man pages section 3: Library Interfaces and Headers

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

Part No: 816–5173–10
January 2005

Copyright 2005 Sun Microsystems, Inc. 4150 Network Circle, Santa Clara, CA 95054 U.S.A. All rights reserved.

This product or document is protected by copyright and distributed under licenses restricting its use, copying, distribution, and decompilation. No part of this product or document may be reproduced in any form by any means without prior written authorization of Sun and its licensors, if any. Third-party software, including font technology, is copyrighted and licensed from Sun suppliers.

Parts of the product may be derived from Berkeley BSD systems, licensed from the University of California. UNIX is a registered trademark in the U.S. and other countries, exclusively licensed through X/Open Company, Ltd.

Sun, Sun Microsystems, the Sun logo, docs.sun.com, AnswerBook, AnswerBook2, and Solaris are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. in the U.S. and other countries. Products bearing SPARC trademarks are based upon an architecture developed by Sun Microsystems, Inc.

The OPEN LOOK and Sun™ Graphical User Interface was developed by Sun Microsystems, Inc. for its users and licensees. Sun acknowledges the pioneering efforts of Xerox in researching and developing the concept of visual or graphical user interfaces for the computer industry. Sun holds a non-exclusive license from Xerox to the Xerox Graphical User Interface, which license also covers Sun's licensees who implement OPEN LOOK GUIs and otherwise comply with Sun's written license agreements.

U.S. Government Rights – Commercial software. Government users are subject to the Sun Microsystems, Inc. standard license agreement and applicable provisions of the FAR and its supplements.

DOCUMENTATION IS PROVIDED "AS IS" AND ALL EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS AND WARRANTIES, INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT, ARE DISCLAIMED, EXCEPT TO THE EXTENT THAT SUCH DISCLAIMERS ARE HELD TO BE LEGALLY INVALID.

Copyright 2005 Sun Microsystems, Inc. 4150 Network Circle, Santa Clara, CA 95054 U.S.A. Tous droits réservés.

Ce produit ou document est protégé par un copyright et distribué avec des licences qui en restreignent l'utilisation, la copie, la distribution, et la décompilation. Aucune partie de ce produit ou document ne peut être reproduite sous aucune forme, par quelque moyen que ce soit, sans l'autorisation préalable et écrite de Sun et de ses bailleurs de licence, s'il y en a. Le logiciel détenu par des tiers, et qui comprend la technologie relative aux polices de caractères, est protégé par un copyright et licencié par des fournisseurs de Sun.

Des parties de ce produit pourront être dérivées du système Berkeley BSD licenciés par l'Université de Californie. UNIX est une marque déposée aux Etats-Unis et dans d'autres pays et licenciée exclusivement par X/Open Company, Ltd.

Sun, Sun Microsystems, le logo Sun, docs.sun.com, AnswerBook, AnswerBook2, et Solaris sont des marques de fabrique ou des marques déposées, de Sun Microsystems, Inc. aux Etats-Unis et dans d'autres pays. Toutes les marques SPARC sont utilisées sous licence et sont des marques de fabrique ou des marques déposées de SPARC International, Inc. aux Etats-Unis et dans d'autres pays. Les produits portant les marques SPARC sont basés sur une architecture développée par Sun Microsystems, Inc.

L'interface d'utilisation graphique OPEN LOOK et Sun™ a été développée par Sun Microsystems, Inc. pour ses utilisateurs et licenciés. Sun reconnaît les efforts de pionniers de Xerox pour la recherche et le développement du concept des interfaces d'utilisation visuelle ou graphique pour l'industrie de l'informatique. Sun détient une licence non exclusive de Xerox sur l'interface d'utilisation graphique Xerox, cette licence couvrant également les licenciés de Sun qui mettent en place l'interface d'utilisation graphique OPEN LOOK et qui en outre se conforment aux licences écrites de Sun.

CETTE PUBLICATION EST FOURNIE "EN L'ETAT" ET AUCUNE GARANTIE, EXPRESSE OU IMPLICITE, N'EST ACCORDEE, Y COMPRIS DES GARANTIES CONCERNANT LA VALEUR MARCHANDE, L'APTITUDE DE LA PUBLICATION A REPONDRE A UNE UTILISATION PARTICULIERE, OU LE FAIT QU'ELLE NE SOIT PAS CONTREFAISANTE DE PRODUIT DE TIERS. CE DENI DE GARANTIE NE S'APPLIQUERAIT PAS, DANS LA MESURE OU IL SERAIT TENU JURIDIQUEMENT NUL ET NON AVENU.

050105@10536

Contents

Preface 9

Introduction 15
Intro(3) 16

Library Interfaces and Headers 29
acct.h(3HEAD) 30
aio.h(3HEAD) 32
ar.h(3HEAD) 33
assert.h(3HEAD) 36
complex.h(3HEAD) 37
cpio.h(3HEAD) 39
dirent.h(3HEAD) 41
errno.h(3HEAD) 42
fcntl.h(3HEAD) 43
fenv.h(3HEAD) 47
float.h(3HEAD) 50
floatingpoint.h(3HEAD) 53
fmtmsg.h(3HEAD) 55
fnmatch.h(3HEAD) 57
ftw.h(3HEAD) 58
glob.h(3HEAD) 59
grp.h(3HEAD) 60
iconv.h(3HEAD) 61
if.h(3HEAD) 62
inet.h(3HEAD) 63

in.h(3HEAD) 64
inttypes.h(3HEAD) 66
ipc.h(3HEAD) 68
iso646.h(3HEAD) 69
langinfo.h(3HEAD) 70
libadm(3LIB) 74
libaio(3LIB) 75
libauto_ef(3LIB) 76
libbsdmalloc(3LIB) 77
libbsm(3LIB) 78
libc(3LIB) 80
libc_db(3LIB) 109
libcfgadm(3LIB) 112
libcontract(3LIB) 113
libcpc(3LIB) 115
libcrypt(3LIB) 117
libcurses(3LIB) 118
libcurses(3LIBUCB) 125
libdat(3LIB) 127
libdbm(3LIBUCB) 130
libdevid(3LIB) 131
libdevinfo(3LIB) 132
libdl(3LIB) 136
libdmi(3LIB) 137
libdmici(3LIB) 138
libdmimi(3LIB) 139
libdoor(3LIB) 140
libdtrace(3LIB) 141
libefi(3LIB) 142
libelf(3LIB) 143
libexacct(3LIB) 145
libform(3LIB) 147
libgen(3LIB) 149
libgen.h(3HEAD) 151
libgss(3LIB) 152
libhbaapi(3LIB) 154
libidnkit(3LIB) 158
libintl(3LIB) 160

libkstat(3LIB) 161
libkvm(3LIB) 162
libl(3LIB) 163
liblayout(3LIB) 164
liblgrp(3LIB) 165
libm(3LIB) 166
libmail(3LIB) 182
libmalloc(3LIB) 183
libmapmalloc(3LIB) 184
libmd5(3LIB) 185
libmenu(3LIB) 186
libmlib(3LIB) 188
libmp(3LIB) 245
libmtmalloc(3LIB) 246
libmvec(3LIB) 247
libnls(3LIB) 248
libnsl(3LIB) 249
libnvpair(3LIB) 257
libpam(3LIB) 260
libpanel(3LIB) 262
libpctx(3LIB) 263
libpicl(3LIB) 264
libpicltree(3LIB) 265
libpkcs11(3LIB) 266
libplot(3LIB) 269
libpool(3LIB) 271
libproject(3LIB) 279
libpthread(3LIB) 280
librac(3LIB) 283
libresolv(3LIB) 284
librpcsoc(3LIBUCB) 286
librpcsvc(3LIB) 287
librsm(3LIB) 288
librt(3LIB) 290
librtld_db(3LIB) 292
libsasl(3LIB) 293
libscf(3LIB) 295
libsctp(3LIB) 299

libsec(3LIB) 300
libsecdb(3LIB) 301
libsendfile(3LIB) 303
libslp(3LIB) 304
libsmartcard(3LIB) 305
libsocket(3LIB) 306
libssagent(3LIB) 308
libssasnmp(3LIB) 309
libsys(3LIB) 310
libsysevent(3LIB) 315
libtecla(3LIB) 316
libtermcap(3LIBUCB) 319
libthread(3LIB) 320
libtnfctl(3LIB) 322
libucb(3LIBUCB) 324
libumem(3LIB) 326
libusb(3LIB) 327
libuuid(3LIB) 329
libvolmgt(3LIB) 330
libw(3LIB) 331
libwsreg(3LIB) 333
libxnet(3LIB) 335
liby(3LIB) 338
limits.h(3HEAD) 339
locale.h(3HEAD) 349
math.h(3HEAD) 351
mman.h(3HEAD) 354
monetary.h(3HEAD) 356
mqueue.h(3HEAD) 357
msg.h(3HEAD) 358
ndbm.h(3HEAD) 359
netdb.h(3HEAD) 360
nl_types.h(3HEAD) 362
poll.h(3HEAD) 363
pthread.h(3HEAD) 364
pwd.h(3HEAD) 366
regex.h(3HEAD) 367
resource.h(3HEAD) 369

sched.h(3HEAD) 371
search.h(3HEAD) 372
select.h(3HEAD) 373
semaphore.h(3HEAD) 374
sem.h(3HEAD) 375
setjmp.h(3HEAD) 377
shm.h(3HEAD) 378
siginfo.h(3HEAD) 379
signal.h(3HEAD) 383
socket.h(3HEAD) 390
spawn.h(3HEAD) 394
stat.h(3HEAD) 395
statvfs.h(3HEAD) 397
stdbool.h(3HEAD) 398
stddef.h(3HEAD) 399
stdint.h(3HEAD) 400
stdio.h(3HEAD) 407
stdlib.h(3HEAD) 409
string.h(3HEAD) 411
strings.h(3HEAD) 412
stropts.h(3HEAD) 413
syslog.h(3HEAD) 418
tar.h(3HEAD) 420
tcp.h(3HEAD) 422
termios.h(3HEAD) 423
tgmath.h(3HEAD) 428
timeb.h(3HEAD) 432
time.h(3HEAD) 433
times.h(3HEAD) 435
types32.h(3HEAD) 436
types.h(3HEAD) 437
ucontext.h(3HEAD) 440
uio.h(3HEAD) 441
ulimit.h(3HEAD) 442
un.h(3HEAD) 443
unistd.h(3HEAD) 444
utime.h(3HEAD) 454
utmpx.h(3HEAD) 455

utsname.h(3HEAD) 456
values.h(3HEAD) 457
wait.h(3HEAD) 459
wchar.h(3HEAD) 461
wctype.h(3HEAD) 463
wordexp.h(3HEAD) 464

Index 467

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.

(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`. See `libc`(3LIB) 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).

The `libpthread` and `libthread` libraries are filter libraries on `libc` that are used for building multithreaded applications: `libpthread` implements the POSIX (see `standards`(5)) threads interface, whereas `libthread` implements the Solaris threads interface. See `MULTITHREADED APPLICATIONS`, below.

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

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

(3UCB) These functions constitute the Source Compatibility (with BSD functions) library. It is implemented as a shared object, `libucb.so`, 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`, 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`, 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`, 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`, 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. 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`, but it is not automatically linked by the C compilation system. See `libslp`(3LIB).

