp – Sun Solstice Enterprise SNMP library

SYNOPSIS

```
cc [ flag . . . ] file . . . -lssasnmp [ library . .  ]
```

DESCRIPTION

The `libssasnmp` library provides low-level SNMP API functions.

- ASN.1 serialization (encoding/decoding) module
- SNMP PDU development routines
- SNMP session module
- Low level SNMP based API functions
- Error-handling module
- Trace (debugging) module

INTERFACES

The shared object `libssasnmp.so.1` provides the public interfaces defined below. See `intro`(3) for additional information on shared object interfaces.

`SSAOidCmp`	`SSAOidCpy`
`SSAOidDup`	`SSAOidFree`
`SSAOidInit`	`SSAOidNew`
`SSAOidStrToOid`	`SSAOidString`
`SSAOidZero`	`SSAStringCpy`
`SSAStringInit`	`SSAStringToChar`
`SSAStringZero`	

FILES

`/usr/lib/libssasnmp.so.1`	shared object
`/usr/lib/64/libssasnmp.so.1`	64-bit shared object

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWsasnm
MT-Level	Unsafe

SEE ALSO

`intro`(3), `libssagent`(3LIB), `attributes`(5)

NAME

libsys – system library

SYNOPSIS

`cc [ flag . . . ] file . . . -lsys [ library . . . ]`

DESCRIPTION

Functions in this library provide basic system services. This library is implemented as a filter on the C library (see `libc`(3LIB)).

INTERFACES

The shared object `libsys.so.1` provides the public interfaces defined below. See `intro`(3) for additional information on shared object interfaces.

`__ctype`	`__huge_val`	`_access`
`_acct`	`_alarm`	`_altzone`
`_catclose`	`_catgets`	`_catopen`
`_chdir`	`_chmod`	`_chown`
`_chroot`	`_close`	`_closedir`
`_creat`	`_daylight`	`_dup`
`_environ`	`_execl`	`_execle`
`_execlp`	`_execv`	`_execve`
`_execvp`	`_exit`	`_fattach`
`_fchdir`	`_fchmod`	`_fchown`
`_fcntl`	`_fdetach`	`_fork`
`_fpathconf`	`_fstat`	`_fstatvfs`
`_fsync`	`_ftok`	`_getcontext`
`_getcwd`	`_getegid`	`_geteuid`
`_getgid`	`_getgrgid`	`_getgrnam`
`_getgroups`	`_getlogin`	`_getmsg`
`_getpgid`	`_getpgrp`	`_getpid`
`_getpmsg`	`_getppid`	`_getpwnam`
`_getpwuid`	`_getrlimit`	`_getsid`
`_gettxt`	`_getuid`	`_grantpt`
`_initgroups`	`_ioctl`	`_isastream`
`_kill`	`_lchown`	`_link`
`_lseek`	`_lstat`	`_makecontext`
`_memcntl`	`_mkdir`	`_mknod`

_mlock	_mmap	_mount
_mprotect	_msgctl	_msgget
_msgrcv	_msgsnd	_msync
_munlock	_munmap	_nice
_numeric	_open	_opendir
_pathconf	_pause	_pipe
_poll	_profil	_ptrace
_ptsname	_putmsg	_putpmsg
_read	_readdir	_readlink
_readv	_rename	_rewinddir
_rmdir	_seekdir	_semctl
_semget	_semop	_setcontext
_setgid	_setgroups	_setpgid
_setpgrp	_setrlimit	_setsid
_setuid	_shmat	_shmctl
_shmdt	_shmget	_sigaction
_sigaddset	_sigaltstack	_sigdelset
_sigemptyset	_sigfillset	_sighold
_sigignore	_sigismember	_siglongjmp
_sigpause	_sigpending	_sigprocmask
_sigrelse	_sigsend	_sigsendset
_sigset	_sigsetjmp	_sigsuspend
_stat	_statvfs	_stime
_swapcontext	_symlink	_sync
_sysconf	_telldir	_time
_times	_timezone	_ttyname
_tzname	_ulimit	_umask
_umount	_uname	_unlink
_unlockpt	_utime	_wait
_waitid	_waitpid	_write

_writev	access	acct
alarm	atexit	calloc
catclose	catgets	catopen
chdir	chmod	chown
chroot	close	closedir
creat	daylight	dup
environ	execl	execle
execlp	execv	execve
execvp	exit	fattach
fchdir	fchmod	fchown
fcntl	fdetach	fork
fpathconf	free	fstat
fstatvfs	fsync	ftok
getcontext	getcwd	getegid
geteuid	getgid	getgrgid
getgrnam	getgroups	getlogin
getmsg	getpgid	getpgrp
getpid	getpmsg	getppid
getpwnam	getpwuid	getrlimit
getsid	gettxt	getuid
grantpt	initgroups	ioctl
isastream	kill	lchown
link	localeconv	lseek
lstat	makecontext	malloc
memcntl	mkdir	mknod
mlock	mmap	mount
mprotect	msgctl	msgget
msgrcv	msgsnd	msync
munlock	munmap	nice
open	opendir	pathconf

pause	pipe	poll
profil	ptrace	ptsname
putmsg	putpmsg	read
readdir	readlink	readv
realloc	remove	rename
rewinddir	rmdir	seekdir
semctl	semget	semop
setcontext	setgid	setgroups
setlocale	setpgid	setpgrp
setrlimit	setsid	setuid
shmat	shmctl	shmdt
shmget	sigaction	sigaddset
sigaltstack	sigdelset	sigemptyset
sigfillset	sighold	sigignore
sigismember	siglongjmp	signal
sigpause	sigpending	sigprocmask
sigrelse	sigsend	sigsendset
sigset	sigsetjmp	sigsuspend
stat	statvfs	stime
strcoll	strerror	strftime
strxfrm	swapcontext	symlink
sync	sysconf	system
telldir	time	times
timezone	ttyname	tzname
ulimit	umask	umount
uname	unlink	unlockpt
utime	wait	waitid
waitpid	write	writev

The following interfaces are unique to the SPARC version of this library:

.div	.mul	.rem
.stret1	.stret2	.stret4
.stret8	.udiv	.umul
.urem	_Q_add	_Q_cmp
_Q_cmpe	_Q_div	_Q_dtoq
_Q_feq	_Q_fge	_Q_fgt
_Q_fle	_Q_flt	_Q_fne
_Q_itoq	_Q_mul	_Q_neg
_Q_qtod	_Q_qtoi	_Q_qtos
_Q_qtou	_Q_sqrt	_Q_stoq
_Q_sub	_Q_utoq	__dtou
__ftou		

The following interfaces are unique to the Intel version of this library:

__flt_rounds	_fp_hw	_fpstart
_fxstat	_lxstat	_nuname
_sbrk	_xmknod	_xstat
nuname	sbrk	

FILES

/usr/lib/libsys.so.1 shared object

ATTRIBUTES

See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsl
MT-Level	Safe

SEE ALSO

pvs(1), intro(2), intro(3), libc(3LIB), attributes(5)

NAME

libsysevent – system event interface library

SYNOPSIS

```
cc [ flag . . . ] file . . . -lsysevent [ library . . . ]
#include <sysevent.h>
```

DESCRIPTION

Functions in this library extract specific identifier, publisher, and attribute information from a system event (sysevent) handle, defined as `sysevent_t`, and allow priviledged user-level applications to queue system events for delivery to the system event daemon, `syseventd`(1M).

INTERFACES

The shared object `libsysevent.so.1` provides the public interfaces defined below. See `intro`(3) for additional information on shared object interfaces.

`sysevent_bind_handle`	`sysevent_free`
`sysevent_get_attr_list`	`sysevent_get_class_name`
`sysevent_get_pid`	`sysevent_get_pub_name`
`sysevent_get_seq`	`sysevent_get_size`
`sysevent_get_subclass_name`	`sysevent_get_time`
`sysevent_get_vendor_name`	`sysevent_post_event`
`sysevent_subscribe_event`	`sysevent_unbind_handle`
`sysevent_unsubscribe_event`	

FILES

`/usr/lib/libsysevent.so.1`
shared object

`/usr/lib/64/libsysevent.so.1`
64-bit shared object

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsl (32-bit) SUNWcslx (64-bit)
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO

`syseventd`(1M), `intro`(3), `attributes`(5)

NAME

libtermcap – terminal independent operation library

SYNOPSIS

```
cc [ flag . . . ] -I /usr/ucbinclude file . . . -L /usr/libucb \
-R /usr/libucb -ltermcap [ library . . . ]
```

DESCRIPTION

Functions in this library extract and use capabilities from the terminal capability database `terminfo`(4).

INTERFACES

The shared object `libtermcap.so.1` provides the public interfaces defined below. See `intro`(3) for additional information on shared object interfaces.

BC	PC	UP	ospeed	tgetent
tgetflag	tgetnum	tgetstr	tgoto	tputs

FILES

`/usr/ucblib/libtermcap.a`
archive library

`/usr/ucblib/libtermcap.so.1`
shared object

`/usr/ucblib/64/libtermcap.so.1`
64-bit shared object

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
MT-Level	Unsafe

SEE ALSO

`intro`(3), `curs_termcap`(3CURSES), `terminfo`(4), `attributes`(5)

NAME

libthread – threads library

SYNOPSIS

`cc -mt [` *flag* `. . . ]` *file* `. . . [` *library* `. . . ]`

DESCRIPTION

Functions in `libthread` provide routines that provide threading support.

INTERFACES

The shared object `libthread.so.1` provides the public interfaces defined below. See `intro`(3) for additional information on shared object interfaces.

`__pthread_cleanup_pop`	`__pthread_cleanup_push`
`__xpg4_putmsg`	`__xpg4_putpmsg`
`_getfp`	`_mutex_held`
`_rw_read_held`	`_rw_write_held`
`_sema_held`	`alarm`
`close`	`cond_broadcast`
`cond_destroy`	`cond_init`
`cond_reltimedwait`	`cond_signal`
`cond_timedwait`	`cond_wait`
`creat`	`fcntl`
`fork`	`fork1`
`fsync`	`getmsg`
`getpmsg`	`kill`
`lockf`	`lwp_self`
`msgrcv`	`msgsnd`
`msync`	`mutex_destroy`
`mutex_init`	`mutex_lock`
`mutex_trylock`	`mutex_unlock`
`open`	`pause`
`poll`	`pread`
`pthread_attr_destroy`	`pthread_attr_getdetachstate`
`pthread_attr_getguardsize`	`pthread_attr_getinheritsched`
`pthread_attr_getschedparam`	`pthread_attr_getschedpolicy`
`pthread_attr_getscope`	`pthread_attr_getstackaddr`

pthread_attr_getstacksize	pthread_attr_init
pthread_attr_setdetachstate	pthread_attr_setguardsize
pthread_attr_setinheritsched	pthread_attr_setschedparam
pthread_attr_setschedpolicy	pthread_attr_setscope
pthread_attr_setstackaddr	pthread_attr_setstacksize
pthread_cancel	pthread_cond_broadcast
pthread_cond_destroy	pthread_cond_init
pthread_cond_reltimedwait_np	pthread_cond_signal
pthread_cond_timedwait	pthread_cond_wait
pthread_condattr_destroy	pthread_condattr_getpshared
pthread_condattr_init	pthread_condattr_setpshared
pthread_create	pthread_detach
pthread_equal	pthread_exit
pthread_getconcurrency	pthread_getschedparam
pthread_getspecific	pthread_join
pthread_key_create	pthread_key_delete
pthread_kill	pthread_mutex_consistent_np
pthread_mutex_destroy	pthread_mutex_getprioceiling
pthread_mutex_init	pthread_mutex_lock
pthread_mutex_setprioceiling	pthread_mutex_trylock
pthread_mutex_unlock	pthread_mutexattr_destroy
pthread_mutexattr_getprioceiling	pthread_mutexattr_getprotocol
pthread_mutexattr_getpshared	pthread_mutexattr_getrobust_np
pthread_mutexattr_gettype	pthread_mutexattr_init
pthread_mutexattr_setprioceiling	pthread_mutexattr_setprotocol
pthread_mutexattr_setpshared	pthread_mutexattr_setrobust_np
pthread_mutexattr_settype	pthread_once
pthread_rwlock_destroy	pthread_rwlock_init
pthread_rwlock_rdlock	pthread_rwlock_tryrdlock
pthread_rwlock_trywrlock	pthread_rwlock_unlock

pthread_rwlock_wrlock	pthread_rwlockattr_destroy
pthread_rwlockattr_getpshared	pthread_rwlockattr_init
pthread_rwlockattr_setpshared	pthread_self
pthread_setcancelstate	pthread_setcanceltype
pthread_setconcurrency	pthread_setschedparam
pthread_setspecific	pthread_sigmask
pthread_testcancel	putmsg
putpmsg	pwrite
read	readv
rw_rdlock	rw_tryrdlock
rw_trywrlock	rw_unlock
rw_wrlock	rwlock_destroy
rwlock_init	select
sema_destroy	sema_init
sema_post	sema_trywait
sema_wait	setcontext
setitimer	sigaction
siglongjmp	sigpause
sigpending	sigprocmask
sigsetjmp	sigsuspend
sigwait	sleep
tcdrain	thr_continue
thr_create	thr_exit
thr_getconcurrency	thr_getprio
thr_getspecific	thr_join
thr_keycreate	thr_kill
thr_main	thr_min_stack
thr_self	thr_setconcurrency
thr_setprio	thr_setspecific
thr_sigsetmask	thr_stksegment

thr_suspend	thr_yield
usleep	wait
wait3	waitid
waitpid	write
writev	

The following interfaces are unique to the 32-bit version of this library:

creat64	lockf64
open64	pread64
pwrite64	

FILES

/usr/lib/libthread.so.1
shared object

/usr/lib/64/libthread.so.1
64-bit shared object

ATTRIBUTES

See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsl (32-bit) SUNWcslx (64-bit)
MT-Level	Safe

SEE ALSO

pvs(1), intro(2), intro(3), libpthread(3LIB), libthread_db(3LIB), libthread_db(3THR), threads(3THR), attributes(5)

NAME

libthread_db – threads debugging library

SYNOPSIS

```
cc [ flag . . . ] file . . . -lthread_db [ library . . . ]
#include <proc_service.h>
#include <thread_db.h>
```

DESCRIPTION

Functions is this library are used to build debuggers for multithreaded programs.