(3SASL) These functions constitute the Simple Authentication and Security Layer (SASL) library, `libsasl`. SASL is a security framework used by connection-oriented network applications primarily for authentication. Another way to describe SASL is that it is a glue layer between a network application and some security mechanisms that allow applications to authenticate each other and provide additional security services such as data encryption. As a glue layer, SASL hides the interface specifics of the security mechanism from the application, which allows greater portability and flexibility as new security mechanisms are implemented.

`libsasl` provides both an API for applications and an SPI for various plug-ins. To link with this library, specify `-lsasl` on the `cc` command line. See `libsasl`(3LIB).

(3SOCKET) These functions constitute the sockets library, `libsocket`. This library is implemented as a shared object, `libsocket.so`, 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).

(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`, 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`, 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`, 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`, 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`, 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`. 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.

Realtime Library Functions

The functions described in this volume constitute the 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).

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`, 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).

(3CONTRACT) These functions constitute the contract managementlibrary, `libcontract`. This library is implemented as a shared object, `libcontract.so`, but is not automatically linked by the C compilation system. Specify `-lcontract` on the `cc` command line to link with this library. See `libcontract`(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).

(3DAT) These functions constitute the direct access transport library, `libdevid`. This library is implemented as a shared object, `libdat.so`, but is not automatically linked by the C compilation system. Specify `-ldat` on the `cc` command line to link with this library. See `libdat`(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`, 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`, 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).
(3HBAAPI)	These functions constitute the common fibre channel HBA information library, `libhbaapi`. This library is implemented as a shared object, `libhbaapi.so`, but is not automatically linked by the C compilation system. Specify `-lhbaapi` on the `cc` command line to link with this library. See `libhbaapi`(3LIB).
(3KSTAT)	These functions constitute the kernel statistics library, which is implemented as a shared object, `libkstat.so`, 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`, 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).

(3LGRP) These functions constitute the locality group library, which is implemented as a shared object, `liblgrp.so`, but is not automatically linked by the C compilation system. Specify `-llgrp` on the `cc` command line to link with this library. See `liblgrp`(3LIB).

(3M) These functions constitute the mathematical library, `libm`. This library is implemented as a shared object, `libm.so`, 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`, 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`, 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`, 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).
(3SCF)	These functions constitute the object-caching memory allocation library, `libscf`. This library is implemented as a shared object, `libscf.so`, but is not automatically linked by the C compilation system. Specify `-lscf` on the `cc` command line to link with this library. See `libscf`(3LIB).
(3SEC)	These functions constitute the file access control library, `libsec`. This library is implemented as a shared object, `libsec.so`, 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).
(3SMARTCARD)	These functions constitute the smartcard library, `libsmartcard`. This library is implemented as a shared object, `libsmartcard.so`, but is not automatically linked by the C compilation system. Specify `-lsmartcard` on the `cc` command line to link with this library. See `libsmartcard`(3LIB).
(3SNMP)	These functions constitute the SNMP libraries, `libssagent` and `libssasnmp`. 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).

(3TECLA) These functions constitute the interactive command-line input library, `libtecla`. This library is implemented as a shared object, `libtecla.so`, but is not automatically linked by the C compilation system. Specify `-ltecla` on the `cc` command line to link with this library. See `libtecla`(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).

(3UUID) These functions constitute the universally unique identifier library, `libuuid`. This library is implemented as a shared object, `libuuid.so`, but is not automatically linked by the C compilation system. Specify `-luuid` on the `cc` command line to link with this library. See `libuuid`(3LIB).

(3VOLMGT) These functions constitute the volume management library, `libvolmgt`. This library is implemented as a shared object, `libvolmgt.so`, 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).

Multimedia Library Functions

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

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.

The `libpthread`(3LIB) and `libthread`(3LIB) libraries are implemented as filters on `libc`(3LIB).

When compiling a multithreaded application, the `-mt` option must be specified on the command line.

There is no need for a multithreaded application to link with `-lthread`. An application must link with `-lpthread` only when POSIX semantics for `fork`(2) are desired. When an application is linked with `-lpthread`, a call to `fork()` assumes the behavior `fork1`(2) rather than the default behavior that forks all threads.

When compiling a POSIX-conforming application, either the `_POSIX_C_SOURCE` or `_POSIX_PTHREAD_SEMANTICS` option must be specified on the command line. For POSIX.1c-conforming applications, define the `_POSIX_C_SOURCE` flag to be >= 199506L:

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

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

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

For Solaris threads behavior, compile as follows:

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

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

The environment variable `LD_BIND_NOW` must be set 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.

FILES

INCDIR	usually `/usr/include`
LIBDIR	usually either `/lib` or `/usr/lib` (32–bit) or either `/lib/64` or `/usr/lib/64` (64–bit)
LIBDIR`/*.so`	shared libraries

SEE ALSO

`ar`(1), `cc`(1B), `ld`(1), `fork`(2), `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

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.h, 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.h, 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_ALLDONE`	A return value indicating that none of the requested operations could be canceled since they are already complete.
`AIO_CANCELED`	A return value indicating that all requested operations have been canceled.
`AIO_NOTCANCELED`	A return value indicating that some of the requested operations could not be canceled since they are in progress.
`LIO_NOP`	A `lio_listio`(3RT) element operation option indicating that no transfer is requested.
`LIO_NOWAIT`	A `lio_listio()` synchronization operation indicating that the calling thread is to continue execution while the `lio_listio()` operation is being performed, and no notification is given when the operation is complete.
`LIO_READ`	A `lio_listio()` element operation option requesting a read.
`LIO_WAIT`	A `lio_listio()` synchronization operation indicating that the calling thread is to suspend until the `lio_listio()` operation is complete.
`LIO_WRITE`	A `lio_listio()` element operation option requesting a write.

SEE ALSO

`lseek`(2), `read`(2), `write`(2), `fsync`(3C), `libaio`(3LIB), `lio_listio`(3RT)

NAME

ar.h, 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

assert.h, assert – verify program assertion

SYNOPSIS

```
#include <assert.h>
```

DESCRIPTION

The `<assert.h>` header defines the `assert()` macro. It refers to the macro `NDEBUG` which is not defined in the header. If `NDEBUG` is defined as a macro name before the inclusion of this header, the `assert()` macro is defined simply as:

```
#define assert(ignore)((void) 0)
```

Otherwise, the macro behaves as described in `assert`(3C).

The `assert()` macro is redefined according to the current state of `NDEBUG` each time `<assert.h>` is included.

The `assert()` macro is implemented as a macro, not as a function. If the macro definition is suppressed in order to access an actual function, the behavior is undefined.

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Standard

SEE ALSO

`assert`(3C), `attributes`(5), `standards`(5)

NAME

complex.h, complex – complex arithmetic

SYNOPSIS

```
#include <complex.h>
```

DESCRIPTION

The <complex.h> header defines the following macros:

complex	Expands to _Complex.
_Complex_I	Expands to a constant expression of type const float _Complex, with the value of the imaginary unit (that is, a number i such that $i^2=-1$).
imaginary	Expands to _Imaginary.
_Imaginary_I	Expands to a constant expression of type const float _Imaginary with the value of the imaginary unit.
I	Expands to either _Imaginary_I or _Complex_I. If _Imaginary_I is not defined, I expands to _Complex_I.

An application can undefine and then, if appropriate, redefine the complex, imaginary, and I macros.

USAGE

Values are interpreted as radians, not degrees.

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Standard

SEE ALSO

cabs(3M), cacos(3M), cacosh(3M), carg(3M), casin(3M), casinh(3M), catan(3M), catanh(3M), ccos(3M), ccosh(3M), cexp(3M), cimag(3M), clog(3M), conj(3M), cpow(3M), cproj(3M), creal(3M), csin(3M), csinh(3M), csqrt(3M), ctan(3M), ctanh(3M), attributes(5), standards(5)

NOTES

The choice of I instead of i for the imaginary unit concedes to the widespread use of the identifier i for other purposes. The application can use a different identifier, say j, for the imaginary unit by following the inclusion of the <complex.h> header with:

```
#undef I
#define j _Imaginary_I
```

An I suffix to designate imaginary constants is not required, as multiplication by I provides a sufficiently convenient and more generally useful notation for imaginary terms. The corresponding real type for the imaginary unit is float, so that use of I for algorithmic or notational convenience does not result in widening types.

On systems with imaginary types, the application has the ability to control whether use of the macro I introduces an imaginary type, by explicitly defining I to be _Imaginary_I or _Complex_I.

Disallowing imaginary types is useful for some applications intended to run on implementations without support for such types.

The macro `_Imaginary_I` provides a test for whether imaginary types are supported. The `cis()` function `(cos(x) + I*sin(x))` was considered but rejected because its implementation is easy and straightforward, even though some implementations could compute sine and cosine more efficiently in tandem.

NAME

cpio.h, cpio – cpio archive values

SYNOPSIS

```
#include <cpio.h>
```

DESCRIPTION

Values needed by the `c_mode` field of the `cpio` archive format are described as follows:

Name	Description
C_IRUSR	Read by owner
C_IWUSR	Write by owner
C_IXUSR	Execute by owner
C_IRGRP	Read by group
C_IWGRP	Write by group
C_IXGRP	Execute by group
C_IROTH	Read by others
C_IWOTH	Write by others
C_IXOTH	Execute by others
C_ISUID	Set user ID
C_ISGID	Set group ID
C_ISVTX	On directories, restricted deletion flag
C_ISDIR	Directory
C_ISFIFO	FIFO
C_ISREG	Regular file
C_ISBLK	Block special
C_ISCHR	Character special
C_ISCTG	Reserved
C_ISLNK	Symbolic link
C_ISSOCK	Socket

The header defines the symbolic constant:

```
MAGIC         "070707"
```

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Standard

SEE ALSO

`pax`(1), `attributes`(5), `standards`(5)

NAME

dirent.h, dirent – format of directory entries

SYNOPSIS

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

DESCRIPTION

The internal format of directories is unspecified. The `<dirent.h>` header defines the following type:

`DIR` A type representing a directory stream.

The header also defines the structure `dirent`, which includes the following members:

```
ino_t d_ino        /* file serial number */
char  d_name[]     /* name of entry */
```

The type `ino_t` is defined as described in `<sys/types.h>`. See types(3HEAD).

The character array `d_name` is of unspecified size, but the number of bytes preceding the terminating null byte must not exceed {`NAME_MAX`}.

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Standard

SEE ALSO

`closedir`(3C), `opendir`(3C), `readdir`(3C), `rewinddir`(3C), `seekdir`(3C), `telldir`(3C), types.h(3HEAD), `attributes`(5), `standards`(5)

NAME

errno.h, errno – system error numbers

SYNOPSIS

```
#include <errno.h>
```

DESCRIPTION

The `<errno.h>` header provides a declaration for `errno` and gives positive values for the symbolic constants listed on the `intro`(2) manual page.

USAGE

Values for `errno` are required to be distinct positive values rather than non-zero values.

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Standard

SEE ALSO

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

NAME

fcntl.h, 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 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 */
```

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Standard

SEE ALSO

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

fenv.h, fenv – floating-point environment

SYNOPSIS

```
#include <fenv.h>
```

DESCRIPTION

The <fenv.h> header defines the following data types through typedef:

`fenv_t` Represents the entire floating-point environment. The floating-point environment refers collectively to any floating-point status flags and control modes supported by the implementation.

`fexcept_t` Represents the floating-point status flags collectively, including any status the implementation associates with the flags. A floating-point status flag is a system variable whose value is set (but never cleared) when a floating-point exception is raised, which occurs as a side effect of exceptional floating-point arithmetic to provide auxiliary information. A floating-point control mode is a system variable whose value can be set by the user to affect the subsequent behavior of floating-point arithmetic.

The <fenv.h> header defines the following constants if and only if the implementation supports the floating-point exception by means of the floating-point functions feclearexcept(), fegetexceptflag(), feraiseexcept(), fesetexceptflag(), and fetestexcept(). Each expands to an integer constant expression with values such that bitwise-inclusive ORs of all combinations of the constants result in distinct values.

```
FE_DIVBYZERO
FE_INEXACT
FE_INVALID
FE_OVERFLOW
FE_UNDERFLOW
```

The <fenv.h> header defines the following constant, which is simply the bitwise-inclusive OR of all floating-point exception constants defined above:

```
FE_ALL_EXCEPT
```

The <fenv.h> header defines the following constants. Each expands to an integer constant expression whose values are distinct non-negative values.

```
FE_DOWNWARD
FE_TONEAREST
FE_TOWARDZERO
FE_UPWARD
```

The <fenv.h> header defines the following constant, which represents the default floating-point environment (that is, the one installed at program startup) and has type pointer to const-qualified fenv_t. It can be used as an argument to the functions within the <fenv.h> header that manage the floating-point environment.

```
FE_DFL_ENV
```

The FENV_ACCESS pragma provides a means to inform the implementation when an application might access the floating-point environment to test floating-point status flags or run under non-default floating-point control modes. The pragma occurs either outside external declarations or preceding all explicit declarations and statements inside a compound statement. When outside external declarations, the pragma takes effect from its occurrence until another FENV_ACCESS pragma is encountered, or until the end of the translation unit. When inside a compound statement, the pragma takes effect from its occurrence until another FENV_ACCESS pragma is encountered (including within a nested compound statement), or until the end of the compound statement; at the end of a compound statement the state for the pragma is restored to its condition just before the compound statement. If this pragma is used in any other context, the behavior is undefined.

If part of an application tests floating-point status flags, sets floating-point control modes, or runs under non-default mode settings, but was translated with the state for the FENV_ ACCESS pragma off, the behavior is undefined. The default state (on or off) for the pragma is implementation-defined. (When execution passes from a part of the application translated with FENV_ACCESS off to a part translated with FENV_ACCESS on, the state of the floating-point status flags is unspecified and the floating-point control modes have their default settings.)

USAGE

This header is designed to support the floating-point exception status flags and directed-rounding control modes required by the IEC 60559: 1989 standard, and other similar floating-point state information. Also, it is designed to facilitate code portability among all systems. Certain application programming conventions support the intended model of use for the floating-point environment:

- A function call does not alter its caller's floating-point control modes, clear its caller's floating-point status flags, or depend on the state of its caller's floating-point status flags unless the function is so documented.
- A function call is assumed to require default floating-point control modes, unless its documentation promises otherwise.
- A function call is assumed to have the potential for raising floating-point exceptions, unless its documentation promises otherwise.

With these conventions, an application can safely assume default floating-point control modes (or be unaware of them). The responsibilities associated with accessing the floating-point environment fall on the application that does so explicitly.

Even though the rounding direction macros might expand to constants corresponding to the values of FLT_ROUNDS, they are not required to do so. For example:

```
#include <fenv.h>
void f(double x)
{
    #pragma STDC FENV_ACCESS ON
    void g(double);
    void h(double);
    /* ... */
```

```
    g(x + 1);
    h(x + 1);
    /* ... */
}
```

If the function *g*() might depend on status flags set as a side effect of the first *x*+1, or if the second *x*+1 might depend on control modes set as a side effect of the call to function *g*(), then the application must contain an appropriately placed invocation as follows:

```
#pragma STDC FENV_ACCESS ON
```

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Standard

SEE ALSO

`feclearexcept`(3M), `fegetenv`(3M), `fegetexceptflag`(3M), `fegetround`(3M), `feholdexcept`(3M), `feraiseexcept`(3M), `fesetenv`(3M), `fesetexceptflag`(3M), `fesetround`(3M), `fetestexcept`(3M), `feupdateenv`(3M), `attributes`(5), `standards`(5)

NAME

float.h, float – floating types

SYNOPSIS

```
#include <float.h>
```

DESCRIPTION

The characteristics of floating types are defined in terms of a model that describes a representation of floating-point numbers and values that provide information about an implementation's floating-point arithmetic.

The following parameters are used to define the model for each floating-point type:

s	sign (±1)
b	base or radix of exponent representation (an integer >1)
e	exponent (an integer between a minimum e_{min} and a maximum e_{max})
p	precision (the number of base-*b* digits in the significand)
f_k	non-negative integers less than *b* (the significand digits)

In addition to normalized floating-point numbers (f_1>0 if $x \neq 0$), floating types might be able to contain other kinds of floating-point numbers, such as subnormal floating-point numbers ($x \neq 0$, $e=e_{min}$, $f_1=0$) and unnormalized floating-point numbers ($x \neq 0$, $e=e_{min}$, $f_1=0$), and values that are not floating-point numbers, such as infinities and NaNs. A *NaN* is an encoding signifying Not-a-Number. A *quiet NaN* propagates through almost every arithmetic operation without raising a floating-point exception; a *signaling NaN* generally raises a floating-point exception when occurring as an arithmetic operand.

The accuracy of the library functions in `math.h`(3HEAD) and `complex.h`(3HEAD) that return floating-point results is defined on the `libm`(3LIB) manual page.

All integer values in the `<float.h>` header, except `FLT_ROUNDS`, are constant expressions suitable for use in `#if` preprocessing directives; all floating values are constant expressions. All except `DECIMAL_DIG`, `FLT_EVAL_METHOD`, `FLT_RADIX`, and `FLT_ROUNDS` have separate names for all three floating-point types. The floating-point model representation is provided for all values except `FLT_EVAL_METHOD` and `FLT_ROUNDS`.

The rounding mode for floating-point addition is characterized by the value of `FLT_ROUNDS`:

`-1`	Indeterminable.
`0`	Toward zero.
`1`	To nearest.
`2`	Toward positive infinity.
`3`	Toward negative infinity.

The values of operations with floating operands and values subject to the usual arithmetic conversions and of floating constants are evaluated to a format whose range and precision might be greater than required by the type. The use of evaluation formats is characterized by the architecture-dependent value of `FLT_EVAL_METHOD`:

`-1` Indeterminable.

`0` Evaluate all operations and constants just to the range and precision of the type.

`1` Evaluate operations and constants of type float and double to the range and precision of the double type; evaluate long double operations and constants to the range and precision of the long double type.

`2` Evaluate all operations and constants to the range and precision of the long double type.

The values given in the following list are defined as constants.

- Radix of exponent representation, *b*.

  ```
  FLT_RADIX
  ```

- Number of base-`FLT_RADIX` digits in the floating-point significand, *p*.

  ```
  FLT_MANT_DIG
  DBL_MANT_DIG
  LDBL_MANT_DIG
  ```

- Number of decimal digits, *n*, such that any floating-point number in the widest supported floating type with p_{max} radix *b* digits can be rounded to a floating-point number with *n* decimal digits and back again without change to the value.

  ```
  DECIMAL_DIG
  ```

- Number of decimal digits, *q*, such that any floating-point number with *q* decimal digits can be rounded into a floating-point number with *p* radix *b* digits and back again without change to the *q* decimal digits.

  ```
  FLT_DIG
  DBL_DIG
  LDBL_DIG
  ```

- Minimum negative integer such that `FLT_RADIX` raised to that power minus 1 is a normalized floating-point number, e_{min}.

  ```
  FLT_MIN_EXP
  DBL_MIN_EXP
  LDBL_MIN_EXP
  ```

- Minimum negative integer such that 10 raised to that power is in the range of normalized floating-point numbers.

  ```
  FLT_MIN_10_EXP
  DBL_MIN_10_EXP
  LDBL_MIN_10_EXP
  ```

- Maximum integer such that `FLT_RADIX` raised to that power minus 1 is a representable finite floating-point number, e_{max}.

  ```
  FLT_MAX_EXP
  DBL_MAX_EXP
  LDBL_MAX_EXP
  ```

- Maximum integer such that 10 raised to that power is in the range of representable finite floating-point numbers.

  ```
  FLT_MAX_10_EXP
  DBL_MAX_10_EXP
  LDBL_MAX_10_EXP
  ```

The values given in the following list are defined as constant expressions with values that are greater than or equal to those shown:

- Maximum representable finite floating-point number.

  ```
  FLT_MAX
  DBL_MAX
  LDBL_MAX
  ```

The values given in the following list are defined as constant expressions with implementation-defined (positive) values that are less than or equal to those shown:

- The difference between 1 and the least value greater than 1 that is representable in the given floating-point type, b^{1-p}.

  ```
  FLT_EPSILON
  DBL_EPSILON
  LDBL_EPSILON
  ```

- Minimum normalized positive floating-point number, $b^{e_{min}-1}$.

  ```
  FLT_MIN
  DBL_MIN
  LDBL_MIN
  ```

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Standard

SEE ALSO

`complex.h`(3HEAD), `math.h`(3HEAD), `attributes`(5), `standards`(5)

NAME

floatingpoint.h, 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

fmtmsg.h, fmtmsg – message display structures

SYNOPSIS

```
#include <fmtmsg.h>
```

DESCRIPTION

The `<fmtmsg.h>` header defines the following macros, which expand to constant integer expressions:

`MM_HARD`	Source of the condition is hardware.
`MM_SOFT`	Source of the condition is software.
`MM_FIRM`	Source of the condition is firmware.
`MM_APPL`	Condition detected by application.
`MM_UTIL`	Condition detected by utility.
`MM_OPSYS`	Condition detected by operating system.
`MM_RECOVER`	Recoverable error.
`MM_NRECOV`	Non-recoverable error.
`MM_HALT`	Error causing application to halt.
`MM_ERROR`	Application has encountered a non-fatal fault.
`MM_WARNING`	Application has detected unusual non-error condition.
`MM_INFO`	Informative message.
`MM_NOSEV`	No severity level provided for the message.
`MM_PRINT`	Display message on standard error.
`MM_CONSOLE`	Display message on system console.

The table below indicates the null values and identifiers for `fmtmsg`(3C) arguments. The `<fmtmsg.h>` header defines the macros in the Identifier column, which expand to constant expressions that expand to expressions of the type indicated in the Type column:

Argument	Type	Null-Value	Identifier
label	`char*`	`(char*) NULL`	`MM_NULLLBL`
severity	`int`	`0`	`MM_NULLSEV`
class	`long`	`0L`	`MM_NULLMC`
text	`char*`	`(char*) NULL`	`MM_NULLTXT`
action	`char*`	`(char*) NULL`	`MM_NULLACT`
tag	`char*`	`(char*) NULL`	`MM_NULLTAG`

The `<fmtmsg.h>` header also defines the following macros for use as return values for `fmtmsg()`:

`MM_OK`	The function succeeded.
`MM_NOTOK`	The function failed completely.
`MM_NOMSG`	The function was unable to generate a message on standard error, but otherwise succeeded.
`MM_NOCON`	The function was unable to generate a console message, but otherwise succeeded.

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Standard

SEE ALSO

`fmtmsg`(3C), `attributes`(5), `standards`(5)

NAME

fnmatch.h, fnmatch – filename-matching types

SYNOPSIS