INTERFACES

The shared object `libthread_db.so.1` provides the public interfaces defined below. See `intro`(3) for additional information on shared object interfaces.

`td_init`	`td_log`
`td_sync_get_info`	`td_sync_get_stats`
`td_sync_setstate`	`td_sync_waiters`
`td_ta_clear_event`	`td_ta_delete`
`td_ta_enable_stats`	`td_ta_event_addr`
`td_ta_event_getmsg`	`td_ta_get_nthreads`
`td_ta_get_ph`	`td_ta_get_stats`
`td_ta_map_addr2sync`	`td_ta_map_id2thr`
`td_ta_map_lwp2thr`	`td_ta_new`
`td_ta_reset_stats`	`td_ta_set_event`
`td_ta_setconcurrency`	`td_ta_sync_iter`
`td_ta_sync_tracking_enable`	`td_ta_thr_iter`
`td_ta_tsd_iter`	`td_thr_clear_event`
`td_thr_dbresume`	`td_thr_dbsuspend`
`td_thr_event_enable`	`td_thr_event_getmsg`
`td_thr_get_info`	`td_thr_getfpregs`
`td_thr_getgregs`	`td_thr_getxregs`
`td_thr_getxregsize`	`td_thr_lockowner`
`td_thr_set_event`	`td_thr_setfpregs`
`td_thr_setgregs`	`td_thr_setprio`
`td_thr_setsigpending`	`td_thr_setxregs`
`td_thr_sigsetmask`	`td_thr_sleepinfo`
`td_thr_tsd`	`td_thr_validate`

FILES

`/usr/lib/libthread_db.so.1`
shared object

`/usr/lib/64/libthread_db.so.1`
64-bit shared object

ATTRIBUTES

See `attributes`(5) for description of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsl (32-bit) SUNWcslx (64-bit)
MT Level	Safe

SEE ALSO

`pvs`(1), `intro`(3), `libpthread`(3LIB), `libthread`(3LIB), `libthread_db`(3THR), `threads`(3THR), `attributes`(5)

NAME

libtnfctl – TNF probe control library

SYNOPSIS

```
cc [ flag . . . ] file . . . -ltnfctl [ library . . . ]
#include <tnf/tnfctl.h>
```

DESCRIPTION

Functions in this library provide TNF probe control routines for use by processes and the kernel.

INTERFACES

The shared object `libtnfctl.so.1` provides the public interfaces defined below. See `intro`(3) for additional information on shared object interfaces.

`tnfctl_buffer_alloc`	`tnfctl_buffer_dealloc`
`tnfctl_check_libs`	`tnfctl_close`
`tnfctl_continue`	`tnfctl_exec_open`
`tnfctl_filter_list_add`	`tnfctl_filter_list_delete`
`tnfctl_filter_list_get`	`tnfctl_filter_state_set`
`tnfctl_indirect_open`	`tnfctl_internal_open`
`tnfctl_kernel_open`	`tnfctl_pid_open`
`tnfctl_probe_apply`	`tnfctl_probe_apply_ids`
`tnfctl_probe_connect`	`tnfctl_probe_disable`
`tnfctl_probe_disconnect_all`	`tnfctl_probe_enable`
`tnfctl_probe_state_get`	`tnfctl_probe_trace`
`tnfctl_probe_untrace`	`tnfctl_register_funcs`
`tnfctl_strerror`	`tnfctl_trace_attrs_get`
`tnfctl_trace_state_set`	

FILES

`/usr/lib/libtnfctl.so.1`
shared object

`/usr/lib/64/libtnfctl.so.1`
64-bit shared object

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWtnfc (32-bit) SUNWtnfcx (64-bit)

ATTRIBUTE TYPE	ATTRIBUTE VALUE
MT Level	MT-Safe with exceptions

SEE ALSO `pvs`(1), `intro`(3), `libtnfctl`(3TNF), `tracing`(3TNF), `attributes`(5)

NOTES This API is MT-Safe. Multiple threads can concurrently operate on independent `tnfctl` handles, which is the typical behavior expected. `libtnfctl` does not support multiple threads operating on the same `tnfctl` handle. If this is desired, it is the client's responsibility to implement locking to ensure that two threads that use the same `tnfctl` handle are not simultaneously present in a `libtnfctl` interface.

NAME

libucb – UCB source compatibility library

SYNOPSIS

```
cc [ flag . . . ] -I /usr/ucbinclude file . . . -L /usr/libucb \
-R /usr/libucb -lucb [ library . . . ]
```

DESCRIPTION

Functions in this library provide UCB source compatibility.

INTERFACES

The shared object `libucb.so.1` provides the public interfaces defined below. See `intro`(3) for additional information on shared object interfaces.

alphasort	bcmp	bcopy
bzero	flock	fopen
fprintf	freopen	fstatfs
ftime	getdtablesize	gethostid
gethostname	getpagesize	getrusage
gettimeofday	getwd	index
killpg	longjmp	mctl
nice	nlist	printf
psignal	rand	re_comp
re_exec	readdir	reboot
rindex	scandir	setbuffer
sethostname	setjmp	setlinebuf
setpgrp	settimeofday	sigblock
siginterrupt	signal	sigpause
sigsetmask	sigstack	sigvec
sigvechandler	sleep	sprintf
srand	statfs	sys_siglist
times	ualarm	usignal
usigpause	usleep	vfprintf
vprintf	vsprintf	wait3
wait4		

The following interfaces are unique to the 32-bit version of this library:

`alphasort64`	`fopen64`	`freopen64`
`readdir64`	`scandir64`	

FILES

`/usr/ucblib/libucb.a`
archive library

`/usr/ucblib/libucb.so.1`
shared object

`/usr/ucblib/64/libucb.so.1`
64-bit shared object

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWscpu, SUNWsra (32-bit) SUNWscpux (64-bit)
MT-Level	Safe with exceptions

SEE ALSO

`pvs`(1), `intro`(3), `attributes`(5)

NAME

libvolmgt – volume management library

SYNOPSIS

```
cc [ flag . . . ] file . . . -lvolmgt [ library . . . ]
#include <volmgt.h>
```

DESCRIPTION

Functions in this library provide access to the volume management services.

INTERFACES

The shared object `libvolmgt.so.1` provides the public interfaces defined below. See `intro`(3) for additional information on shared object interfaces.

`media_findname`	`media_getattr`
`media_getid`	`media_setattr`
`volmgt_acquire`	`volmgt_check`
`volmgt_feature_enabled`	`volmgt_inuse`
`volmgt_ownspath`	`volmgt_release`
`volmgt_root`	`volmgt_running`
`volmgt_symdev`	`volmgt_symname`

FILES

`/usr/lib/libvolmgt.a`
archive library

`/usr/lib/libvolmgt.so.1`
shared object

`/usr/lib/64/libvolmgt.so.1`
64-bit shared object

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsl, SUNWarc (32-bit) SUNWcslx (64-bit)
MT-Level	Safe with exceptions

SEE ALSO

`pvs`(1), `intro`(3), `media_findname`(3VOLMGT), `attributes`(5)

NOTES

The MT-Level for this library of interfaces is Safe, except for `media_findname`(3VOLMGT), which is Unsafe.

NAME

libw – wide character library

SYNOPSIS

```
cc [ flag . . . ] file . . .  [ library . . . ]
#include <wchar.h>
```

DESCRIPTION

Historically, functions in this library provided wide character translations. This functionality now resides in `libc`(3LIB).

This library is maintained to provide backward compatibility for both runtime and compilation environments. The shared object version is implemented as a filter on `libw.so.1`, and the archive version is implemented as a null archive. New application development need not reference either version of `libw`.

INTERFACES

The shared object `libw.so.1` provides the public interfaces defined below. See `intro`(3) for additional information on shared object interfaces.

fgetwc	fgetws	fputwc
fputws	getwc	getwchar
getws	isenglish	isideogram
isnumber	isphonogram	isspecial
iswalnum	iswalpha	iswcntrl
iswctype	iswdigit	iswgraph
iswlower	iswprint	iswpunct
iswspace	iswupper	iswxdigit
putwc	putwchar	putws
strtows	towlower	towupper
ungetwc	watoll	wcscat
wcschr	wcscmp	wcscoll
wcscpy	wcscspn	wcsftime
wcslen	wcsncat	wcsncmp
wcsncpy	wcspbrk	wcsrchr
wcsspn	wcstod	wcstok
wcstol	wcstoul	wcswcs
wcswidth	wcsxfrm	wctype
wcwidth	wscasecmp	wscat
wschr	wscmp	wscol

wscoll	wscpy	wscspn
wsdup	wslen	wsncasecmp
wsncat	wsncmp	wsncpy
wspbrk	wsprintf	wsrchr
wsscanf	wsspn	wstod
wstok	wstol	wstoll
wstostr	wsxfrm	

FILES

/usr/lib/libw.a	a link to /usr/lib/null.a
/usr/lib/libw.so.1	a filter on libc.so.1
/usr/lib/64/libw.so.1	a filter on 64/libc.so.1

ATTRIBUTES

See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsl, SUNWarc (32-bit) SUNWcslx (64-bit)
MT-Level	Safe

SEE ALSO

pvs(1), intro(3), libc(3LIB), attributes(5)

NAME

libwsreg – product install registry library

SYNOPSIS

```
cc [ flag . . . ] file . . . -lwsreg [ library . . . ]
#include <wsreg.h>
```

DESCRIPTION

Functions in this library provide access to the product install registry.

INTERFACES

The shared object `libwsreg.so.1` provides the public interfaces defined below. See `intro`(3) for additional information on shared object interfaces.

wsreg_add_child_component	wsreg_add_compatible_version
wsreg_add_dependent_component	wsreg_add_display_name
wsreg_add_required_component	wsreg_can_access_registry
wsreg_clone_component	wsreg_components_equal
wsreg_create_component	wsreg_free_component
wsreg_free_component_array	wsreg_get
wsreg_get_all	wsreg_get_child_components
wsreg_get_compatible_versions	wsreg_get_data
wsreg_get_data_pairs	wsreg_get_dependent_components
wsreg_get_display_languages	wsreg_get_display_name
wsreg_get_id	wsreg_get_instance
wsreg_get_location	wsreg_get_parent
wsreg_get_required_components	wsreg_get_type
wsreg_get_uninstaller	wsreg_get_unique_name
wsreg_get_vendor	wsreg_get_version
wsreg_initialize	wsreg_query_create
wsreg_query_free	wsreg_query_get_id
wsreg_query_get_instance	wsreg_query_get_location
wsreg_query_get_unique_name	wsreg_query_get_version
wsreg_query_set_id	wsreg_query_set_instance
wsreg_query_set_location	wsreg_query_set_unique_name
wsreg_query_set_version	wsreg_register
wsreg_remove_child_component	wsreg_remove_compatible_version
wsreg_remove_dependent_component	wsreg_remove_display_name

wsreg_remove_required_component	wsreg_set_data
wsreg_set_id	wsreg_set_instance
wsreg_set_location	wsreg_set_parent
wsreg_set_type	wsreg_set_uninstaller
wsreg_set_unique_name	wsreg_set_vendor
wsreg_set_version	wsreg_unregister

FILES

`/usr/lib/libwsreg.so.1`
shared object

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWwsr2
MT-Level	Unsafe

SEE ALSO

`prodreg`(1M), `intro`(3), `attributes`(5)

NAME

libxfn – X/Open Federated Naming (XFN) library

SYNOPSIS

```
cc [ flag . . . ] file . . . -lxfn [ library . . . ]
#include <xfn/xfn.h>
```

DESCRIPTION

Functions in this library provide the implementation of XFN, the X/Open Federated Naming specification (see `xfn`(3XFN) and `fns`(5)).

INTERFACES

The shared object `libxfn.so.2` provides the public interfaces defined below. See `intro`(3) for additional information on shared object interfaces.