```
#include <fnmatch.h>
```

DESCRIPTION

The `<fnmatch.h>` header defines the following constants:

`FNM_NOMATCH`	The string does not match the specified pattern.
`FNM_PATHNAME`	Slash in string only matches slash in pattern.
`FNM_PERIOD`	Leading period in string must be exactly matched by period in pattern.
`FNM_NOESCAPE`	Disable backslash escaping.
`FNM_NOSYS`	Reserved.

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Standard

SEE ALSO

`fnmatch`(3C), `attributes`(5), `standards`(5)

NAME

ftw.h, ftw – file tree traversal

SYNOPSIS

```
#include <ftw.h>
```

DESCRIPTION

The <ftw.h> header defines the FTW structure that includes the following members:

```
int base
int level
```

The <ftw.h> header defines macros for use as values of the third argument to the application-supplied function that is passed as the second argument to ftw() and nftw() (see ftw(3C)):

FTW_F	file
FTW_D	directory
FTW_DNR	directory without read permission
FTW_DP	directory with subdirectories visited
FTW_NS	unknown type; stat() failed
FTW_SL	symbolic link
FTW_SLN	symbolic link that names a nonexistent file

The <ftw.h> header defines macros for use as values of the fourth argument to nftw():

FTW_PHYS	Physical walk, does not follow symbolic links. Otherwise, nftw() follows links but does not walk down any path that crosses itself.
FTW_MOUNT	The walk does not cross a mount point.
FTW_DEPTH	All subdirectories are visited before the directory itself.
FTW_CHDIR	The walk changes to each direct ory before reading it.

The <ftw.h> header defines the stat structure and the symbolic names for st_mode and the file type test macros as described in <sys/stat.h>.

Inclusion of the <ftw.h> header might also make visible all symbols from <sys/stat.h>.

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Standard

SEE ALSO

ftw(3C), stat.h(3HEAD), attributes(5), standards(5)

NAME

glob.h, glob – pathname pattern-matching types

SYNOPSIS

```
#include <glob.h>
```

DESCRIPTION

The <glob.h> header defines the structures and symbolic constants used by the glob(3C).

The structure type glob_t contains the following members:

```
size_t gl_pathc       /* count of paths matched by pattern */
char   **gl_pathv     /* pointer to a list of matched pathnames */
size_t gl_offs        /* lots to reserve at the beginning of gl_pathv */
```

The following constants are provided as values for the flags argument:

GLOB_APPEND — Append generated pathnames to those previously obtained.

GLOB_DOOFFS — Specify how many null pointers to add to the beginning of gl_pathv.

GLOB_ERR — Cause glob() to return on error.

GLOB_MARK — Each pathname that is a directory that matches pattern has a slash appended.

GLOB_NOCHECK — If pattern does not match any pathname, then return a list consisting of only pattern.

GLOB_NOESCAPE — Disable backslash escaping.

GLOB_NOSORT — Do not sort the pathnames returned.

The following constants are defined as error return values:

GLOB_ABORTED — The scan was stopped because GLOB_ERR was set or (*errfunc)() returned non-zero.

GLOB_NOMATCH — The pattern does not match any existing pathname, and GLOB_NOCHECK was not set in flags.

GLOB_NOSPACE — An attempt to allocate memory failed.

GLOB_NOSYS — Reserved.

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Standard

SEE ALSO

glob(3C), attributes(5), standards(5)

NAME

grp.h, grp – group structure

SYNOPSIS

```
#include <grp.h>
```

DESCRIPTION

The <grp.h> header declares the structure group, which includes the following members:

```
char  *gr_name     /* name of the group */
gid_t gr_gid       /* numerical group ID */
char  **gr_mem     /* pointer to a null-terminated array of character */
                   /* pointers to member names */
```

The gid_t type is defined as described in <sys/types.h> (see types(3HEAD)).

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Standard

SEE ALSO

getgrnam(3C), types.h(3HEAD), attributes(5), standards(5)

NAME

iconv.h, iconv – codeset conversion facility

SYNOPSIS

```
#include <iconv.h>
```

DESCRIPTION

The `<iconv.h>` header defines the following type:

`iconv_t` Identifies the conversion from one codeset to another.

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Standard

SEE ALSO

`iconv`(3C), `iconv_close`(3C), `iconv_open`(3C), `attributes`(5), `standards`(5)

NAME

if.h, if – sockets local interfaces

SYNOPSIS

```
#include <net/if.h>
```

DESCRIPTION

The <net/if.h> header defines the if_nameindex structure, which includes the following members:

```
unsigned if_index     /* numeric index of the interface */
char     *if_name     /* null-terminated name of the interface */
```

The <net/if.h> header defines the following macro for the length of a buffer containing an interface name (including the terminating null character):

IF_NAMESIZE interface name length

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Standard

SEE ALSO

if_nametoindex(3XNET), attributes(5), standards(5)

NAME

inet.h, 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.h`(3HEAD).

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

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Standard

SEE ALSO

`Intro`(3), `htonl`(3SOCKET), `htonl`(3XNET), `inet_addr`(3SOCKET), `inet_addr`(3XNET), `in.h`(3HEAD), `attributes`(5), `standards`(5)

NAME

in.h, 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.h(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]
```

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Standard

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.h(3HEAD), attributes(5), standards(5)

NAME

inttypes.h, inttypes – fixed size integer types

SYNOPSIS

```
#include <inttypes.h>
```

DESCRIPTION

The <inttypes.h> header includes the <stdint.h> header.

The <inttypes.h> header includes a definition of the following type:

`imaxdiv_t` structure type that is the type of the value returned by the `imaxdiv()` function.

The following macros are defined. Each expands to a character string literal containing a conversion specifier, possibly modified by a length modifier, suitable for use within the format argument of a formatted input/output function when converting the corresponding integer type. These macros have the general form of `PRI` (character string literals for the `fprintf()` and `fwprintf()` family of functions) or `SCN` (character string literals for the `fscanf()` and `fwscanf()` family of functions), followed by the conversion specifier, followed by a name corresponding to a similar type name in <stdint.h>. In these names, *N* represents the width of the type as described in <stdint.h>. For example, `PRIdFAST32` can be used in a format string to print the value of an integer of type `int_fast32_t`.

The `fprintf()` macros for signed integers are:

`PRId`*N* `PRIdLEAST`*N* `PRIdFAST`*N* `PRIdMAX` `PRIdPTR`
`PRIi`*N* `PRIiLEAST`*N* `PRIiFAST`*N* `PRIiMAX` `PRIiPTR`

The `fprintf()` macros for unsigned integers are:

`PRIo`*N* `PRIoLEAST`*N* `PRIoFAST`*N* `PRIoMAX` `PRIoPTR`
`PRIu`*N* `PRIuLEAST`*N* `PRIuFAST`*N* `PRIuMAX` `PRIuPTR`
`PRIx`*N* `PRIxLEAST`*N* `PRIxFAST`*N* `PRIxMAX` `PRIxPTR`
`PRIX`*N* `PRIXLEAST`*N* `PRIXFAST`*N* `PRIXMAX` `PRIXPTR`

The `fscanf()` macros for signed integers are:

`SCNd`*N* `SCNdLEAST`*N* `SCNdFAST`*N* `SCNdMAX` `SCNdPTR`
`SCNi`*N* `SCNiLEAST`*N* `SCNiFAST`*N* `SCNiMAX` `SCNiPTR`

The `fscanf()` macros for unsigned integers are:

`SCNo`*N* `SCNoLEAST`*N* `SCNoFAST`*N* `SCNoMAX` `SCNoPTR`
`SCNu`*N* `SCNuLEAST`*N* `SCNuFAST`*N* `SCNuMAX` `SCNuPTR`
`SCNx`*N* `SCNxLEAST`*N* `SCNxFAST`*N* `SCNxMAX` `SCNxPTR`

For each type that the implementation provides in <stdint.h>, the corresponding `fprintf()` and `fwprintf()` macros must be defined. The corresponding `fscanf()` and `fwscanf()` macros must be defined as well, unless the implementation does not have a suitable modifier for the type.

USAGE

The purpose of `<inttypes.h>` is to provide a set of integer types whose definitions are consistent across machines and independent of operating systems and other implementation idiosyncrasies. It defines, with a `typedef`, integer types of various sizes. Implementations are free to `typedef` them as ISO C standard integer types or extensions that they support. Consistent use of this header greatly increases the portability of applications across platforms.

EXAMPLES

EXAMPLE 1 Use of Macro

The following code uses one of the macros available through `<inttypes.h>`.

```
#include <inttypes.h>
#include <wchar.h>
int main(void)
{
     uintmax_t i = UINTMAX_MAX; // This type always exists.
     wprintf("The largest integer value is %020"
          PRIxMAX, "\n", i);
     return 0;
}
```

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Standard

SEE ALSO

`imaxdiv`(3C), `attributes`(5), `standards`(5)

NAME

ipc.h, ipc – XSI interprocess communication access structure

SYNOPSIS

```
#include <sys/ipc.h>
```

DESCRIPTION

The `<sys/ipc.h>` header is used by three mechanisms for interprocess communication (IPC): messages, semaphores, and shared memory. All use a common structure type, `ipc_perm`, to pass information used in determining permission to perform an IPC operation.

The `ipc_perm` structure contains the following members:

```
uid_t  uid     /* owner's user ID */
gid_t  gid     /* owner's group ID */
uid_t  cuid    /* creator's user ID */
gid_t  cgid    /* creator's group ID */
mode_t mode    /* read/write permission
```

The `uid_t`, `gid_t`, `mode_t`, and `key_t` types are defined as described in `<sys/types.h>`. See `types.h`(3HEAD).

Definitions are provided for the constants listed below.

Mode bits:

`IPC_CREAT`	Create entry if key does not exist.
`IPC_EXCL`	Fail if key exists.
`IPC_NOWAIT`	Error if request must wait.

Keys:

`IPC_PRIVATE`	Private key.

Control commands:

`IPC_RMID`	Remove identifier.
`IPC_SET`	Set options.
`IPC_STAT`	Get options.

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Standard

SEE ALSO

`ftok`(3C), `types.h`(3HEAD), `attributes`(5), `standards`(5)

NAME

iso646.h, iso646 – alternative spellings

SYNOPSIS

```
#include <iso646.h>
```

DESCRIPTION

The `<iso646.h>` header defines the following macros (on the left) that expand to the corresponding tokens (on the right):

```
and             &&
and_eq          &=
bitand          &
bitor           |
compl           ~
not             !
not_eq          !=
or              ||
or_eq           |=
xor             ^
xor_eq          ^=
```

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Standard

SEE ALSO

`attributes`(5), `standards`(5)

NAME

langinfo.h, 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`.

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Standard

SEE ALSO

`mkmsgs`(1), `localeconv`(3C), `nl_langinfo`(3C), `nl_types.h`(3HEAD), `setlocale`(3C), `strftime`(3C), `attributes`(5), `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.

`circf`	`loc1`
`loc2`	`locs`
`nbra`	`pkgdir`
`read_vtoc`	`sed`
`write_vtoc`	

FILES

`/lib/libadm.so.1`	shared object
`/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 (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 perform asynchronous I/O operations.

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`

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

`aioread64`	`aiowrite64`

FILES

`/lib/libaio.so.1`	shared object
`/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), `aio.h`(3HEAD), `attributes`(5)

NAME

libauto_ef – auto encoding finder library

SYNOPSIS

```
cc [ flag... ] file... -lauto_ef [ library... ]
#include <auto_ef.h>
```

DESCRIPTION

Functions in this library provide automatic encoding identification.