`_pure_error_`	`fn_attr_bind`
`fn_attr_create_subcontext`	`fn_attr_get`
`fn_attr_get_ids`	`fn_attr_get_values`
`fn_attr_modify`	`fn_attr_multi_get`
`fn_attr_multi_modify`	`fn_attribute_add`
`fn_attribute_assign`	`fn_attribute_copy`
`fn_attribute_create`	`fn_attribute_destroy`
`fn_attribute_first`	`fn_attribute_identifier`
`fn_attribute_next`	`fn_attribute_remove`
`fn_attribute_syntax`	`fn_attribute_valuecount`
`fn_attrmodlist_add`	`fn_attrmodlist_assign`
`fn_attrmodlist_copy`	`fn_attrmodlist_count`
`fn_attrmodlist_create`	`fn_attrmodlist_destroy`
`fn_attrmodlist_first`	`fn_attrmodlist_next`
`fn_attrset_add`	`fn_attrset_assign`
`fn_attrset_copy`	`fn_attrset_count`
`fn_attrset_create`	`fn_attrset_destroy`
`fn_attrset_first`	`fn_attrset_get`
`fn_attrset_next`	`fn_attrset_remove`
`fn_bindinglist_destroy`	`fn_bindinglist_next`
`fn_composite_name_append_comp`	`fn_composite_name_append_name`
`fn_composite_name_assign`	`fn_composite_name_copy`
`fn_composite_name_count`	`fn_composite_name_create`

fn_composite_name_delete_comp	fn_composite_name_destroy
fn_composite_name_first	fn_composite_name_from_str
fn_composite_name_from_string	fn_composite_name_insert_comp
fn_composite_name_insert_name	fn_composite_name_is_empty
fn_composite_name_is_equal	fn_composite_name_is_prefix
fn_composite_name_is_suffix	fn_composite_name_last
fn_composite_name_next	fn_composite_name_prefix
fn_composite_name_prepend_comp	fn_composite_name_prepend_name
fn_composite_name_prev	fn_composite_name_suffix
fn_compound_name_append_comp	fn_compound_name_assign
fn_compound_name_copy	fn_compound_name_count
fn_compound_name_delete_all	fn_compound_name_delete_comp
fn_compound_name_destroy	fn_compound_name_first
fn_compound_name_from_syntax_ attrs	fn_compound_name_get_syntax_attrs
fn_compound_name_insert_comp	fn_compound_name_is_empty
fn_compound_name_is_equal	fn_compound_name_is_prefix
fn_compound_name_is_suffix	fn_compound_name_last
fn_compound_name_next	fn_compound_name_prefix
fn_compound_name_prepend_comp	fn_compound_name_prev
fn_compound_name_suffix	fn_ctx_bind
fn_ctx_create_subcontext	fn_ctx_destroy_subcontext
fn_ctx_get_ref	fn_ctx_get_syntax_attrs
fn_ctx_handle_destroy	fn_ctx_handle_from_initial
fn_ctx_handle_from_ref	fn_ctx_list_bindings
fn_ctx_list_names	fn_ctx_lookup
fn_ctx_lookup_link	fn_ctx_rename
fn_ctx_unbind	fn_multigetlist_destroy
fn_multigetlist_next	fn_namelist_destroy
fn_namelist_next	fn_ref_addr_assign
fn_ref_addr_copy	fn_ref_addr_create

fn_ref_addr_data	fn_ref_addr_description
fn_ref_addr_destroy	fn_ref_addr_length
fn_ref_addr_type	fn_ref_addrcount
fn_ref_append_addr	fn_ref_assign
fn_ref_copy	fn_ref_create
fn_ref_create_link	fn_ref_delete_addr
fn_ref_delete_all	fn_ref_description
fn_ref_destroy	fn_ref_first
fn_ref_insert_addr	fn_ref_is_link
fn_ref_link_name	fn_ref_next
fn_ref_prepend_addr	fn_ref_type
fn_status_advance_by_name	fn_status_append_remaining_name
fn_status_append_resolved_name	fn_status_assign
fn_status_code	fn_status_copy
fn_status_create	fn_status_description
fn_status_destroy	fn_status_diagnostic_message
fn_status_is_success	fn_status_link_code
fn_status_link_diagnostic_message	fn_status_link_remaining_name
fn_status_link_resolved_name	fn_status_link_resolved_ref
fn_status_remaining_name	fn_status_resolved_name
fn_status_resolved_ref	fn_status_set
fn_status_set_code	fn_status_set_diagnostic_message
fn_status_set_link_code	fn_status_set_link_diagnostic_message
fn_status_set_link_remaining_name	fn_status_set_link_resolved_name
fn_status_set_link_resolved_ref	fn_status_set_remaining_name
fn_status_set_resolved_name	fn_status_set_resolved_ref
fn_status_set_success	fn_string_assign
fn_string_bytecount	fn_string_charcount
fn_string_code_set	fn_string_compare
fn_string_compare_substring	fn_string_contents

fn_string_copy	fn_string_create
fn_string_destroy	fn_string_from_composite_name
fn_string_from_compound_name	fn_string_from_contents
fn_string_from_str	fn_string_from_str_n
fn_string_from_strings	fn_string_from_substring
fn_string_is_empty	fn_string_lang_terr
fn_string_next_substring	fn_string_prev_substring
fn_string_str	fn_valuelist_destroy
fn_valuelist_next	prelim_fn_attr_ext_search
prelim_fn_attr_search	prelim_fn_ctx_equivalent_name
prelim_fn_ext_searchlist_destroy	prelim_fn_ext_searchlist_next
prelim_fn_search_control_assign	prelim_fn_search_control_copy
prelim_fn_search_control_create	prelim_fn_search_control_destroy
prelim_fn_search_control_follow_links	prelim_fn_search_control_max_names
prelim_fn_search_control_return_attr_ids	prelim_fn_search_control_return_ref
prelim_fn_search_control_scope	prelim_fn_search_filter_arguments
prelim_fn_search_filter_assign	prelim_fn_search_filter_copy
prelim_fn_search_filter_create	prelim_fn_search_filter_destroy
prelim_fn_search_filter_expression	prelim_fn_searchlist_destroy
prelim_fn_searchlist_next	

FILES

/usr/lib/libxfn.so.2	shared object
/usr/lib/64/libxfn.so.2	64-bit shared object
/usr/lib/libxfn.so.1	shared object for backward compatibility only
/usr/lib/64/libxfn.so.1	64-bit shared object for backward compatibility only

ATTRIBUTES

See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWfns (32-bit) SUNWfnsx (64-bit)
MT-Level	Safe

SEE ALSO pvs(1), intro(3), xfn(3XFN), attributes(5), fns(5)

NAME

libxnet – X/Open Networking library

SYNOPSIS

```
cc [ flag . . . ] file . . . -lxnet [ library . . . ]
```

DESCRIPTION

Functions in this library provide networking interfaces which comply with the X/Open CAE Specification, Networking Services, Issue 4.

INTERFACES

The shared object `libxnet.so.1` provides the public interfaces defined below. See `intro`(3) for additional information on shared object interfaces.

`__t_errno`	`__xnet_bind`
`__xnet_connect`	`__xnet_getsockopt`
`__xnet_listen`	`__xnet_recvmsg`
`__xnet_sendmsg`	`__xnet_sendto`
`__xnet_socket`	`__xnet_socketpair`
`_xti_accept`	`_xti_alloc`
`_xti_bind`	`_xti_close`
`_xti_connect`	`_xti_error`
`_xti_free`	`_xti_getinfo`
`_xti_getprotaddr`	`_xti_getstate`
`_xti_listen`	`_xti_look`
`_xti_open`	`_xti_optmgmt`
`_xti_rcv`	`_xti_rcvconnect`
`_xti_rcvdis`	`_xti_rcvrel`
`_xti_rcvreldata`	`_xti_rcvudata`
`_xti_rcvuderr`	`_xti_rcvv`
`_xti_rcvvudata`	`_xti_snd`
`_xti_snddis`	`_xti_sndrel`
`_xti_sndreldata`	`_xti_sndudata`
`_xti_sndv`	`_xti_sndvudata`
`_xti_strerror`	`_xti_sync`
`_xti_sysconf`	`_xti_unbind`
`_xti_xns5_accept`	`_xti_xns5_snd`
`accept`	`bind`

connect	endhostent
endnetent	endprotoent
endservent	gethostbyaddr
gethostbyname	gethostent
gethostname	getnetbyaddr
getnetbyname	getnetent
getpeername	getprotobyname
getprotobynumber	getprotoent
getservbyname	getservbyport
getservent	getsockname
getsockopt	h_errno
htonl	htons
inet_addr	inet_lnaof
inet_makeaddr	inet_netof
inet_network	inet_ntoa
listen	ntohl
ntohs	recv
recvfrom	recvmsg
send	sendmsg
sendto	sethostent
setnetent	setprotoent
setservent	setsockopt
shutdown	socket
socketpair	t_errno

FILES

/usr/lib/libxnet.so.1	shared object
/usr/lib/64/libxnet.so.1	64-bit shared object

ATTRIBUTES

See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsl (32-bit) SUNWcslx (64-bit)
MT-Level	Safe

SEE ALSO `intro`(3), `attributes`(5), `standards`(5)

NAME

liby – yacc library

SYNOPSIS

```
cc [ flag . . . ] file . . . -ly [ library . . . ]
```

DESCRIPTION

The function in this library provides a user interface to the `yacc`(1) library.

INTERFACES

The shared object `liby.so.1` provides the public interface defined below. See `intro`(3) for additional information on shared object interfaces.

```
yyerror
```

FILES

`/usr/lib/liby.a`	archive library
`/usr/lib/liby.so.1`	shared object
`/usr/lib/sparcv9/liby.so.1`	64-bit shared object

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsl, SUNWbtool (32-bit) SUNWcslx (64-bit)
MT-Level	Unsafe

SEE ALSO

`yacc`(1), `intro`(3), `attributes`(5)

NAME

math – math functions and constants

SYNOPSIS

```
#include <math.h>
```

DESCRIPTION

This file contains declarations of all the functions in the Math Library (described in Section 3M), as well as various functions in the C Library (Section 3C) that return floating-point values.

It defines the structure and constants used by the `matherr`(3M) error-handling mechanisms, including the following constant used as a error-return value:

`HUGE`	The maximum value of a single-precision floating-point number.

The following mathematical constants are defined for user convenience:

`M_E`	The base of natural logarithms (*e*).
`M_LOG2E`	The base-2 logarithm of *e*.
`M_LOG10E`	The base-10 logarithm of *e*.
`M_LN2`	The natural logarithm of 2.
`M_LN10`	The natural logarithm of 10.
`M_PI`	`pi`, the ratio of the circumference of a circle to its diameter.
`M_PI_2`	`pi/2`.
`M_PI_4`	`pi/4`.
`M_1_PI`	`1/pi`.
`M_2_PI`	`2/pi`.
`M_2_SQRTPI`	2 over the square root of `pi`.
`M_SQRT2`	The positive square root of 2.
`M_SQRT1_2`	The positive square root of `1/2`.

The following mathematical constants are also defined in this header file:

`MAXFLOAT`	The maximum value of a non-infinite single-precision floating point number.
`HUGE_VAL`	positive infinity. For the definitions of various machine-dependent constants see `values`(3HEAD).

SEE ALSO

`intro`(3), `matherr`(3M), `values`(3HEAD)

NAME

mqueue – message queues

SYNOPSIS

```
#include <mqueue.h>
```

DESCRIPTION

The <mqueue.h> header defines the mqd_t type, which is used for message queue descriptors. This will not be an array type. A message queue descriptor may be implemented using a file descriptor, in which case applications can open up to at least OPEN_MAX file and message queues.

The <mqueue.h> header defines the sigevent structure (as described in <signal.h>, see signal(3HEAD)) and the mq_attr structure, which is used in getting and setting the attributes of a message queue. Attributes are initially set when the message queue is created. A mq_attr structure has the following members:

```
long    mq_flags       message queue flags
long    mq_maxmsg      maximum number of messages
long    mq_msgsize     maximum message size
long    mq_curmsgs     number of messages currently queued
```

Inclusion of the <mqueue.h> header may make visible symbols defined in the headers <fcntl.h>, <signal.h>, <sys/types.h>, and <time.h>.

SEE ALSO

fcntl(3HEAD), signal(3HEAD), time(3HEAD), types(3HEAD)

NAME

ndbm – definitions for ndbm database operations

SYNOPSIS

```
#include <ndbm.h>
```

DESCRIPTION

The <ndbm.h> header defines the datum type as a structure that includes at least the following members:

```
void   *dptr     /* pointer to the application's data */
size_t dsize     /* size of the object pointed to by dptr */
```

The size_t type is defined through typedef as described in <stddef.h>.

The <ndbm.h> header defines the DBM type through typedef.

The following constants are defined as possible values for the *store_mode* argument to dbm_store():

DBM_INSERT Insertion of new entries only.

DBM_REPLACE Allow replacing existing entries.

SEE ALSO

ndbm(3C), standards(5)

NAME

netdb – definitions for network database operations

SYNOPSIS

```
#include <netdb.h>
```

DESCRIPTION

The <netdb.h> header defines the type in_port_t and the type in_addr_t as described in in(3HEAD).

The <netdb.h> header defines the hostent structure that includes the following members:

char	*h_name	Official name of the host.
char	**h_aliases	A pointer to an array of pointers to alternative host names, terminated by a null pointer.
int	h_addrtype	Address type.
int	h_length	The length, in bytes, of the address.
char	**h_addr_list	A pointer to an array of pointers to network addresses (in network byte order) for the host, terminated by a null pointer.

The <netdb.h> header defines the netent structure that includes the following members:

char	*n_name	Official, fully-qualified (including the domain) name of the network.
char	**n_aliases	A pointer to an array of pointers to alternative network names, terminated by a null pointer.
int	n_addrtype	The address type of the network.
in_addr_t	n_net	The network number, in host byte order.

The <netdb.h> header defines the protoent structure that includes the following members:

char	*p_name	Official name of the protocol.
char	**p_aliases	A pointer to an array of pointers to alternative protocol names, terminated by a null pointer.
int	p_proto	The protocol number.

The <netdb.h> header defines the servent structure that includes the following members:

`char`	`*s_name`	Official name of the service.
`char`	`**s_aliases`	A pointer to an array of pointers to alternative service names, terminated by a null pointer.
`int`	`s_port`	The port number at which the service resides, in network byte order.
`char`	`*s_proto`	The name of the protocol to use when contacting the service.

The `<netdb.h>` header defines the macro `IPPORT_RESERVED` with the value of the highest reserved Internet port number.

The `<netdb.h>` header provides a declaration for `h_errno`:

extern int h_errno;

The `<netdb.h>` header defines the following macros for use as error values for `gethostbyaddr()` and `gethostbyname()`:

`HOST_NOT_FOUND`	`NO_DATA`
`NO_RECOVERY`	`TRY_AGAIN`

Inclusion of the `<netdb.h>` header may also make visible all symbols from `in`(3HEAD).

Default For applications that do not require standard-conforming behavior (those that use the socket interfaces described in section 3N of the reference manual; see `Intro`(3) and `standards`(5)), the following are declared as functions and can also be defined as macros:

```
int                 endhostent(void);
int                 endnetent(void);
int                 endprotoent(void);
int                 endservent(void);
struct hostent      *gethostbyaddr(const void *addr, int len, int type);
struct hostent      *gethostbyname(const char *name);
struct hostent      *gethostent(void);
struct netent       *getnetbyaddr(long net, int type);
```

```
struct netent         *getnetbyname(const char *name);
struct netent         *getnetent(void);
struct protoent       *getprotobyname(const char *name);
struct protoent       *getprotobynumber(int proto);
struct protoent       *getprotoent(void);
struct servent        *getservbyname(const char *name, const char *proto);
struct servent        *getservbyport(int port, const char *proto);
struct servent        *getservent(void);
int                   sethostent(int stayopen);
int                   setnetent(int stayopen);
int                   setprotoent(int stayopen);
int                   setservent(int stayopen);
```

Standard conforming

For applications that require standard-conforming behavior (those that use the socket interfaces described in section 3XN of the reference manual; see `Intro`(3) and `standards`(5)), the following are declared as functions and can also be defined as macros:

```
void                  endhostent(void);
void                  endnetent(void);
void                  endprotoent(void);
void                  endservent(void);
struct hostent        *gethostbyaddr(const void *addr, size_t len, int type);
struct hostent        *gethostbyname(const char *name);
struct hostent        *gethostent(void);
struct netent         *getnetbyaddr(in_addr_t net, int type);
struct netent         *getnetbyname(const char *name);
struct netent         *getnetent(void);
struct protoent       *getprotobyname(const char *name);
struct protoent       *getprotobynumber(int proto);
struct protoent       *getprotoent(void);
struct servent        *getservbyname(const char *name, const char *proto);
```

```
struct servent          *getservbyport(int port, const char *proto);
struct servent          *getservent(void);
void                    sethostent(int stayopen);
void                    setnetent(int stayopen);
void                    setprotoent(int stayopen);
void                    setservent(int stayopen);
```

SEE ALSO Intro(3), endhostent(3NSL), endhostent(3XNET), endnetent(3SOCKET), endnetent(3XNET), endprotoent(3SOCKET), endprotoent(3XNET), endservent(3SOCKET), endservent(3XNET), in(3HEAD), standards(5)

NAME

nl_types – native language data types

SYNOPSIS

```
#include <nl_types.h>
```

DESCRIPTION

This header contains the following definitions:

`nl_catd`	Used by the message catalog functions `catopen`, `catgets` and `catclose` to identify a catalog.
`nl_item`	Used by `nl_langinfo` to identify items of langinfo data. Values for objects of type `nl_item` are defined in `<langinfo.h>`.
`NL_SETD`	Used by `gencat` when no `$set` directive is specified in a message text source file. This constant can be used in subsequent calls to `catgets` as the value of the set identifier parameter.
`NL_MGSMAX`	Maximum number of messages per set.
`NL_SETMAX`	Maximum number of sets per catalog.
`NL_TEXTMAX`	Maximum size of a message.

SEE ALSO

`gencat`(1), `catgets`(3C), `catopen`(3C), `nl_langinfo`(3C), `langinfo`(3HEAD)

NAME

sched – execution scheduling

SYNOPSIS

```
#include <sched.h>
```

DESCRIPTION

The <sched.h> header defines the sched_param structure, which contains the scheduling parameters required for implementation of each supported scheduling policy. This structure contains the following member:

```
int    sched_priority    process execution scheduling priority
```

Each process is controlled by an associated scheduling policy and priority. Associated with each policy is a priority range. Each policy definition specifies the minimum priority range for that policy. The priority ranges for each policy may overlap the priority ranges of other policies.

The scheduling policies are indicated by the values of the following symbolic constants:

SCHED_FIFO	Processes are scheduled according to the First-In-First-Out (FIFO) policy. Processes scheduled to this policy, if not pre-empted by a higher priority or interrupted by a signal, will proceed until completion.
SCHED_RR	Processes are scheduled according to the Round-Robin (RR) policy. Processes scheduled to this policy, if not pre-empted by a higher priority or interrupted by a signal, will execute for a time period, returned by sched_rr_get_interval(3RT) or by the system.
SCHED_IA	Processes are scheduled according to the Inter-Active Class (IA) policy as described in priocntl(2).
SCHED_OTHER	Processes are scheduled according to another policy not described above.

The values of these constants are distinct.

Inclusion of the <sched.h> header will make visible symbols defined in the header <time.h>.

SEE ALSO

priocntl(2), sched_rr_get_interval(3RT), time(3HEAD)

NAME

siginfo – signal generation information

SYNOPSIS

```
#include <siginfo.h>
```

DESCRIPTION

If a process is catching a signal, it might request information that tells why the system generated that signal. See `sigaction`(2). If a process is monitoring its children, it might receive information that tells why a child changed state. See `waitid`(2). In either case, the system returns the information in a structure of type `siginfo_t`, which includes the following information:

```
int             si_signo        /* signal number */
int             si_errno        /* error number */
int             si_code         /* signal code */
union sigval    si_value        /* signal value */
```

`si_signo` contains the system-generated signal number. For the `waitid`(2) function, `si_signo` is always `SIGCHLD`.

If `si_errno` is non-zero, it contains an error number associated with this signal, as defined in `<errno.h>`.

`si_code` contains a code identifying the cause of the signal.

If the value of the `si_code` member is `SI_NOINFO`, only the `si_signo` member of `siginfo_t` is meaningful, and the value of all other members is unspecified.

User Signals

If the value of `si_code` is less than or equal to 0, then the signal was generated by a user process (see `kill`(2), `_lwp_kill`(2), `sigqueue`(3RT), `sigsend`(2), `abort`(3C), and `raise`(3C)) and the `siginfo` structure contains the following additional information:

```
typedef   long pid_t   si_pid   /* sending process ID */
typedef   long uid_t   si_uid   /* sending user ID */
```

If the signal was generated by a user process, the following values are defined for `si_code`:

`SI_USER`	The implementation sets `si_code` to `SI_USER` if the signal was sent by `kill`(2), `sigsend`(2), `raise`(3C) or `abort`(3C).
`SI_LWP`	The signal was sent by `_lwp_kill`(2).
`SI_QUEUE`	The signal was sent by `sigqueue`(3RT).
`SI_TIMER`	The signal was generated by the expiration of a timer created by `timer_settime`(3RT).
`SI_ASYNCIO`	The signal was generated by the completion of an asynchronous I/O request.
`SI_MESGQ`	The signal was generated by the arrival of a message on an empty message queue. See `mq_notify`(3RT).

`si_value` contains the application specified value, which is passed to the application's signal-catching function at the time of the signal delivery if `si_code` is any of `SI_QUEUE`, `SI_TIMER`, `SI_ASYNCHIO`, or `SI_MESGQ`.

System Signals

Non-user generated signals can arise for a number of reasons. For all of these cases, `si_code` contains a positive value reflecting the reason why the system generated the signal:

Signal	Code	Reason
SIGILL	ILL_ILLOPC	illegal opcode
	ILL_ILLOPN	illegal operand
	ILL_ILLADR	illegal addressing mode
	ILL_ILLTRP	illegal trap
	ILL_PRVOPC	privileged opcode
	ILL_PRVREG	privileged register
	ILL_COPROC	co-processor error
	ILL_BADSTK	internal stack error
SIGFPE	FPE_INTDIV	integer divide by zero
	FPE_INTOVF	integer overflow
	FPE_FLTDIV	floating point divide by zero
	FPE_FLTOVF	floating point overflow
	FPE_FLTUND	floating point underflow
	FPE_FLTRES	floating point inexact result
	FPE_FLTINV	invalid floating point operation
	FPE_FLTSUB	subscript out of range
SIGSEGV	SEGV_MAPERR	address not mapped to object
	SEGV_ACCERR	invalid permissions for mapped object
SIGBUS	BUS_ADRALN	invalid address alignment
	BUS_ADRERR	non-existent physical address
	BUS_OBJERR	object specific hardware error
SIGTRAP	TRAP_BRKPT	process breakpoint
	TRAP_TRACE	process trace trap
SIGCHLD	CLD_EXITED	child has exited

	CLD_KILLED	child was killed
	CLD_DUMPED	child terminated abnormally
	CLD_TRAPPED	traced child has trapped
	CLD_STOPPED	child has stopped
	CLD_CONTINUED	stopped child had continued
SIGPOLL	POLL_IN	data input available
	POLL_OUT	output buffers available
	POLL_MSG	input message available
	POLL_ERR	I/O error
	POLL_PRI	high priority input available
	POLL_HUP	device disconnected

Signals can also be generated from the resource control subsystem. Where these signals do not already possess kernel-level `siginfo` codes, the `siginfo` `si_code` will be filled with `SI_RCTL` to indicate a kernel-generated signal from an established resource control value.

Signal	Code	Reason
SIGXRES	SI_RCTL	resource–control generated signal
SIGHUP		
SIGTERM		

The uncatchable signals `SIGSTOP` and `SIGKILL` have undefined `siginfo` codes.

Signals sent with a `siginfo` code of `SI_RCTL` contain code-dependent information for kernel-generated signals:

Code	Field	Value
SI_RCTL	hr_time si_entity	process-model entity of control

In addition, the following signal-dependent information is available for kernel-generated signals:

Signal	Field	Value

SIGILL	caddr_t si_addr	address of faulting instruction
SIGFPE		
SIGSEGV	caddr_t si_addr	address of faulting memory reference
SIGBUS		
SIGCHLD	pid_t si_pid	child process ID
	int si_status	exit value or signal
SIGPOLL	long si_band	band event for `POLL_IN`, `POLL_OUT`, or `POLL_MSG`

SEE ALSO

`_lwp_kill`(2), `kill`(2), `setrctl`(2), `sigaction`(2), `sigsend`(2), `waitid`(2), `abort`(3C), `aio_read`(3RT), `mq_notify`(3RT), `raise`(3C), `signal`(3HEAD), `sigqueue`(3RT), `timer_create`(3RT), `timer_settime`(3RT)

NOTES

For `SIGCHLD` signals, if `si_code` is equal to `CLD_EXITED`, then `si_status` is equal to the exit value of the process; otherwise, it is equal to the signal that caused the process to change state. For some implementations, the exact value of `si_addr` might not be available; in that case, `si_addr` is guaranteed to be on the same page as the faulting instruction or memory reference.

NAME

signal – base signals

SYNOPSIS

```
#include <signal.h>
```

DESCRIPTION

A signal is an asynchronous notification of an event. A signal is said to be generated for (or sent to) a process when the event associated with that signal first occurs. Examples of such events include hardware faults, timer expiration and terminal activity, as well as the invocation of the `kill`(2) or `sigsend`(2) functions. In some circumstances, the same event generates signals for multiple processes. A process may request a detailed notification of the source of the signal and the reason why it was generated. See `siginfo`(3HEAD).

Signals can be generated synchronously or asynchronously. Events directly caused by the execution of code by a thread, such as a reference to an unmapped, protected, or bad memory can generate `SIGSEGV` or `SIGBUS`; a floating point exception can generate `SIGFPE`; and the execution of an illegal instruction can generate `SIGILL`. Such events are referred to as traps; signals generated by traps are said to be synchronously generated. Synchronously generated signals are initiated by a specific thread and are delivered to and handled by that thread.