INTERFACE LEVEL

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

`auto_ef_file`	`auto_ef_free`
`auto_ef_get_encoding`	`auto_ef_get_score`
`auto_ef_str`	

FILES

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

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWautoef (32-bit) SUNWautoefx (64-bit)
Interface Stability	Stable
MT-Level	MT-Safe

SEE ALSO

`auto_ef`(1), `auto_ef`(3EXT), `attributes`(5)

International Language Environments Guide

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.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 (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_process_ex	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

/lib/libbsm.so.1	shared object
/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 (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

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, as are the facilities previously defined in the multithreading libraries, `libthread` and `libpthread`.

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`	`_Exit`
`_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_info	_lwp_kill

_lwp_mutex_lock	_lwp_mutex_trylock
_lwp_mutex_unlock	_lwp_self
_lwp_sema_init	_lwp_sema_post
_lwp_sema_trywait	_lwp_sema_wait
_lwp_suspend	_lwp_suspend2
_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	_stack_grow
_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
alphasort	altzone
ascftime	asctime
asctime_r	atexit
atof	atoi
atol	atoll
atomic_add_16	atomic_add_16_nv
atomic_add_32	atomic_add_32_nv
atomic_add_64	atomic_add_64_nv
atomic_add_long	atomic_add_long_nv
atomic_and_32	atomic_and_uint
atomic_or_32	atomic_or_uint
attropen	basename
bcmp	bcopy
bindtextdomain	bind_textdomain_codeset
brk	bsd_signal
bsearch	btowc
bzero	calloc
catclose	catgets
catopen	cfgetispeed
cfgetospeed	cfsetispeed
cfsetospeed	cftime
chdir	chmod

chown	chroot
clearerr	clock
close	closedir
closefrom	closelog
cond_broadcast	cond_destroy
cond_init	cond_reltimedwait
cond_signal	cond_timedwait
cond_wait	confstr
creat	crypt
crypt_genhash_impl	crypt_gensalt
crypt_gensalt_impl	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
dladdr	dladdr1
dlclose	dldump
dlerror	dlinfo
dlmopen	dlopen
dlsym	dngettext

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	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	forkall
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
getisax	getitimer
getloadavg	getlogin
getlogin_r	getmntany
getmntent	getmsg
getnetgrent	getnetgrent_r
getopt	getopt_clip
getopt_long	getopt_long_only
getpagesize	getpagesizes
getpass	getpassphrase
getpeerucred	getpflags
getpgid	getpgrp
getpid	getpmsg
getppid	getppriv
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	getustack
getutent	getutid
getutline	getutmp
getutmpx	getutxent
getutxid	getutxline
getvfsany	getvfsent
getvfsfile	getvfsspec
getw	getwc
getwchar	getwd
getwidth	getws
getzoneid	getzoneidbyname
getzonenamebyid	glob
globfree	gmtime
gmtime_r	grantpt
gsignal	hasmntopt
hcreate	hdestroy
hsearch	iconv
iconv_close	iconv_open
imaxabs	imaxdiv
index	initgroups
initstate	innetgr
insque	ioctl
isaexec	isalnum
isalpha	isascii
isastream	isatty

isblank	iscntrl
isdigit	isenglish
isgraph	isideogram
islower	isnan
isnand	isnanf
isnumber	isphonogram
isprint	ispunct
issetugid	isspace
isspecial	isupper
iswalnum	iswalpha
iswblank	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
port_alert	port_associate
port_create	port_dissociate
port_get	port_getn
port_send	port_sendn
posix_openpt	posix_spawn
posix_spawn_file_actions_addclose	posix_spawn_file_actions_adddup2
posix_spawn_file_actions_addopen	posix_spawn_file_actions_destroy
posix_spawn_file_actions_init	posix_spawnattr_destroy
posix_spawnattr_getflags	posix_spawnattr_getpgroup
posix_spawnattr_getschedparam	posix_spawnattr_getschedpolicy
posix_spawnattr_getsigdefault	posix_spawnattr_getsigmask
posix_spawnattr_init	posix_spawnattr_setflags
posix_spawnattr_setpgroup	posix_spawnattr_setschedparam

posix_spawnattr_setschedpolicy	posix_spawnattr_setsigdefault
posix_spawnattr_setsigmask	posix_spawnp
pread	printf
printstack	priocntl
priocntlset	priv_addset
priv_allocset	priv_copyset
priv_delset	priv_emptyset
priv_fillset	priv_freeset
priv_getbyname	priv_getbynum
priv_getsetbyname	priv_getsetbynum
priv_gettext	priv_ineffect
priv_intersect	priv_inverse
priv_isemptyset	priv_isequalset
priv_isfullset	priv_ismember
priv_issubset	priv_set
priv_set_to_str	priv_str_to_set
priv_union	processor_bind
processor_info	profil
pselect	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_getstack	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_setstack	pthread_attr_setstackaddr
pthread_attr_setstacksize	pthread_barrier_destroy
pthread_barrier_init	pthread_barrier_wait
pthread_barrierattr_destroy	pthread_barrierattr_getpshared
pthread_barrierattr_init	pthread_barrierattr_setpshared
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_getclock
pthread_condattr_getpshared	pthread_condattr_init
pthread_condattr_setclock	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_reltimedlock_np	pthread_mutex_setprioceiling
pthread_mutex_timedlock	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_reltimedrdlock_np
pthread_rwlock_reltimedwrlock_np	pthread_rwlock_timedrdlock
pthread_rwlock_timedwrlock	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_setspecific
pthread_sigmask	pthread_setschedparam
pthread_setschedprio	pthread_spin_destroy
pthread_spin_init	pthread_spin_lock
pthread_spin_trylock	pthread_spin_unlock
pthread_testcancel	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_recipient_pid
rctlblk_set_value	rctlblk_size
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	scandir
scanf	schedctl_exit

schedctl_init	schedctl_lookup
schedctl_start	schedctl_stop
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
setenv	seteuid
setgid	setgrent
setgroups	sethostname
setitimer	setjmp
setkey	setlabel
setlinebuf	setlocale
setlogmask	setnetgrent
setpflags	setpgid
setpgrp	setppriv
setpriority	setpwent
setrctl	setregid
setreuid	setrlimit
setsid	setspent
setstate	settaskid
settimeofday	setuid
setusershell	setustack
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
stack_getbounds	stack_inbounds
stack_setbounds	stack_violation
stat	statfs
statvfs	stime
str2sig	strcasecmp
strcat	strchr
strcmp	strcoll
strcpy	strcspn
strdup	strerror
strerror_r	strfmon

strftime	string_to_decimal
strlcat	strlcpy
strlen	strncasecmp
strncat	strncmp
strncpy	strpbrk
strptime	strrchr
strsignal	strspn
strstr	strtod
strtof	strtoimax
strtok	strtok_r
strtol	strtold
strtoll	strtoul
strtoull	strtoumax
strtows	strxfrm
swab	swapcontext
swapctl	swprintf
swscanf	symlink
sync	sync_instruction_memory
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	ucred_free
ucred_get	ucred_getegid
ucred_geteuid	ucred_getgroups
ucred_getpflags	ucred_getpid
ucred_getprivset	ucred_getprojid
ucred_getrgid	ucred_getruid
ucred_getsgid	ucred_getsuid
ucred_getzoneid	ucred_size
ulckpwdf	ulimit
ulltostr	umask
umount	umount2
uname	ungetc

ungetwc	unlink
unlinkat	unlockpt
unordered	unsetenv
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
wcstof	wcstoimax

wcstok	wcstol
wcstold	wcstoll
wcstombs	wcstoul
wcstoull	wcstoumax
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	ptrace
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 x86 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

/lib/libc.so.1	shared object
/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 (32-bit)

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

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

NAME

libc_db – threads debugging library

SYNOPSIS

```
cc [ flag ... ] file ... -lc_db [ library ... ]
#include <proc_service.h>
#include <thread_db.h>
```

DESCRIPTION

The `libc_db` library provides support for monitoring and manipulating threads-related aspects of a multithreaded program. There are at least two processes involved, the controlling process and one or more target processes. The controlling process is the `libc_db` client, which links with `libc_db` and uses `libc_db` to inspect or modify threads-related aspects of one or more target processes. The target processes must be multithreaded processes that use `libc`. The controlling process mignt or might not be multithreaded itself.

The most commonly anticipated use for `libc_db` is that the controlling process will be a debugger for a multithreaded program, hence the "db" in `libc_db`.

The `libc_db` library is dependent on the internal implementation details of `libc`. It is a "friend" of `libc` in the C++ sense, which is precisely the "value added" by `libc_db`. It encapsulates the knowledge of `libc` internals that a debugger needs to manipulate the threads-related state of a target process.

To be able to inspect and manipulate target processes, `libc_db` makes use of certain process control primitives that must be provided by the process using `libc_db`. The imported interfaces are defined in `proc_service`(3PROC). In other words, the controlling process is linked with `libc_db` and calls routines in `libc_db`. In turn, `libc_db` calls certain routines that it expects the controlling process to provide. These process control primitives allow `libc_db` to:

- Look up symbols in a target process.
- Stop and continue individual lightweight processes (LWPs) within a target process.
- Stop and continue an entire target process.
- Read and write memory and registers in a target process.

Initially, a controlling process obtains a handle for a target process. Through that handle it can then obtain handles for the component objects of the target process, its threads, its synchronization objects, and its thread-specific-data keys.

When `libc_db` needs to return sets of handles to the controlling process, for example, when returning handles for all the threads in a target process, it uses an iterator function. An iterator function calls back a client-specified function once for each handle to be returned, passing one handle back on each call to the callback function. The calling function also passes another parameter to the iterator function, which the iterator function passes on to the callback function. This makes it easy to build a linked list of thread handles for a particular target process. The additional parameter is the head of the linked list, and the callback function simply inserts the current handle into the linked list.

Callback functions are expected to return an integer. Iteration terminates early if a callback function returns a non-zero value. Otherwise, iteration terminates when there are no more handles to pass back.

INTERFACES

The shared object `libc_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

`/lib/libc_db.so.1`	shared object
`/lib/64/libc_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 `intro`(3), `td_ta_new`(3C_DB), `attributes`(5), `threads`(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

libcontract – contract management library

SYNOPSIS

```
cc [ flag... ] `getconf LFS_CFLAGS` file... -lcontract [ library... ]
#include <libcontract.h>
```

DESCRIPTION

Functions in this library provide various interfaces to interact with the `contract`(4) file system. The header provides structure and function declarations for all library interfaces.