Signals may also be generated by calling `kill()`, `sigqueue()`, or `sigsend()`. Events such as keyboard interrupts generate signals, such as `SIGINT`, which are sent to the target process. Such events are referred to as interrupts; signals generated by interrupts are said to be asynchronously generated. Asynchronously generated signals are not directed to a particular thread but are handled by an arbitrary thread that meets either of the following conditions:

- The thread is blocked in a call to `sigwait`(2) whose argument includes the type of signal generated.
- The thread has a signal mask that does not include the type of signal generated. A process responds to signals in similar ways whether it is using threads or it is using lightweight processes (LWPs). See `thr_create`(3THR). Each process may specify a system action to be taken in response to each signal sent to it, called the signal's disposition. All threads or LWPs in the process share the disposition. The set of system signal actions for a process is initialized from that of its parent. Once an action is installed for a specific signal, it usually remains installed until another disposition is explicitly requested by a call to either `sigaction()`, `signal()` or `sigset()`, or until the process `execs()`. See `sigaction`(2) and `signal`(3C). When a process execs, all signals whose disposition has been set to catch the signal will be set to `SIG_DFL`. Alternatively, a process may request that the system automatically reset the disposition of a signal to `SIG_DFL` after it has been caught. See `sigaction`(2) and `signal`(3C).

SIGNAL DELIVERY

A signal is said to be delivered to a process when a thread or LWP within the process takes the appropriate action for the disposition of the signal. Delivery of a signal can be blocked. There are two methods for handling delivery of a signal in a multithreaded application. The first method specifies a signal handler function to execute when the signal is received by the process. See `sigaction`(2). The second method creates a thread to handle the receipt of the signal `sigaction()` can be used

for both synchronously and asynchronously generated signals. `sigwait()` will only work for asynchronously generated signals, as synchronously generated signals are sent to the thread that caused the event. `sigwait()` is the recommended interface for use with a multithreaded application. See `sigwait`(2).

SIGNAL MASK

Each thread or LWP has a signal mask that defines the set of signals currently blocked from delivery to it. The signal mask of the main thread or LWP is inherited from the signal mask of the thread or LWP that created it in the parent process. The selection of the thread or LWP within the process that is to take the appropriate action for the signal is based on the method of signal generation and the signal masks of the threads or LWPs in the receiving process. Signals that are generated by action of a particular thread or LWP such as hardware faults are delivered to the thread or LWP that caused the signal. See `thr_sigsetmask`(3THR) or `sigprocmask`(2). See `alarm`(2) for current semantics of delivery of `SIGALRM`. Signals that are directed to a particular thread or LWP are delivered to the targeted thread or LWP. See `thr_kill`(3THR) or `_lwp_kill`(2). If the selected thread or LWP has blocked the signal, it remains pending on the thread or LWP until it is unblocked. For all other types of signal generation (for example, `kill`(2), `sigsend`(2), terminal activity, and other external events not ascribable to a particular thread or LWP) one of the threads or LWPs that does not have the signal blocked is selected to process the signal. If all the threads or LWPs within the process block the signal, it remains pending on the process until a thread or LWP in the process unblocks it. If the action associated with a signal is set to ignore the signal then both currently pending and subsequently generated signals of this type are discarded immediately for this process.

The determination of which action is taken in response to a signal is made at the time the signal is delivered to a thread or LWP within the process, allowing for any changes since the time of generation. This determination is independent of the means by which the signal was originally generated.

The signals currently defined by `<signal.h>` are as follows:

Name	Value	Default	Event
`SIGHUP`	1	Exit	Hangup (see `termio`(7I))
`SIGINT`	2	Exit	Interrupt (see `termio`(7I))
`SIGQUIT`	3	Core	Quit (see `termio`(7I))
`SIGILL`	4	Core	Illegal Instruction
`SIGTRAP`	5	Core	Trace or Breakpoint Trap
`SIGABRT`	6	Core	Abort
`SIGEMT`	7	Core	Emulation Trap
`SIGFPE`	8	Core	Arithmetic Exception

Name	Value	Default	Event
SIGKILL	9	Exit	Killed
SIGBUS	10	Core	Bus Error
SIGSEGV	11	Core	Segmentation Fault
SIGSYS	12	Core	Bad System Call
SIGPIPE	13	Exit	Broken Pipe
SIGALRM	14	Exit	Alarm Clock
SIGTERM	15	Exit	Terminated
SIGUSR1	16	Exit	User Signal 1
SIGUSR2	17	Exit	User Signal 2
SIGCHLD	18	Ignore	Child Status Changed
SIGPWR	19	Ignore	Power Fail or Restart
SIGWINCH	20	Ignore	Window Size Change
SIGURG	21	Ignore	Urgent Socket Condition
SIGPOLL	22	Exit	Pollable Event (see streamio(7I))
SIGSTOP	23	Stop	Stopped (signal)
SIGTSTP	24	Stop	Stopped (user) (see termio(7I))
SIGCONT	25	Ignore	Continued
SIGTTIN	26	Stop	Stopped (tty input) (see termio(7I))
SIGTTOU	27	Stop	Stopped (tty output) (see termio(7I))
SIGVTALRM	28	Exit	Virtual Timer Expired
SIGPROF	29	Exit	Profiling Timer Expired
SIGXCPU	30	Core	CPU time limit exceeded (see getrlimit(2))
SIGXFSZ	31	Core	File size limit exceeded (see getrlimit(2))
SIGWAITING	32	Ignore	Concurrency signal reserved by threads library
SIGLWP	33	Ignore	Inter-LWP signal reserved by threads library
SIGFREEZE	34	Ignore	Check point Freeze
SIGTHAW	35	Ignore	Check point Thaw
SIGCANCEL	36	Ignore	Cancellation signal reserved by threads library

Name	Value	Default	Event
SIGXRES	37	Ignore	Resource control exceeded (see setrctl(2))
SIGRTMIN	*	Exit	First real time signal
(SIGRTMIN+1)	*	Exit	Second real time signal
. . .			
(SIGRTMAX-1)	*	Exit	Second-to-last real time signal
SIGRTMAX	*	Exit	Last real time signal

The symbols SIGRTMIN through SIGRTMAX are evaluated dynamically in order to permit future configurability.

SIGNAL DISPOSITION

A process, using a signal(3C), sigset(3C) or sigaction(2) system call, may specify one of three dispositions for a signal: take the default action for the signal, ignore the signal, or catch the signal.

Default Action: SIG_DFL

A disposition of SIG_DFL specifies the default action. The default action for each signal is listed in the table above and is selected from the following:

Exit — When it gets the signal, the receiving process is to be terminated with all the consequences outlined in exit(2).

Core — When it gets the signal, the receiving process is to be terminated with all the consequences outlined in exit(2). In addition, a ''core image'' of the process is constructed in the current working directory.

Stop — When it gets the signal, the receiving process is to stop. When a process is stopped, all the threads and LWPs within the process also stop executing.

Ignore — When it gets the signal, the receiving process is to ignore it. This is identical to setting the disposition to SIG_IGN.

Ignore Signal: SIG_IGN

A disposition of SIG_IGN specifies that the signal is to be ignored. Setting a signal action to SIG_IGN for a signal that is pending causes the pending signal to be discarded, whether or not it is blocked. Any queued values pending are also discarded, and the resources used to queue them are released and made available to queue other signals.

Catch Signal: function address

A disposition that is a function address specifies that, when it gets the signal, the thread or LWP within the process that is selected to process the signal will execute the signal handler at the specified address. Normally, the signal handler is passed the signal number as its only argument; if the disposition was set with the sigaction() however, additional arguments may be requested (see sigaction(2)). When the signal handler returns, the receiving process resumes execution at the point it was interrupted, unless the signal handler makes other arrangements. If an invalid function address is specified, results are undefined.

If the disposition has been set with the `sigset()` or `sigaction()`, the signal is automatically blocked in the thread or LWP while it is executing the signal catcher. If a `longjmp()` is used to leave the signal catcher, then the signal must be explicitly unblocked by the user. See `setjmp`(3C), `signal`(3C) and `sigprocmask`(2).

If execution of the signal handler interrupts a blocked function call, the handler is executed and the interrupted function call returns `−1` to the calling process with `errno` set to `EINTR`. However, if the `SA_RESTART` flag is set, the function call will be transparently restarted.

Some signal-generating functions, such as high resolution timer expiration, asynchronous I/O completion, inter-process message arrival, and the `sigqueue`(3RT) function, support the specification of an application defined value, either explicitly as a parameter to the function, or in a `sigevent` structure parameter. The `sigevent` structure is defined by `<signal.h>` and contains at least the following members:

Member	Member	
Type	Name	Description
int	sigev_notify	Notification type
int	sigev_signo	Signal number
union sigval	sigev_value	Signal value

The `sigval` union is defined by `<signal.h>` and contains at least the following members:

Member	Member	
Type	Name	Description
int	sival_int	Integer signal value
void *	sival_ptr	Pointer signal value

The `sigev_notify` member specifies the notification mechanism to use when an asynchronous event occurs. The `sigev_notify` member may be defined with the following values:

`SIGEV_NONE` No asynchronous notification is delivered when the event of interest occurs.

`SIGEV_SIGNAL` A queued signal, with its value application-defined, is generated when the event of interest occurs.

Your implementation may define additional notification mechanisms.

The sigev_signo member specifies the signal to be generated.

The sigev_value member references the application defined value to be passed to the signal-catching function at the time of the signal delivery as the si_value member of the siginfo_t structure.

The sival_int member is used when the application defined value is of type int, and the sival_ptr member is used when the application defined value is a pointer.

When a signal is generated by sigqueue(3RT) or any signal–generating function which supports the specification of an application defined value, the signal is marked pending and, if the SA_SIGINFO flag is set for that signal, the signal is queued to the process along with the application specified signal value. Multiple occurrences of signals so generated are queued in FIFO order. If the SA_SIGINFO flag is not set for that signal, later occurrences of that signal's generation, when a signal is already queued, are silently discarded.

SEE ALSO

intro(2), _lwp_kill(2), alarm(2), exit(2), getrlimit(2), ioctl(2), kill(2), pause(2), setrctl(2), sigaction(2), sigaltstack(2), sigprocmask(2), sigsend(2), sigsuspend(2), sigwait(2), wait(2), setjmp(3C), siginfo(3HEAD), signal(3C), sigqueue(3RT), sigsetops(3C), thr_create(3THR), thr_kill(3THR), thr_sigsetmask(3THR), ucontext(3HEAD)

NOTES

The dispositions of the SIGKILL and SIGSTOP signals cannot be altered from their default values. The system generates an error if this is attempted.

The SIGKILL and SIGSTOP signals cannot be blocked. The system silently enforces this restriction.

Whenever a process receives a SIGSTOP, SIGTSTP, SIGTTIN, or SIGTTOU signal, regardless of its disposition, any pending SIGCONT signal are discarded.

Whenever a process receives a SIGCONT signal, regardless of its disposition, any pending SIGSTOP, SIGTSTP, SIGTTIN, and SIGTTOU signals is discarded. In addition, if the process was stopped, it is continued.

SIGPOLL is issued when a file descriptor corresponding to a STREAMS file has a "selectable" event pending. See intro(2). A process must specifically request that this signal be sent using the I_SETSIG ioctl call. Otherwise, the process will never receive SIGPOLL.

If the disposition of the SIGCHLD signal has been set with signal or sigset, or with sigaction and the SA_NOCLDSTOP flag has been specified, it will only be sent to the calling process when its children exit; otherwise, it will also be sent when the calling process's children are stopped or continued due to job control.

The name SIGCLD is also defined in this header and identifies the same signal as SIGCHLD. SIGCLD is provided for backward compatibility, new applications should use SIGCHLD.

The disposition of signals that are inherited as `SIG_IGN` should not be changed.

Signals which are generated synchronously should not be masked. If such a signal is blocked and delivered, the receiving process is killed.

NAME

socket – Internet Protocol family

SYNOPSIS

```
#include <sys/socket.h>
```

DESCRIPTION

The <sys/socket.h> header defines the unsigned integral type sa_family_t through typedef.

The <sys/socket.h> header defines the sockaddr structure that includes the following members:

sa_family_t	sa_family	/* address family */
char	sa_data[ ]	/* socket address (variable-length data) */

The <sys/socket.h> header defines the msghdr structure that includes the following members:

void	*msg_name	/* optional address */
size_t	msg_namelen	/* size of address */
struct iovec	*msg_iov	/* scatter/gather array */
int	msg_iovlen	/* members in msg_iov */
void	*msg_control	/* ancillary data, see below */
size_t	msg_controllen	/* ancillary data buffer len */
int	msg_flags	/* flags on received message */

The <sys/socket.h> header defines the cmsghdr structure that includes the following members:

size_t	cmsg_len	/* data byte count, including hdr */
int	cmsg_level	/* originating protocol */

int	cmsg_type	/* protocol-specific type */

Ancillary data consists of a sequence of pairs, each consisting of a cmsghdr structure followed by a data array. The data array contains the ancillary data message, and the cmsghdr structure contains descriptive information that allows an application to correctly parse the data.

The values for cmsg_level will be legal values for the level argument to the getsockopt() and setsockopt() functions. The SCM_RIGHTS type is supported for level SOL_SOCKET.

Ancillary data is also possible at the socket level. The <sys/socket.h> header defines the following macro for use as the cmsg_type value when cmsg_level is SOL_SOCKET:

SCM_RIGHTS — Indicates that the data array contains the access rights to be sent or received.

The <sys/socket.h> header defines the following macros to gain access to the data arrays in the ancillary data associated with a message header:

CMSG_DATA(*cmsg*) — If the argument is a pointer to a cmsghdr structure, this macro returns an unsigned character pointer to the data array associated with the cmsghdr structure.

CMSG_NXTHDR(*mhdr,cmsg*) — If the first argument is a pointer to a msghdr structure and the second argument is a pointer to a cmsghdr structure in the ancillary data, pointed to by the msg_control field of that msghdr structure, this macro returns a pointer to the next cmsghdr structure, or a null pointer if this structure is the last cmsghdr in the ancillary data.

CMSG_FIRSTHDR(*mhdr*) — If the argument is a pointer to a msghdr structure, this macro returns a pointer to the first cmsghdr structure in the ancillary data associated with this msghdr structure, or a null pointer if there is no ancillary data associated with the msghdr structure.

The <sys/socket.h> header defines the linger structure that includes the following members:

```
int                     l_onoff                 /* indicates whether
                                                   linger option is
                                                   enabled */

int                     l_linger                /* linger time, in
                                                   seconds */
```

The <sys/socket.h> header defines the following macros:

SOCK_DGRAM	Datagram socket
SOCK_STREAM	Byte-stream socket
SOCK_SEQPACKET	Sequenced-packet socket

The <sys/socket.h> header defines the following macro for use as the *level* argument of setsockopt() and getsockopt().

SOL_SOCKET	Options to be accessed at socket level, not protocol level.

The <sys/socket.h> header defines the following macros: for use as the *option_name* argument in getsockopt() or setsockopt() calls:

SO_DEBUG	Debugging information is being recorded.
SO_ACCEPTCONN	Socket is accepting connections.
SO_BROADCAST	Transmission of broadcast messages is supported.
SO_REUSEADDR	Reuse of local addresses is supported.
SO_KEEPALIVE	Connections are kept alive with periodic messages.
SO_LINGER	Socket lingers on close.
SO_OOBINLINE	Out-of-band data is transmitted in line.
SO_SNDBUF	Send buffer size.
SO_RCVBUF	Receive buffer size.
SO_ERROR	Socket error status.
SO_TYPE	Socket type.

The <sys/socket.h> header defines the following macros for use as the valid values for the msg_flags field in the msghdr structure, or the flags parameter in recvfrom(), recvmsg(), sendto(), or sendmsg() calls:

MSG_CTRUNC	Control data truncated.
MSG_EOR	Terminates a record (if supported by the protocol).
MSG_OOB	Out-of-band data.
MSG_PEEK	Leave received data in queue.

MSG_TRUNC	Normal data truncated.
MSG_WAITALL	Wait for complete message.

The <sys/socket.h> header defines the following macros:

AF_UNIX	UNIX domain sockets
AF_INET	Internet domain sockets

The <sys/socket.h> header defines the following macros:

SHUT_RD	Disables further receive operations.
SHUT_WR	Disables further send operations.
SHUT_RDWR	Disables further send and receive operations.

The following are declared as functions, and may also be defined as macros:

int accept(int *socket*, struct sockaddr **address*, size_t **address_len*);

int bind(int *socket*, const struct sockaddr **address*, size_t *address_len*);

int connect(int *socket*, const struct sockaddr **address*, size_t *address_len*);

int getpeername(int *socket*, struct sockaddr **address*, size_t **address_len*);

int getsockname(int *socket*, struct sockaddr **address*, size_t **address_len*);

int getsockopt(int *socket*, int *level*, int *option_name*, void **option_value*, size_t **option_len*);

int listen(int *socket*, int *backlog*);

ssize_t recv(int *socket*, void **buffer*, size_t *length*, int *flags*);

ssize_t recvfrom(int *socket*, void **buffer*, size_t *length*, int *flags*, struct sockaddr **address*, size_t **address_len*);

ssize_t recvmsg(int *socket*, struct msghdr **message*, int *flags*);

ssize_t send(int *socket*, const void **message*, size_t *length*, int *flags*);

ssize_t sendmsg(int *socket*, const struct msghdr **message*, int *flags*);

ssize_t sendto(int *socket*, const void **message*, size_t *length*, int *flags*, const struct sockaddr **dest_addr*, size_t *dest_len*);

int setsockopt(int *socket*, int *level*, int *option_name*, const void **option_value*, size_t *option_len*);

int shutdown(int *socket*, int *how*);

`int socket(int` *domain*, int `type`, int *protocol*);

`int socketpair(int` *domain*, int `type`, int *protocol*, `int` *socket_vector*[2]`)`;

SEE ALSO

`accept`(3SOCKET), `accept`(3XNET), `bind`(3SOCKET), `bind`(3XNET), `connect`(3SOCKET), `connect`(3XNET), `getpeername`(3SOCKET), `getpeername`(3XNET), `getsockname`(3SOCKET), `getsockname`(3XNET), `getsockopt`(3SOCKET), `getsockopt`(3XNET), `listen`(3SOCKET), `listen`(3XNET), `recv`(3SOCKET), `recv`(3XNET), `recvfrom`(3SOCKET), `recvfrom`(3XNET), `recvmsg`(3SOCKET), `recvmsg`(3XNET), `send`(3SOCKET), `send`(3XNET), `sendmsg`(3SOCKET), `sendmsg`(3XNET), `sendto`(3SOCKET), `sendto`(3XNET), `setsockopt`(3SOCKET), `setsockopt`(3XNET), `shutdown`(3SOCKET), `shutdown`(3XNET), `socket`(3SOCKET), `socket`(3XNET), `socketpair`(3SOCKET) `socketpair`(3XNET)

NAME

stat – data returned by stat system call

SYNOPSIS

```
#include <sys/types.h>

#include <sys/stat.h>
```

DESCRIPTION

The system calls `stat`, `lstat` and `fstat` return data in a `stat` structure, which is defined in `<stat.h>`.

The constants used in the `st_mode` field are also defined in this file:

```
#define     S_IFMT          /* type of file */
#define     S_IAMB          /* access mode bits */
#define     S_IFIFO         /* fifo */
#define     S_IFCHR         /* character special */
#define     S_IFDIR         /* directory */
#define     S_IFNAM         /* XENIX special named file */
#define     S_INSEM         /* XENIX semaphore subtype of IFNAM */
#define     S_INSHD         /* XENIX shared data subtype of IFNAM */
#define     S_IFBLK         /* block special */
#define     S_IFREG         /* regular */
#define     S_IFLNK         /* symbolic link */
#define     S_IFSOCK        /* socket */
#define     S_IFDOOR        /* door */
#define     S_ISUID         /* set user id on execution */
#define     S_ISGID         /* set group id on execution */
#define     S_ISVTX         /* save swapped text even after use */
#define     S_IREAD         /* read permission, owner */
#define     S_IWRITE        /* write permission, owner */
#define     S_IEXEC         /* execute/search permission, owner */
#define     S_ENFMT         /* record locking enforcement flag */
#define     S_IRWXU         /* read, write, execute: owner */
#define     S_IRUSR         /* read permission: owner */
#define     S_IWUSR         /* write permission: owner */
#define     S_IXUSR         /* execute permission: owner */
```

```
#define     S_IRWXG         /* read, write, execute: group */
#define     S_IRGRP         /* read permission: group */
#define     S_IWGRP         /* write permission: group */
#define     S_IXGRP         /* execute permission: group */
#define     S_IRWXO         /* read, write, execute: other */
#define     S_IROTH         /* read permission: other */
#define     S_IWOTH         /* write permission: other */
#define     S_IXOTH         /* execute permission: other */
```

The following macros are for POSIX conformance (see `standards`(5)):

```
#define     S_ISBLK(mode)   block special file
#define     S_ISCHR(mode)   character special file
#define     S_ISDIR(mode)   directory file
#define     S_ISFIFO(mode)  pipe or fifo file
#define     S_ISREG(mode)   regular file
#define     S_ISSOCK(mode)  socket file
```

SEE ALSO `stat`(2), `standards`(5), `types`(3HEAD)

NAME

stdarg – handle variable argument list

SYNOPSIS

```
#include <stdarg.h>
va_list pvar;

void va_start(va_list pvar, void parmN);

(type *) va_arg(va_list pvar, type);

void va_copy(va_list dest, va_list src);

void va_end(va_list pvar);
```

DESCRIPTION

This set of macros allows portable procedures that accept variable numbers of arguments of variable types to be written. Routines that have variable argument lists (such as `printf`) but do not use *stdarg* are inherently non-portable, as different machines use different argument-passing conventions.

`va_list` is a type defined for the variable used to traverse the list.

The `va_start()` macro is invoked before any access to the unnamed arguments and initializes `pvar` for subsequent use by `va_arg()` and `va_end()`. The parameter *parmN* is the identifier of the rightmost parameter in the variable parameter list in the function definition (the one just before the `, ...`). If this parameter is declared with the `register` storage class or with a function or array type, or with a type that is not compatible with the type that results after application of the default argument promotions, the behavior is undefined.

The parameter *parmN* is required under strict ANSI C compilation. In other compilation modes, *parmN* need not be supplied and the second parameter to the `va_start()` macro can be left empty (for example, `va_start(pvar, );`). This allows for routines that contain no parameters before the `...` in the variable parameter list.

The `va_arg()` macro expands to an expression that has the type and value of the next argument in the call. The parameter `pvar` should have been previously initialized by `va_start()`. Each invocation of `va_arg()` modifies `pvar` so that the values of successive arguments are returned in turn. The parameter `type` is the type name of the next argument to be returned. The type name must be specified in such a way so that the type of a pointer to an object that has the specified type can be obtained simply by postfixing a `*` to `type`. If there is no actual next argument, or if `type` is not compatible with the type of the actual next argument (as promoted according to the default argument promotions), the behavior is undefined.

The `va_copy()` macro saves the state represented by the `va_list` *src* in the `va_list` *dest*. The `va_list` passed as *dest* should not be initialized by a previous call to `va_start()`, and must be passed to `va_end()` before being reused as a parameter to `va_start()` or as the *dest* parameter of a subsequent call to `va_copy()`. The behavior is undefined should any of these restrictions not be met.

The `va_end()` macro is used to clean up.

Multiple traversals, each bracketed by `va_start` and `va_end`, are possible.

EXAMPLES

EXAMPLE 1 A sample program.

This example gathers into an array a list of arguments that are pointers to strings (but not more than `MAXARGS` arguments) with function `f1`, then passes the array as a single argument to function `f2`. The number of pointers is specified by the first argument to `f1`.

```
   #include <stdarg.h>
#define MAXARGS    31
void f1(int n_ptrs, ...)
{
    va_list ap;
    char *array[MAXARGS];
    int ptr_no = 0;

    if (n_ptrs > MAXARGS)
        n_ptrs = MAXARGS;
    va_start(ap, n_ptrs);
    while (ptr_no < n_ptrs)
        array[ptr_no++] = va_arg(ap, char*);
    va_end(ap);
    f2(n_ptrs, array);
}
```

Each call to `f1` shall have visible the definition of the function or a declaration such as

```
void f1(int, ...)
```

SEE ALSO

`vprintf`(3C)

NOTES

It is up to the calling routine to specify in some manner how many arguments there are, since it is not always possible to determine the number of arguments from the stack frame. For example, `execl` is passed a zero pointer to signal the end of the list. `printf` can tell how many arguments there are by the format. It is non-portable to specify a second argument of `char`, `short`, or `float` to `va_arg`, because arguments seen by the called function are not `char`, `short`, or `float`. C converts `char` and `short` arguments to `int` and converts `float` arguments to `double` before passing them to a function.

NAME

time – time types

SYNOPSIS

```
#include <time.h>
```

DESCRIPTION

The <time.h> header declares the structure tm, which includes the following members:

```
int        tm_sec          seconds [0,61]
int        tm_min          minutes [0,59]
int        tm_hour         hour [0,23]
int        tm_mday         day of month [1,31]
int        tm_mon          month of year [0,11]
int        tm_year         years since 1900
int        tm_wday         day of week [0,6] (Sunday = 0)
int        tm_yday         day of year [0,365]
int        tm_isdst        daylight savings flag
```

The value of tm_isdst is positive if Daylight Saving Time is in effect, 0 if Daylight Saving Time is not in effect, and negative if the information is not available.

This header defines the following symbolic names:

NULL	Null pointer constant.
CLK_TCK	Number of clock ticks per second returned by the times(2) function.
CLOCKS_PER_SEC	A number used to convert the value returned by the clock(3C) function into seconds.

The <time.h> header declares the structure timespec, which has the following members:

```
time_t  tv_sec    seconds
long    tv_nsec   nanoseconds
```

This header also declares the itimerspec structure, which has at least the following members:

```
struct timespec   it_interval  timer period
struct timespec   it_value     timer expiration
```

The following manifest constants are defined:

CLOCK_REALTIME	The identifier of the systemwide realtime clock.
TIMER_ABSTIME	Flag indicating time is absolute with respect to the clock associated with a timer.

The clock_t, size_t and time_t types are defined as described in <sys/types.h>.