INTERFACES

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

`ct_ctl_abandon`	`ct_ctl_ack`
`ct_ctl_adopt`	`ct_ctl_newct`
`ct_ctl_qack`	`ct_event_free`
`ct_event_get_ctid`	`ct_event_get_evid`
`ct_event_get_flags`	`ct_event_get_nevid`
`ct_event_get_newct`	`ct_event_get_type`
`ct_event_read`	`ct_event_read_critical`
`ct_event_reliable`	`ct_event_reset`
`ct_pr_event_get_exitstatus`	`ct_pr_event_get_gcorefile`
`ct_pr_event_get_pcorefile`	`ct_pr_event_get_pid`
`ct_pr_event_get_ppid`	`ct_pr_event_get_sender`
`ct_pr_event_get_senderct`	`ct_pr_event_get_signal`
`ct_pr_event_get_zcorefile`	`ct_pr_status_get_contracts`
`ct_pr_status_get_fatal`	`ct_pr_status_get_members`
`ct_pr_status_get_param`	`ct_pr_tmpl_get_fatal`
`ct_pr_tmpl_get_param`	`ct_pr_tmpl_get_transfer`
`ct_pr_tmpl_set_fatal`	`ct_pr_tmpl_set_param`
`ct_pr_tmpl_set_transfer`	`ct_status_free`
`ct_status_get_cookie`	`ct_status_get_critical`
`ct_status_get_holder`	`ct_status_get_id`
`ct_status_get_informative`	`ct_status_get_nevents`
`ct_status_get_nevid`	`ct_status_get_ntime`
`ct_status_get_qtime`	`ct_status_get_state`

`ct_status_get_type`	`ct_status_get_zoneid`
`ct_status_read`	`ct_tmpl_activate`
`ct_tmpl_clear`	`ct_tmpl_create`
`ct_tmpl_get_cookie`	`ct_tmpl_get_critical`
`ct_tmpl_get_informative`	`ct_tmpl_set_cookie`
`ct_tmpl_set_critical`	`ct_tmpl_set_informative`

FILES

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

ATTRIBUTES

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

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

SEE ALSO

`pvs`(1), `intro`(3), `contract`(4), `attributes`(5), `lfcompile`(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_cpu
cpc_bind_curlwp	cpc_bind_event
cpc_bind_pctx	cpc_buf_add
cpc_buf_copy	cpc_buf_create
cpc_buf_destroy	cpc_buf_get
cpc_buf_hrtime	cpc_buf_set
cpc_buf_sub	cpc_buf_tick
cpc_buf_zero	cpc_caps
cpc_cciname	cpc_close
cpc_cpuref	cpc_count_sys_events
cpc_count_usr_events	cpc_disable
cpc_enable	cpc_event_accum
cpc_event_diff	cpc_eventtostr
cpc_getcciname	cpc_getcpuref
cpc_getcpuver	cpc_getnpic
cpc_getusage	cpc_npic
cpc_open	cpc_pctx_bind_event
cpc_pctx_invalidate	cpc_pctx_rele
cpc_pctx_take_sample	cpc_rele
cpc_request_preset	cpc_set_add_request
cpc_set_create	cpc_set_destroy
cpc_set_restart	cpc_set_sample
cpc_seterrfn	cpc_seterrhndlr
cpc_shared_bind_event	cpc_shared_close

`cpc_shared_open`	`cpc_shared_rele`
`cpc_shared_take_sample`	`cpc_strtoevent`
`cpc_take_sample`	`cpc_unbind`
`cpc_version`	`cpc_walk_attrs`
`cpc_walk_events_all`	`cpc_walk_events_pic`
`cpc_walk_names`	`cpc_walk_requests`

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
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.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

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

libdat – direct access transport library

SYNOPSIS

```
cc [ flag... ] file... -ldat [ library... ]
#include <dat/udat.h>
```

DESCRIPTION

The libdat library provides an application with the User Direct Access Programming Library (uDAPL) 1.2 functions to access the underlying RDMA-able interconnects. Different uDAPL service providers listed in the DAT static registry `dat.conf`(4) can be registered during runtime with the DAT library. After an application opens an interface adapter belonging to a particular service provider, all function calls will be redirected to that service provider's library.

INTERFACES

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

uDAPL 1.1

dat_cno_create	dat_cno_free
dat_cno_modify_agent	dat_cno_query
dat_cno_wait	dat_cr_accept
dat_cr_handoff	dat_cr_query
dat_cr_reject	dat_ep_connect
dat_ep_create	dat_ep_disconnect
dat_ep_dup_connect	dat_ep_free
dat_ep_get_status	dat_ep_modify
dat_ep_post_rdma_read	dat_ep_post_rdma_write
dat_ep_post_recv	dat_ep_post_send
dat_ep_query	dat_ep_reset
dat_evd_clear_unwaitable	dat_evd_create
dat_evd_dequeue	dat_evd_disable
dat_evd_enable	dat_evd_free
dat_evd_modify_cno	dat_evd_post_se
dat_evd_query	dat_evd_resize
dat_evd_set_unwaitable	dat_evd_wait
dat_get_consumer_context	dat_get_handle_type
dat_ia_close	dat_ia_open
dat_ia_query	dat_lmr_create
dat_lmr_free	dat_lmr_query

dat_provider_fini	dat_provider_init
dat_psp_create	dat_psp_create_any
dat_psp_free	dat_psp_query
dat_pz_create	dat_pz_free
dat_pz_query	dat_registry_list_providers
dat_rmr_bind	dat_rmr_create
dat_rmr_free	dat_rmr_query
dat_rsp_create	dat_rsp_free
dat_rsp_query	dat_set_consumer_context
dat_strerror	

uDAPL 1.2

dat_ep_create_with_srq	dat_ep_recv_query
dat_ep_set_watermark	dat_lmr_sync_rdma_read
dat_lmr_sync_rdma_write	dat_srq_create
dat_srq_free	dat_srq_post_recv
dat_srq_query	dat_srq_resize
dat_srq_set_lw	

The shared object `libdat.so.1` also provides the public interfaces defined below for service providers.

dat_registry_add_provider	dat_registry_remove_provider

FILES

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

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWudaplu (user) SUNWudaplr (root)
Interface Stability	Standard: uDAPL, 1.1, 1.2
MT-Level	Unsafe

SEE ALSO | `datadm`(1M), `intro`(3), `dat.conf`(4), `attributes`(5)

NOTES | The `libdat` library supports service providers written according to the uDAPL 1.2 specification. A service provider library has to be a dynamic loadable shared object with two public entry points exported:

```
dat_provider_init                    dat_provider_fini
```

In terms of installation, the service provider package should include a `service_provider.conf`(4) file. The `datadm`(1M) administrative configuration program should be used to add and remove service provider's entries in the system-wide `dat.conf`(4).

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`
`dbmclose`	`dbminit`
`dbrdonly`	`delete`
`dirbuf`	`dirf`
`fetch`	`firstkey`
`hmask`	`maxbno`
`nextkey`	`pagbuf`
`pagf`	`store`

FILES

`/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

`/lib/libdevid.so.1`	shared object.
`/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)
Interface Stability	Stable
MT-Level	MT–Safe

SEE ALSO

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

NAME

libdevinfo – device information library

SYNOPSIS

```
cc [ flag... ] file... -ldevinfo [ library... ]
#include <libdevinfo.h>
```

DESCRIPTION

Functions in this library access device configuration information.

Device configuration data is organized as a tree of device nodes, defined as `di_node_t` in the `libdevinfo` interfaces. Each `di_node_t` represents a physical or logical (pseudo) device. Three types of data are associated with device nodes:

- data defined for all device nodes (attributes)
- properties specific to each device
- minor node data

All device nodes have a set of common attributes, such as a node name, an instance number, and a driver binding name. Common device node attributes are accessed by calling interfaces listed on the `di_binding_name`(3DEVINFO) manual page. Each device node also has a physical path, which is accessed by calling `di_devfs_path`(3DEVINFO).

Properties provide device specific information for device configuration and usage. Properties can be defined by software (`di_prop_t`) or by firmware (`di_prom_prop_t`). One way to access each `di_prop_t` is to make successive calls to `di_prop_next`(3DEVINFO) until `DI_PROP_NIL` is returned. For each `di_prop_t`, use interfaces on the `di_prop_bytes`(3DEVINFO) manual page to obtain property names and values. Another way to access these properties is to call `di_prop_lookup_bytes`(3DEVINFO) to find the value of a property with a given name. Accessing a `di_prom_prop_t` is similar to accessing a `di_prop_t`, except that the interface names start with `di_prom_prop` and additional calls to `di_prom_init`(3DEVINFO) and `di_prom_fini`(3DEVINFO) are required.

Minor nodes contain information exported by the device for creating special files for the device. Each device node has 0 or more minor nodes associated with it. A list minor nodes (`di_minor_t`) can be obtained by making successive calls to `di_minor_next`(3DEVINFO) until `DI_MINOR_NIL` is returned. For each minor node, `di_minor_devt`(3DEVINFO) and related interfaces are called to get minor node data.

Using libdevinfo involves three steps:

- Creating a snapshot of the device tree
- Traversing the device tree to get information of interest
- Destroying the snapshot of the device tree

A snapshot of the device tree is created by calling `di_init`(3DEVINFO) and destroyed by calling `di_fini`(3DEVINFO). An application can specify the data to be included in the snapshot (full or partial tree, include or exclude properties and minor nodes) and get a handle to the root of the device tree. See `di_init`(3DEVINFO) for details. The application then traverses the device tree in the snapshot to obtain device configuration data.

The device tree is normally traversed through parent-child-sibling linkage. Each device node contains references to its parent, its next sibling, and the first of its children. Given the `di_node_t` returned from `di_init()`, one can find all children by first calling `di_child_node`(3DEVINFO), followed by successive calls to `di_sibling_node`(3DEVINFO) until `DI_NODE_NIL` is returned. By following this procedure recursively, an application can visit all device nodes contained in the snapshot. Two interfaces,The `di_walk_node`(3DEVINFO) and `di_walk_minor`(3DEVINFO) functions are provided to facilitate device tree traversal. The `di_walk_node()` function visits all device nodes and executes a user-supplied callback function for each node visited. The `di_walk_minor()` function does the same for each minor node in the device tree.

An alternative way to traverse the device tree is through the per-driver device node linkage. Device nodes contain a reference to the next device node bound to the same driver. Given the `di_node_t` returned from `di_init()`, an application can find all device nodes bound to a driver by first calling `di_drv_first_node`(3DEVINFO), followed by successive calls to `di_drv_next_node`(3DEVINFO) until `DI_NODE_NIL` is returned. Traversing the per-driver device node list works only when the snapshot includes all device nodes.

See `di_init`(3DEVINFO) for examples of `libdevinfo` usage. See *Writing Device Drivers* for information about Solaris 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_minor_path`	`di_devfs_path`
`di_devfs_path_free`	`di_devid`
`di_driver_major`	`di_driver_name`
`di_driver_ops`	`di_drv_first_node`
`di_drv_next_node`	`di_fini`
`di_init`	`di_instance`
`di_link_next_by_lnode`	`di_link_next_by_node`
`di_link_private_get`	`di_link_private_set`
`di_link_spectype`	`di_link_to_lnode`
`di_lnode_devinfo`	`di_lnode_devt`
`di_lnode_name`	`di_lnode_next`

di_lnode_private_get	di_lnode_private_set
di_minor_devt	di_minor_name
di_minor_next	di_minor_nodetype
di_minor_private_get	di_minor_private_set
di_minor_spectype	di_minor_type
di_node_name	di_nodeid
di_parent_node	di_node_private_get
di_node_private_set	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_int64
di_prop_ints	di_prop_lookup_bytes
di_prop_lookup_int64	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_link
di_walk_lnode	di_walk_minor
di_walk_node	

EXAMPLES

EXAMPLE 1 Information accessible through `libdevinfo` interfaces

The following example illustrates the kind of information accessible through `libdevinfo` interfaces for a device node representing a hard disk (sd2):

```
Attributes
    node name:  sd
    instance:   2
    physical path:  /sbus@1f,0/espdma@e,8400000/esp@e,8800000/sd@2,0

Properties
    target=2
    lun=0

Minor nodes
    (disk partition /dev/dsk/c0t2d0s0)
        name:       a
        dev_t:      0x0080010 (32/16)
```

EXAMPLE 1 Information accessible through `libdevinfo` interfaces *(Continued)*

```
        spectype:   IF_BLK (block special)
    (disk partition /dev/rdsk/c0t2d0s2)
        name:       c,raw
        dev_t:      0x0080012 (32/18)
        spectype:   IF_CHR (character special)
```

FILES

`/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)
Interface Stability	Evolving
MT-Level	Safe

SEE ALSO

`pvs`(1), `devlinks`(1M), `prtconf`(1M), `intro`(3), `di_binding_name`(3DEVINFO), `di_child_node`(3DEVINFO), `di_devfs_path`(3DEVINFO), `di_drv_first_node`(3DEVINFO), `di_drv_next_node`(3DEVINFO), `di_fini`(3DEVINFO), `di_prom_init`(3DEVINFO), `di_minor_devt`(3DEVINFO), `di_minor_next`(3DEVINFO), `di_prom_fini`(3DEVINFO), `di_prom_init`(3DEVINFO), `di_prop_bytes`(3DEVINFO), `di_prop_lookup_bytes`(3DEVINFO), `di_prop_next`(3DEVINFO), `di_sibling_node`(3DEVINFO), `di_walk_minor`(3DEVINFO), `di_walk_node`(3DEVINFO), `attributes`(5)

Writing Device Drivers

NAME

libdl – dynamic linking library

SYNOPSIS

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

DESCRIPTION

Historically, functions in `libdl` provided for dynamic linking support. This functionality now resides in `libc`(3LIB).