Although the value of CLOCKS_PER_SEC is 1 million on all Solaris systems, it may be variable on other systems and it should not be assumed that CLOCKS_PER_SEC is a compile-time constant.

The value of CLK_TCK is currently the same as the value of sysconf(_SC_CLK_TCK); however, new applications should call sysconf(3C) because the CLK_TCK macro may be withdrawn in a future issue.

The <time.h> header provides a declaration for getdate_err.

The following are declared as variables:

```
extern int       daylight;
extern long int  timezone;
extern char      *tzname[ ];
```

USAGE

The range [0,61] for tm_sec allows for the occasional leap second or double leap second.

tm_year is a signed value, therefore years before 1900 may be represented.

SEE ALSO

time(2), times(2), utime(2), asctime(3C), clock(3C), clock_settime(3RT), ctime(3C), difftime(3C), getdate(3C), gmtime(3C), localtime(3C), mktime(3C), nanosleep(3RT), strftime(3C), strptime(3C), sysconf(3C), timer_create(3RT), timer_delete(3RT), timer_settime(3RT), tzset(3C)

NAME

types32 – fixed-width data types

SYNOPSIS

```
#include <sys/types32.h>
```

DESCRIPTION

The following fixed-width data types defined in `<sys/types32.h>` correspond to the sign and sizes of types in the 32-bit environment that can be used for compatibility and interoperability purposes in either the 32-bit or 64-bit environment.

typedef	int32_t	blkcnt32_t
typedef	uint32_t	caddr32_t
typedef	int32_t	clock32_t
typedef	int32_t	daddr32_t
typedef	uint32_t	dev32_t
typedef	uint32_t	fsblkcnt32_t
typedef	uint32_t	fsfilcnt32_t
typedef	int32_t	gid32_t
typedef	int32_t	id32_t
typedef	uint32_t	ino32_t
typedef	int32_t	key32_t
typedef	uint32_t	major32_t
typedef	uint32_t	minor32_t
typedef	uint32_t	mode32_t
typedef	uint32_t	nlink32_t
typedef	int32_t	pid32_t
typedef	uint32_t	rlim32_t
typedef	uint32_t	size32_t
typedef	int32_t	ssize32_t
typedef	time32_t	int32_t
typedef	uid32_t	int32_t

NAME

types – primitive system data types

SYNOPSIS

```
#include <sys/types.h>
```

DESCRIPTION

The data types defined in <sys/types.h> are discussed.

32-bit Solaris

The data types listed below are defined in <sys/types.h> for 32-bit Solaris.

```
typedef    struct    { int r[1]; } *physadr;
typedef    long              clock_t;
typedef    long              daddr_t;
typedef    char *            caddr_t;
typedef    unsigned char     unchar;
typedef    unsigned short    ushort;
typedef    unsigned int      uint;
typedef    unsigned long     ulong_t;
typedef    unsigned long     ino_t;
typedef    long              uid_t;
typedef    long              gid_t;
typedef    ulong_t           nlink_t;
typedef    ulong_t           mode_t;
typedef    short             cnt_t;
typedef    long              time_t;
typedef    int               label_t[10];
typedef    ulong_t           dev_t;
typedef    long              off_t;
typedef    long              pid_t;
typedef    long              paddr_t;
typedef    int               key_t;
typedef    unsigned char     use_t;
typedef    short             sysid_t;
typedef    short             index_t;
typedef    short             lock_t;
typedef    unsigned int      size_t;
typedef    long              clock_t;
typedef    long              pid_t;
```

64-bit Solaris

The data types listed below are defined in <sys/types.h> for 64-bit Solaris.

```
typedef     long        blkcnt_t
typedef     long        clock_t
typedef     long        daddr_t
typedef     ulong_t     dev_t
typedef     ulong_t     fsblkcnt_t
typedef     ulong_t     fsfilcnt_t
typedef     int         gid_t
typedef     int         id_t
typedef     long        ino_t
typedef     int         key_t
typedef     uint_t      major_t
typedef     uint_t      minor_t
typedef     uint_t      mode_t
typedef     uint_t      nlink_t
typedef     int         pid_t
typedef     ptrdiff_t   intptr_t
typedef     ulong_t     rlim_t
typedef     ulong_t     size_t
typedef     uint_t      speed_t
```

```
typedef     long        ssize_t
typedef     long        suseconds_t
typedef     uint_t      tcflag_t
typedef     long        time_t
typedef     int         uid_t
typedef     int         wchar_t
```

USAGE

The `daddr_t` type is used for disk addresses except in an inode on disk. Times are encoded in seconds since 00:00:00 UTC, January 1, 1970. The major and minor parts of a device code specify kind and unit number of a device and are installation-dependent. Offsets are measured in bytes from the beginning of a file.

The `label_t[ ]` types are used to save the processor state while another process is running.

NOTES

For 32-bit programs, pointers and the C data types `int` and `long` are all 32-bit quantities. For 64-bit programs, pointers and the C data type `long` are defined as 64-bit quantities.

The preprocessor symbol `_ILP32`, made visible by the inclusion of `<sys/types.h>` can be used with the preprocessor `#ifdef` construct to define sections of code that will *only* be compiled as part of a 32-bit version of a given C program.

The preprocessor symbol `_LP64` can be used in the same way to define sections of code that will *only* be compiled as part of a 64-bit version of a given C program.

For example:

```
#include <sys/types.h>
       ...

   #ifdef _LP64
       printf("The data model is LP64 in this environment\n");
   #else
   #ifdef _ILP32
       printf("The data model is ILP32 in this environment\n");
   #else
   #error    "Unknown data model!"
   #endif
   #endif
```

NAME

ucontext – user context

SYNOPSIS

```
#include <ucontext.h>
```

DESCRIPTION

The `ucontext` structure defines the context of a thread of control within an executing process.

This structure includes at least the following members:

```
ucontext_t  uc_link
sigset_t    uc_sigmask
stack_t     uc_stack
mcontext_t  uc_mcontext
```

`uc_link` is a pointer to the context that to be resumed when this context returns. If `uc_link` is equal to 0, then this context is the main context, and the process exits when this context returns.

`uc_sigmask` defines the set of signals that are blocked when this context is active [see `sigprocmask`(2)].

`uc_stack` defines the stack used by this context [see `sigaltstack`(2)].

`uc_mcontext` contains the saved set of machine registers and any implementation specific context data. Portable applications should not modify or access `uc_mcontext`.

SEE ALSO

`getcontext`(2), `sigaction`(2), `sigaltstack`(2), `sigprocmask`(2), `makecontext`(3C)

NAME

un – definitions for UNIX-domain sockets

SYNOPSIS

```
#include <sys/un.h>
```

DESCRIPTION

The <sys/un.h> header defines the sockaddr_un structure that includes the following members:

```
sa_family_t  sun_family   /* address family */
char         sun_path[]   /* socket pathname */
```

The sockaddr_un structure is used to store addresses for UNIX domain sockets. Values of this type must be cast to struct sockaddr for use with the socket interfaces.

The <sys/un.h> header defines the type sa_family_t as described in socket(3HEAD).

SEE ALSO

bind(3SOCKET), bind(3XNET), socket(3HEAD), socket(3SOCKET), socket(3XNET), socketpair(3SOCKET), socketpair(3XNET)

NAME

unistd – standard symbolic constants and types

SYNOPSIS

```
#include <unistd.h>
```

DESCRIPTION

The `<unistd.h>` header defines the symbolic constants and structures which are not already defined or declared in some other header. The contents of this header are shown below.

Version Test Macros

The following symbolic constants are defined (with fixed values):

`_POSIX_VERSION`	Integer value indicating version of the POSIX standard (C language binding). See `standards`(5).
`_POSIX2_VERSION`	Integer value indicating version of the POSIX.2 standard (Commands). `_POSIX2_C_VERSION` Integer value indicating version of the POSIX.2 standard (C language binding).
`_XOPEN_VERSION`	Integer value indicating version of the XPG to which system conforms.
`_XOPEN_XCU_VERSION`	Integer value indicating the version of the XCU specification to which the implementation conforms. If this constant is not defined, use the `sysconf`(3C) function to determine which features are supported.

Mandatory Symbolic Constants

The following symbolic constants are either undefined or defined with a value other than –1. If a constant is undefined, an application should use the `sysconf`(3C), `pathconf`(2), or `fpathconf`(2) functions to determine which features are present on the system at that time or for the particular pathname in question.

`_POSIX_JOB_CONTROL`	Implementation supports job control.
`_POSIX_SAVED_IDS`	The exec functions (see `exec`(2)) save the effective user and group.
`_POSIX_THREADS`	The implementation supports the threads option.
`_POSIX_THREAD_ATTR_STACKADDR`	The implementation supports the thread stack address attribute option.
`_POSIX_THREAD_ATTR_STACKSIZE`	The implementation supports the thread stack size attribute option.
`_POSIX_THREAD_PROCESS_SHARED`	The implementation supports the process-shared synchronization option.
`_POSIX_THREAD_SAFE_FUNCTIONS`	The implementation supports the thread-safe functions option.
`_XOPEN_XPG3`	X/Open Specification, February 1992, System Interfaces and Headers, Issue 3 (ISBN: 1-872630-37-5, C212); this

	specification was formerly X/Open Portability Guide, Issue 3, Volume 2, January 1989, XSI System Interface and Headers (ISBN: 0-13-685843-0, XO/XPG/89/003).
`_XOPEN_XPG4`	X/Open CAE Specification, July 1992, System Interfaces and Headers, Issue 4 (ISBN: 1-872630-47-2, C202).
`_XOPEN_UNIX`	X/Open CAE Specification, January 1997, System Interfaces and Headers, Issue 5 (ISBN: 1-85912-181-0, C606).

Constants for Options and Feature Groups

The following symbolic constants are defined to have the value –1 if the implementation will never provide the feature, and to have a value other than –1 if the implementation always provides the feature. If these are undefined, the `sysconf()` function can be used to determine whether the feature is provided for a particular invocation of the application.

`_POSIX2_C_BIND`
Implementation supports the C Language Binding option.

`_POSIX2_C_DEV`
Implementation supports the C Language Development Utilities option.

`_POSIX2_CHAR_TERM`
Implementation supports at least one terminal type.

`_POSIX2_LOCALEDEF`
Implementation supports the creation of locales by the `localedef`(1) utility.

`_POSIX2_SW_DEV`
Implementation supports the Software Development Utilities option.

`_POSIX2_UPE`
The implementation supports the User Portability Utilities option.

`_XOPEN_ENH_I18N`
The implementation supports the Issue 4, Version 2 Enhanced Internationalization Feature Group.

`_XOPEN_LEGACY`
The implementation supports the Legacy Feature Group.

`_XOPEN_REALTIME`
The implementation supports the X/Open Realtime Feature Group.

`_XOPEN_SHM`
The implementation supports the Issue 4, Version 2 Shared Memory Feature Group.

`_XBS5_ILP32_OFF32`
Implementation provides a C-language compilation environment with 32-bit `int`, `long`, `pointer` and `off_t` types.

_XBS5_ILP32_OFFBIG
Implementation provides a C-language compilation environment with 32-bit int, long and pointer types and an off_t type using at least 64 bits.

_XBS5_LP64_OFF64
Implementation provides a C-language compilation environment with 32-bit int and 64-bit long, pointer and off_t types.

_XBS5_LPBIG_OFFBIG
Implementation provides a C-language compilation environment with an int type using at least 32 bits and long, pointer and off_t types using at least 64 bits.

If _XOPEN_REALTIME is defined to have a value other than –1 then the following symbolic constants will be defined to an unspecified value to indicate that the features are supported.

_POSIX_ASYNCHRONOUS_IO	Implementation supports the Asynchronous Input and Output option.
_POSIX_MEMLOCK	Implementation supports the Process Memory Locking option.
_POSIX_MEMLOCK_RANGE	Implementation supports the Range Memory Locking option.
_POSIX_MESSAGE_PASSING	Implementation supports the Message Passing option.
_POSIX_PRIORITY_SCHEDULING	Implementation supports the Process Scheduling option.
_POSIX_REALTIME_SIGNALS	Implementation supports the Realtime Signals Extension option.
_POSIX_SEMAPHORES	Implementation supports the Semaphores option.
_POSIX_SHARED_MEMORY_OBJECTS	Implementation supports the Shared Memory Objects option.
_POSIX_SYNCHRONIZED_IO	Implementation supports the Synchronized Input and Output option.
_POSIX_TIMERS	Implementation supports the Timers option.

The following symbolic constants are always defined to unspecified values to indicate that the functionality is always present on XSI-conformant systems.

_POSIX_FSYNC	Implementation supports the File Synchronisation option.
_POSIX_MAPPED_FILES	Implementation supports the Memory Mapped Files option.

`_POSIX_MEMORY_PROTECTION`	Implementation supports the Memory Protection option.

Execution-time Symbolic Constants

If any of the following constants are not defined in the header `<unistd.h>`, the value varies depending on the file to which it is applied.

If any of the following constants are defined to have value –1 in the header `<unistd.h>`, the implementation will not provide the option on any file; if any are defined to have a value other than –1 in the header `<unistd.h>`, the implementation will provide the option on all applicable files.

All of the following constants, whether defined in `<unistd.h>` or not, may be queried with respect to a specific file using the `pathconf()` or `fpathconf()` functions.

`_POSIX_ASYNC_IO`	Asynchronous input or output operations may be performed for the associated file.
`_POSIX_PRIO_IO`	Prioritized input or output operations may be performed for the associated file.
`_POSIX_SYNC_IO`	Synchronized input or output operations may be performed for the associated file.

Constants for Functions

The following constant is defined:

`NULL` Null pointer.

The following symbolic constants are defined for the `access`(2) function:

`R_OK` Test for read permission.

`W_OK` Test for write permission.

`X_OK` Test for execute (search) permission.

`F_OK` Test for existence of file. The constants `F_OK`, `R_OK`, `W_OK`, and `X_OK`, and the expressions `R_OK | W_OK`, `R_OK | X_OK`, and `R_OK | W_OK | X_OK` all have distinct values.

The following symbolic constants are defined for the `lockf`(3C) function:

`F_ULOCK`	Unlock a previously locked region.
`F_LOCK`	Lock a region for exclusive use.
`F_TLOCK`	Test and lock a region for exclusive use.
`F_TEST`	Test a region for other processes locks.

The following symbolic constants are defined for the `lseek`(2) and `fcntl`(2) functions (they have distinct values):

`SEEK_SET` Set file offset to *offset*.

SEEK_CUR	Set file offset to current plus *offset*.
SEEK_END	Set file offset to EOF plus *offset*.

The following symbolic constants are defined for the confstr(3C) function for both SPARC and IA:

_CS_LFS64_CFLAGS	_CS_LFS64_LDFLAGS
_CS_LFS64_LIBS	_CS_LFS64_LINTFLAGS
_CS_LFS_CFLAGS	_CS_LFS_LDFLAGS
_CS_LFS_LIBS	_CS_LFS_LINTFLAGS
_CS_PATH	_CS_XBS5_ILP32_OFF32_CFLAGS
_CS_XBS5_ILP32_OFF32_LDFLAGS	_CS_XBS5_ILP32_OFF32_LIBS
_CS_XBS5_ILP32_OFF32_LINTFLAGS	_CS_XBS5_ILP32_OFFBIG_CFLAGS
_CS_XBS5_ILP32_OFFBIG_LDFLAGS	_CS_XBS5_ILP32_OFFBIG_LIBS
_CS_XBS5_ILP32_OFFBIG_LINTFLAGS	

The followwing symbolic constants are defined for the confstr() function for SPARC only:

_CS_XBS5_LP64_OFF64_CFLAGS	_CS_XBS5_LP64_OFF64_LDFLAGS
_CS_XBS5_LP64_OFF64_LIBS	_CS_XBS5_LP64_OFF64_LINTFLAGS
_CS_XBS5_LPBIG_OFFBIG_CFLAGS	_CS_XBS5_LPBIG_OFFBIG_LDFLAGS
_CS_XBS5_LPBIG_OFFBIG_LIBS	_CS_XBS5_LPBIG_OFFBIG_LINTFLAGS

The following symbolic constants are defined for the sysconf(3C) function:

_SC_2_C_BIND	_SC_2_C_DEV
_SC_2_C_VERSION	_SC_2_FORT_DEV
_SC_2_FORT_RUN	_SC_2_LOCALEDEF
_SC_2_SW_DEV	_SC_2_UPE
_SC_2_VERSION	_SC_AIO_LISTIO_MAX
_SC_AIO_MAX	_SC_AIO_PRIO_DELTA_MAX
_SC_ARG_MAX	_SC_ASYNCHRONOUS_IO

_SC_ATEXIT_MAX	_SC_AVPHYS_PAGES
_SC_BC_BASE_MAX	_SC_BC_DIM_MAX
_SC_BC_SCALE_MAX	_SC_BC_STRING_MAX
_SC_CHILD_MAX	_SC_CLK_TCK
_SC_COLL_WEIGHTS_MAX	_SC_DELAYTIMER_MAX
_SC_EXPR_NEST_MAX	_SC_FSYNC
_SC_GETGR_R_SIZE_MAX	_SC_GETPW_R_SIZE_MAX
_SC_IOV_MAX	_SC_JOB_CONTROL
_SC_LINE_MAX	_SC_LOGIN_NAME_MAX
_SC_LOGNAME_MAX	_SC_MAPPED_FILES
_SC_MEMLOCK	_SC_MEMLOCK_RANGE
_SC_MEMORY_PROTECTION	_SC_MESSAGE_PASSING
_SC_MQ_OPEN_MAX	_SC_MQ_PRIO_MAX
_SC_NGROUPS_MAX	_SC_NPROCESSORS_CONF
_SC_NPROCESSORS_ONLN	_SC_OPEN_MAX
_SC_PAGESIZE	_SC_PAGE_SIZE
_SC_PASS_MAX	_SC_PHYS_PAGES
_SC_PRIORITIZED_IO	_SC_PRIORITY_SCHEDULING
_SC_REALTIME_SIGNALS	_SC_RE_DUP_MAX
_SC_RTSIG_MAX	_SC_SAVED_IDS
_SC_SEMAPHORES	_SC_SEM_NSEMS_MAX
_SC_SEM_VALUE_MAX	_SC_SHARED_MEMORY_OBJECTS
_SC_SIGQUEUE_MAX	_SC_STREAM_MAX
_SC_SYNCHRONIZED_IO	_SC_THREAD_ATTR_STACKADDR
_SC_THREAD_ATTR_STACKSIZE	_SC_THREAD_DESTRUCTOR_ITERATIONS
_SC_THREAD_KEYS_MAX	_SC_THREAD_PRIO_INHERIT
_SC_THREAD_PRIO_PROTECT	_SC_THREAD_PRIORITY_SCHEDULING
_SC_THREAD_PROCESS_SHARED	_SC_THREADS
_SC_THREAD_SAFE_FUNCTIONS	_SC_THREAD_STACK_MIN
_SC_THREAD_THREADS_MAX	_SC_TIMER_MAX

_SC_TIMERS	_SC_TTY_NAME_MAX
_SC_TZNAME_MAX	_SC_VERSION
_SC_XBS5_ILP32_OFF32	_SC_XBS5_ILP32_OFFBIG
_SC_XBS5_LP64_OFF64	_SC_XBS5_LPBIG_OFFBIG
_SC_XOPEN_CRYPT	_SC_XOPEN_ENH_I18N
_SC_XOPEN_SHM	_SC_XOPEN_UNIX
_SC_XOPEN_VERSION	_SC_XOPEN_XCU_VERSION

The two constants _SC_PAGESIZE and _SC_PAGE_SIZE may be defined to have the same value.

The following symbolic constants are defined for the fpathconf(2) function:

_PC_ASYNC_IO	_PC_CHOWN_RESTRICTED
_PC_FILESIZEBITS	_PC_LINK_MAX
_PC_MAX_CANON	_PC_MAX_INPUT
_PC_NAME_MAX	_PC_NO_TRUNC
_PC_PATH_MAX	_PC_PIPE_BUF
_PC_PRIO_IO	_PC_SYNC_IO
_PC_VDISABLE	_PC_XATTR_ENABLED
_PC_XATTR_EXISTS	

The following symbolic constants are defined for file streams:

STDIN_FILENO	File number (0) of stdin.
STDOUT_FILENO	File number (1) of stout.
STDERR_FILENO	File number (2) of stderr. The following pathnames are defined:
GF_PATH	Pathname of the group file.
PF_PATH	Pathname of the passwd file.

SEE ALSO access(2), exec(2), fcntl(2), fpathconf(2), lseek(2), confstr(3C), lockf(3C), sysconf(3C), termios(3C), group(4), passwd(4), standards(5), termio(7I)

NAME

values – machine-dependent values

SYNOPSIS

```
#include <values.h>
```

DESCRIPTION

This file contains a set of manifest constants, conditionally defined for particular processor architectures.

The model assumed for integers is binary representation (one's or two's complement), where the sign is represented by the value of the high-order bit.

`BITS(type)`	The number of bits in a specified type (for example, `int`).
`HIBITS`	The value of a short integer with only the high-order bit set.
`HIBITL`	The value of a long integer with only the high-order bit set.
`HIBITI`	The value of a regular integer with only the high-order bit set.
`MAXSHORT`	The maximum value of a signed short integer.
`MAXLONG`	The maximum value of a signed long integer.
`MAXINT`	The maximum value of a signed regular integer.
`MAXFLOAT`, `LN_MAXFLOAT`	The maximum value of a single-precision floating-point number, and its natural logarithm.
`MAXDOUBLE`, `LN_MAXDOUBLE`	The maximum value of a double-precision floating-point number, and its natural logarithm.
`MINFLOAT`, `LN_MINFLOAT`	The minimum positive value of a single-precision floating-point number, and its natural logarithm.
`MINDOUBLE`, `LN_MINDOUBLE`	The minimum positive value of a double-precision floating-point number, and its natural logarithm.
`FSIGNIF`	The number of significant bits in the mantissa of a single-precision floating-point number.
`DSIGNIF`	The number of significant bits in the mantissa of a double-precision floating-point number.

SEE ALSO | `intro`(3) `math`(3HEAD)

NAME

varargs – handle variable argument list

SYNOPSIS

```
#include <varargs.h>
va_alist
va_dcl
va_list pvar;

void va_start(va_listpvar);

type va_arg(va_list pvar, type);

void va_end(va_list pvar);
```

DESCRIPTION

This set of macros allows portable procedures that accept variable argument lists to be written. Routines that have variable argument lists (such as `printf`(3C)) but do not use `varargs` are inherently non-portable, as different machines use different argument-passing conventions.

`va_alist` is used as the parameter list in a function header.

`va_dcl` is a declaration for `va_alist`. No semicolon should follow `va_dcl`.

`va_list` is a type defined for the variable used to traverse the list.

`va_start` is called to initialize `pvar` to the beginning of the list.

`va_arg` will return the next argument in the list pointed to by `pvar`. `type` is the type the argument is expected to be. Different types can be mixed, but it is up to the routine to know what type of argument is expected, as it cannot be determined at runtime.

`va_end` is used to clean up.

Multiple traversals, each bracketed by `va_start` and `va_end`, are possible.

EXAMPLES

EXAMPLE 1 A sample program.

This example is a possible implementation of `execl` (see `exec`(2)).

```
#include <unistd.h>
#include <varargs.h>
#define MAXARGS    100
/*    execl is called by
        execl(file, arg1, arg2, ..., (char *)0);
*/
execl(va_alist)
va_dcl
{
    va_list ap;
    char *file;
    char *args[MAXARGS];        /* assumed big enough*/
    int argno = 0;

    va_start(ap);
    file = va_arg(ap, char *);
    while ((args[argno++] = va_arg(ap, char *)) != 0)
```

EXAMPLE 1 A sample program. *(Continued)*

```
        ;
    va_end(ap);
    return execv(file, args);
}
```

SEE ALSO exec(2), printf(3C), vprintf(3C), stdarg(3HEAD)

NOTES It is up to the calling routine to specify in some manner how many arguments there are, since it is not always possible to determine the number of arguments from the stack frame. For example, `execl` is passed a zero pointer to signal the end of the list. `printf` can tell how many arguments are there by the format.

It is non-portable to specify a second argument of `char`, `short`, or `float` to `va_arg`, since arguments seen by the called function are not `char`, `short`, or `float`. C converts `char` and `short` arguments to `int` and converts `float` arguments to `double` before passing them to a function.

`stdarg` is the preferred interface.

NAME

wstat – wait status

SYNOPSIS

```
#include <sys/wait.h>
```

DESCRIPTION

When a process waits for status from its children via either the `wait` or `waitpid` function, the status returned may be evaluated with the following macros, defined in `<sys/wait.h>`. These macros evaluate to integral expressions. The *stat* argument to these macros is the integer value returned from `wait` or `waitpid`.

`WIFEXITED(`*stat*`)`	Evaluates to a non-zero value if status was returned for a child process that terminated normally.
`WEXITSTATUS(`*stat*`)`	If the value of `WIFEXITED(`*stat*`)` is non-zero, this macro evaluates to the exit code that the child process passed to `_exit()` (see `exit`(2)) or `exit`(3C), or the value that the child process returned from `main`.
`WIFSIGNALED(`*stat*`)`	Evaluates to a non-zero value if status was returned for a child process that terminated due to the receipt of a signal.
`WTERMSIG(`*stat*`)`	If the value of `WIFSIGNALED(`*stat*`)` is non-zero, this macro evaluates to the number of the signal that caused the termination of the child process.
`WIFSTOPPED(`*stat*`)`	Evaluates to a non-zero value if status was returned for a child process that is currently stopped.
`WSTOPSIG(`*stat*`)`	If the value of `WIFSTOPPED(`*stat*`)` is non-zero, this macro evaluates to the number of the signal that caused the child process to stop.
`WIFCONTINUED(`*stat*`)`	Evaluates to a non-zero value if status was returned for a child process that has continued.
`WCOREDUMP(`*stat*`)`	If the value of `WIFSIGNALED` (*stat*) is non-zero, this macro evaluates to a non-zero value if a core image of the terminated child was created.

SEE ALSO

`exit`(2), `wait`(2), `waitpid`(2), `exit`(3C)

Index

M

N

O

P

R

S

T

U

V

W

X

Y