This library is maintained to provide backward compatibility for both runtime and compilation environments. The shared object is implemented as a filter on the runtime linker. See `ld.so.1`(1). New application development need not specify `-ldl`.

INTERFACES

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

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

FILES

`/lib/libdl.so.1`	shared object
`/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), `libc`(3LIB), `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`

FILES

`/usr/lib/libdmici.so.1`	shared object
`/usr/lib/64/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

libdoor – doors library

SYNOPSIS

```
cc [ flag... ] file... -ldoor [ library... ]
#include <door.h>
```

DESCRIPTION

The functions in this library provide programmatic access to doors, including the ability to create and call them. Doors are a fast light-weight RPC mechanism for secure control transfer between processes on the same machine. Conceptually, a thread in one process can issue a call using a door descriptor that causes code to be executed in another process and then returns using the traditional synchronous RPC model. Doors can also be used to pass data and file descriptors between processes.

INTERFACES

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

`door_bind`	`door_call`
`door_create`	`door_cred`
`door_info`	`door_return`
`door_revoke`	`door_server_create`
`door_ucred`	`door_unbind`

FILES

`/lib/libdoor.so.1`	shared object
`/lib/64/libdoor.so.1`	64–bit shared object

ATTRIBUTES

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

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

SEE ALSO

`intro(3)`, `door_bind`(3DOOR), `door_call`(3DOOR), `door_create`(3DOOR), `door_cred`(3DOOR), `door_info`(3DOOR), `door_return`(3DOOR), `door_revoke`(3DOOR), `door_server_create`(3DOOR), `door_ucred`(3DOOR), `attributes`(5)

Stevens, W. Richard. *UNIX Network Programming, Volume 2: Interprocess Communications, 2/e*. Tucson, Ariz.: Prentice Hall, 1999.

NAME

libdtrace – DTrace dynamic tracing software library

DESCRIPTION

Functions in this library define the interface for interacting with the DTrace dynamic tracing software, including the D language compiler and facilities for enabling probes and consuming trace data.

INTERFACES

The interfaces provided by libdtrace.so.1 are currently private to the implementation of the Solaris system and DTrace subsystem and are subject to change at any time without notice. Applications using these interfaces might fail to run on future releases. Refer to the *Solaris Dynamic Tracing Guide* for a description of the public documented interfaces available for the DTrace facility.

FILES

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

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWdtrc
Interface Stability	Private
MT-Level	Unsafe

SEE ALSO

`dtrace`(1M), `attributes`(5), `dtrace`(7D)

Solaris Dynamic Tracing Guide

NAME

libefi – EFI partition table library

SYNOPSIS

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

DESCRIPTION

The functions in this library manipulate a disk's EFI partition table.

INTERFACES

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

`efi_alloc_and_init`	`efi_alloc_and_read`
`efi_free`	`efi_write`

FILES

`/lib/libefi.so.1`	shared object
`/lib/64/libefi.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	Unsafe

SEE ALSO

`intro`(3), `efi_alloc_and_init`(3EXT), `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_getshnum	elf_getshstrndx
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_getcap	gelf_getclass
gelf_getdyn	gelf_getehdr
gelf_getmove	gelf_getphdr
gelf_getrel	gelf_getrela
gelf_getshdr	gelf_getsym
gelf_getsyminfo	gelf_getsymshndx
gelf_newehdr	gelf_newphdr
gelf_update_cap	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_symshndx
gelf_update_syminfo	gelf_xlatetof
gelf_xlatetom	nlist

FILES

/lib/libelf.so.1	shared object
/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 (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)
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.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 (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

`/lib/libgen.so.1`	shared object
`/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 (32–bit) SUNWcslx (64–bit)

ATTRIBUTE TYPE	ATTRIBUTE VALUE
MT-Level	Safe

SEE ALSO intro(3), attributes(5)

NAME

libgen.h, libgen – definitions for pattern matching functions

SYNOPSIS

```
#include <libgen.h>
```

DESCRIPTION

The `<libgen.h>` header lists definitions used for string pattern-matching and pathname manipulation. See `libgen`(3LIB).

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Standard

SEE ALSO

`basename`(3C), `dirname`(3C), `libgen`(3LIB), `attributes`(5), `standards`(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.

When `libgss` fails to load or initialize a mechanism listed in `/etc/gss/mech`, a message is sent to `syslog`(3C).

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_store_cred	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)`, `syslog`(3C), `attributes`(5)

Solaris Security for Developers Guide

NAME

libhbaapi – Common Fibre Channel HBA information library

SYNOPSIS

```
cc [ flag... ] file... -lHBAAPI [ library... ]
#include <hbaapi.h>
```

DESCRIPTION

The functions in this library access Fibre Channel HBA data.

Fibre Channel HBA information is provided through a standard interface in a vendor independent manner. This common interface provides access to the following information:

- Local HBA attributes
- Local HBA port attributes and statistics
- Mapping between FCP-2 discovered devices and operating system SCSI information
- Discovered devices port attributes
- SCSI commands for discovered FCP-2 devices (Report LUNS, Read Capacity, and Inquiry)
- Common Transport commands to discover Fabric details

INTERFACES

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

HBA_CloseAdapter	HBA_FreeLibrary
HBA_GetAdapterAttributes	HBA_GetAdapterName
HBA_GetAdapterPortAttributes	HBA_GetBindingCapability
HBA_GetBindingSupport	HBA_GetDiscoveredPortAttributes
HBA_GetEventBuffer	HBA_GetFC4Statistics
HBA_GetFCPStatistics	HBA_GetFcpPersistentBinding
HBA_GetFcpTargetMapping	HBA_GetFcpTargetMappingV2
HBA_GetNumberOfAdapters	HBA_GetPersistentBindingV2
HBA_GetPortAttributesByWWN	HBA_GetPortStatistics
HBA_GetRNIDMgmtInfo	HBA_GetVendorLibraryAttributes
HBA_GetVersion	HBA_GetWrapperLibraryAttributes
HBA_LoadLibrary	HBA_OpenAdapter
HBA_OpenAdapterByWWN	HBA_RefreshAdapterConfiguration
HBA_RefreshInformation	HBA_RegisterForAdapterAddEvents
HBA_RegisterForAdapterEvents	HBA_RegisterForAdapterPortEvents

HBA_RegisterForAdapterPortStatEvents	HBA_RegisterForLinkEvents
HBA_RegisterForTargetEvents	HBA_RemoveAllPersistentBindings
HBA_RemoveCallback	HBA_RemovePersistentBinding
HBA_ResetStatistics	HBA_ScsiInquiryV2
HBA_ScsiReadCapacityV2	HBA_ScsiReportLUNsV2
HBA_SendCTPassThru	HBA_SendCTPassThruV2
HBA_SendLIRR	HBA_SendRLS
HBA_SendRNID	HBA_SendRNIDV2
HBA_SendRPL	HBA_SendRPS
HBA_SendReadCapacity	HBA_SendReportLUNs
HBA_SendSRL	HBA_SendScsiInquiry
HBA_SetBindingSupport	HBA_SetPersistentBindingV2
HBA_SetRNIDMgmtInfo	

USAGE

Client applications link with the Common Library (using `-l`HBAAPI) to access the interfaces. The Common Library dynamically loads individual Vendor-Specific Libraries (VSL) listed in `/etc/hba.conf` described on the `hba.conf`(4).

Using the `libhbaapi` involves the following steps:

1. Optionally determining the version of the library by calling `HBA_GetVersion`(3HBAAPI).
2. Initializing the Common Library by calling `HBA_LoadLibrary`(3HBAAPI).
3. Determine the number of HBAs known to the common library by calling `HBA_GetNumberOfAdapters`(3HBAAPI).
4. Determine each HBA name in turn by calling `HBA_GetAdapterName`(3HBAAPI).
5. Open each HBA in turn by calling `HBA_OpenAdapter`(3HBAAPI).
6. Operate on a given HBA by calling the following:
   - `HBA_GetAdapterAttributes`(3HBAAPI)
   - `HBA_GetAdapterPortAttributes`(3HBAAPI)
   - `HBA_GetDiscoveredPortAttributes`(3HBAAPI)
   - `HBA_GetPortAttributesByWWN`(3HBAAPI)
   - `HBA_SendCTPassThru`(3HBAAPI)
   - `HBA_SendCTPassThruV2`(3HBAAPI)
   - `HBA_GetEventBuffer`(3HBAAPI)
   - `HBA_SetRNIDMgmtInfo`(3HBAAPI)
   - `HBA_GetRNIDMgmtInfo`(3HBAAPI)
   - `HBA_SendRNID`(3HBAAPI)
   - `HBA_SendRNIDV2`(3HBAAPI)
   - `HBA_RefreshInformation`(3HBAAPI)

- HBA_RefreshAdapterConfiguration(3HBAAPI)
- HBA_GetVendorLibraryAttributes(3HBAAPI)
- HBA_GetWrapperLibraryAttributes(3HBAAPI)
- HBA_ResetStatistics(3HBAAPI)
- HBA_GetFcpTargetMapping(3HBAAPI)
- HBA_GetFcpTargetMappingV2(3HBAAPI)
- HBA_GetFcpPersistentBinding(3HBAAPI)
- HBA_SendScsiInquiry(3HBAAPI)
- HBA_SendReportLUNs(3HBAAPI)
- HBA_ScsiReportLUNsV2(3HBAAPI)
- HBA_SendReadCapacity(3HBAAPI)
- HBA_SendRLS(3HBAAPI)

7. Close open HBAs by calling HBA_CloseAdapter(3HBAAPI).
8. Unload the library by calling HBA_FreeLibrary(3HBAAPI).

ERRORS

Errors are generally returned from the underlying VSL and can include any of the following values:

HBA_STATUS_OK
: Request completed successfully. (No Error)

HBA_STATUS_ERROR
: Non-specific error encountered.

HBA_STATUS_ERROR_NOT_SUPPORTED
: The VSL does not support this interface.

HBA_STATUS_ERROR_INVALID_HANDLE
: The *handle* argument does not refer to an open HBA handle.

HBA_STATUS_ERROR_ARG
: An argument in the request was invalid.

HBA_STATUS_ERROR_ILLEGAL_WWN
: A WWN in the request was not recognized.

HBA_STATUS_ERROR_ILLEGAL_INDEX
: An index in the request was not recognized.

HBA_STATUS_ERROR_MORE_DATA
: A larger buffer is required to complete the requested operation.

HBA_STATUS_ERROR_STALE_DATA
: The state of the HBA has changed, possibly due to Dynamic Reconfiguration or devices being added or removed. The caller should call HBA_RefreshInformation(3HBAAPI) and reissue any discovery logic to reset all indexes related to this HBA.

HBA_STATUS_SCSI_CHECK_CONDITION
: A SCSI check-condition was encountered during the I/O operation. Not all VSLs report this error value. Some might return HBA_STATUS_ERROR when a check-condition is encountered, or HBA_STATUS_OK.

`HBA_STATUS_ERROR_BUSY`
The requested device is busy. A retry might be effective.

`HBA_STATUS_ERROR_TRY_AGAIN`
The requested I/O timed out. A retry might be effective.

`HBA_STATUS_ERROR_UNAVAILABLE`
The requested HBA has been removed or deactivated.

All other error values are reserved.

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcfcl (32–bit) SUNWcfclx (64–bit)
Interface Stability	Standard: FC-MI 1.92 (API version 1) Standard: FC-HBA Version 4 (API version 2)
MT-Level	Safe

SEE ALSO

`HBA_GetAdapterAttributes`(3HBAAPI), `HBA_GetAdapterName`(3HBAAPI), `HBA_GetAdapterPortAttributes`(3HBAAPI), `HBA_GetBindingCapability`(3HBAAPI), `HBA_GetDiscoveredPortAttributes`(3HBAAPI), `HBA_GetEventBuffer`(3HBAAPI), `HBA_GetFcpPersistentBinding`(3HBAAPI), `HBA_GetFcpTargetMapping`(3HBAAPI), `HBA_GetNumberOfAdapters`(3HBAAPI), `HBA_GetPortAttributesByWWN`(3HBAAPI), `HBA_GetPortStatistics`(3HBAAPI), `HBA_GetVersion`(3HBAAPI), `HBA_GetWrapperLibraryAttributes`(3HBAAPI), `HBA_LoadLibrary`(3HBAAPI), `HBA_OpenAdapter`(3HBAAPI), `HBA_RefreshInformation`(3HBAAPI), `HBA_RegisterForAdapterEvents`(3HBAAPI), `HBA_SendCTPassThru`(3HBAAPI), `HBA_SendRLS`(3HBAAPI), `HBA_SendScsiInquiry`(3HBAAPI), `HBA_SetRNIDMgmtInfo`(3HBAAPI), `hba.conf`(4), `attributes`(5)

T11 FC-MI Specification

NAME

libidnkit – IDN conversion library

SYNOPSIS

```
cc [ flag... ] file... -lidnkit [ library... ]
#include <idn/api.h>
```

DESCRIPTION

Functions in this library provide conversions between ACE string and multibyte character string of the current locale or a specified codeset. They support various manipulations of internationalized domain names, including encoding conversion and name preparation. They are designed according to IDNA framework where each application must do necessary preparations for the internationalized domain names before passing them to the resolver. The library provides easy-to-use, high-level interfaces to help applications with the preparation.

The `libidnkit` library internally uses `iconv`(3C) to provide encoding conversion from UTF-8 to the local encoding (such as ISO8859-1, usually determined by the current locale), and from the local encoding to UTF-8.

INTERFACES

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

`idn_decodename`	`idn_decodename2`
`idn_enable`	`idn_encodename`
`idn_nameinit`	

FILES

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

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWidnl
Interface Stability	Evolving
MT-Level	Unsafe

SEE ALSO

`intro`(3), `iconv`(3C), `idn_decodename`(3EXT), `setlocale`(3C), `hosts`(4), `attributes`(5), `environ`(5)

RFC 3490	Internationalizing Domain Names in Applications (IDNA)
RFC 3491	Nameprep: A Stringprep Profile for Internationalized Domain Names (IDN)
RFC 3492	Punycode: A Bootstring encoding of Unicode for Internationalized Domain Names in Applications (IDNA)
RFC 3454	Preparation of Internationalized Strings ("stringprep")

RFC 952	DoD Internet Host Table Specification
RFC 921	Domain Name System Implementation Schedule - Revised
STD 3, RFC 1122	Requirements for Internet Hosts -- Communication Layers
STD 3, RFC 1123	Requirements for Internet Hosts -- Applications and Support

Unicode Standard Annex #15: Unicode Normalization Forms, Version 3.2.0. http://www.unicode.org/unicode/reports/tr15/tr15-22.html

International Language Environments Guide (for this version of Solaris)

COPYRIGHT AND LICENSE

Copyright (c) 2000-2002 Japan Network Information Center. All rights reserved.

By using this file, you agree to the terms and conditions set forth bellow.

LICENSE TERMS AND CONDITIONS

The following License Terms and Conditions apply, unless a different license is obtained from Japan Network Information Center ("JPNIC"), a Japanese association, Kokusai-Kougyou-Kanda Bldg 6F, 2-3-4 Uchi-Kanda, Chiyoda-ku, Tokyo 101-0047, Japan.

1. Use, Modification and Redistribution (including distribution of any modified or derived work) in source and/or binary forms is permitted under this License Terms and Conditions.
2. Redistribution of source code must retain the copyright notices as they appear in each source code file, this License Terms and Conditions.
3. Redistribution in binary form must reproduce the Copyright Notice, this License Terms and Conditions, in the documentation and/or other materials provided with the distribution. For the purposes of binary distribution the "Copyright Notice" refers to the following language: "Copyright (c) 2000-2002 Japan Network Information Center. All rights reserved."
4. The name of JPNIC may not be used to endorse or promote products derived from this Software without specific prior written approval of JPNIC.
5. Disclaimer/Limitation of Liability: THIS SOFTWARE IS PROVIDED BY JPNIC "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL JPNIC BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

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 is implemented as a filter on `libc.so.1`. New application development need not specify `-lintl`.

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

`/lib/libintl.so.1`	a filter on `/lib/libc.so.1`
`/lib/64/libintl.so.1`	a filter on `/lib/64/libc.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 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

`/lib/libkstat.so.1`	shared object
`/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)
Interface Stability	Stable
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.

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) SUNWcslx (64–bit)
Interface Stability	`kvm_read()` and `kvm_write()` are Obsolete; the remaining functions are Stable.
MT-Level	Unsafe

SEE ALSO

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

NAME

libl – lex library

SYNOPSIS

```
cc [ flag... ] file...  [ 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.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
MT-Level	MT–Safe

SEE ALSO

intro(3), attributes(5)

NAME

liblgrp – locality group library

SYNOPSIS

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

DESCRIPTION

The functions in this library traverse the lgroup (locality group) hierarchy, discover its contents, and set a thread's affinity for an lgroup. A locality group represents the set of CPU-like and memory-like hardware devices that are at most some locality apart from each other.

INTERFACES

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

`lgrp_affinity_get`	`lgrp_affinity_set`
`lgrp_children`	`lgrp_cookie_stale`
`lgrp_cpus`	`lgrp_fini`
`lgrp_home`	`lgrp_init`
`lgrp_latency`	`lgrp_mem_size`
`lgrp_nlgrps`	`lgrp_parents`
`lgrp_root`	`lgrp_version`
`lgrp_view`	

FILES

`/usr/lib/liblgrp.so.1`	shared object
`/usr/lib/64/liblgrp.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

`intro`(3), `lgrp_affinity_get`(3LGRP), `lgrp_children`(3LGRP), `lgrp_cookie_stale`(3LGRP), `lgrp_cpus`(3LGRP), `lgrp_fini`(3LGRP), `lgrp_home`(3LGRP), `lgrp_init`(3LGRP), `lgrp_latency`(3LGRP), `lgrp_mem_size`(3LGRP), `lgrp_nlgrps`(3LGRP), `lgrp_parents`(3LGRP), `lgrp_root`(3LGRP), `lgrp_version`(3LGRP), `lgrp_view`(3LGRP), `attributes`(5)

NAME

libm – C math library

SYNOPSIS

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

DESCRIPTION

Functions in this library provide common elementary mathematical functions and floating point environment routines defined by System V, ANSI C, POSIX, and so on. See `standards`(5). Additional functions in this library provide extended support for handling floating point exceptions.

INTERFACES

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

acos	acosf
acosh	acoshf
acoshl	acosl
asin	asinf
asinh	asinhf
asinhl	asinl
atan	atan2
atan2f	atan2l
atanf	atanh
atanhf	atanhl
atanl	cabs
cabsf	cabsl
cacos	cacosf
cacosh	cacoshf
cacoshl	cacosl
carg	cargf
cargl	casin
casinf	casinh
casinhf	casinhl
casinl	catan
catanf	catanh
catanhf	catanhl
catanl	cbrt

cbrtf	cbrtl
ccos	ccosf
ccosh	ccoshf
ccoshl	ccosl
ceil	ceilf
ceill	cexp
cexpf	cexpl
cimag	cimagf
cimagl	clog
clogf	clogl
conj	conjf
conjl	copysign
copysignf	copysignl
cos	cosf
cosh	coshf
coshl	cosl
cpow	cpowf
cpowl	cproj
cprojf	cprojl
creal	crealf
creall	csin
csinf	csinh
csinhf	csinhl
csinl	csqrt
csqrtf	csqrtl
ctan	ctanf
ctanh	ctanhf
ctanhl	ctanl
erf	erfc
erfcf	erfcl

erff	erfl
exp	exp2
exp2f	exp2l
expf	expl
expm1	expm1f
expm1l	fabs
fabsf	fabsl
fdim	fdimf
fdiml	feclearexcept
fegetenv	fegetexceptflag
fegetround	feholdexcept
feraiseexcept	fesetenv
fesetexceptflag	fesetround
fetestexcept	feupdateenv
fex_get_handling	fex_get_log
fex_get_log_depth	fex_getexcepthandler
fex_log_entry	fex_merge_flags
fex_set_handling	fex_set_log
fex_set_log_depth	fex_setexcepthandler
floor	floorf
floorl	fma
fmaf	fmal
fmax	fmaxf
fmaxl	fmin
fminf	fminl
fmod	fmodf
fmodl	frexp
frexpf	frexpl
gamma	gamma_r
gammaf	gammaf_r

gammal	gammal_r
hypot	hypotf
hypotl	ilogb
ilogbf	ilogbl
isnan	j0
j0f	j0l
j1	j1f
j1l	jn
jnf	jnl
ldexp	ldexpf
ldexpl	lgamma
lgamma_r	lgammaf
lgammaf_r	lgammal
lgammal_r	llrint
llrintf	llrintl
llround	llroundf
llroundl	log
log10	log10f
log10l	log1p
log1pf	log1pl
log2	log2f
log2l	logb
logbf	logbl
logf	logl
lrint	lrintf
lrintl	lround
lroundf	lroundl
matherr	modf
modff	modfl
nan	nanf

nanl	nearbyint
nearbyintf	nearbyintl
nextafter	nextafterf
nextafterl	nexttoward
nexttowardf	nexttowardl
pow	powf
powl	remainder
remainderf	remainderl
remquo	remquof
remquol	rint
rintf	rintl
round	roundf
roundl	scalb
scalbf	scalbl
scalbln	scalblnf
scalblnl	scalbn
scalbnf	scalbnl
signgam	signgamf
signgaml	significand
significandf	significandl
sin	sincos
sincosf	sincosl
sinf	sinh
sinhf	sinhl
sinl	sqrt
sqrtf	sqrtl
tan	tanf
tanh	tanhf
tanhl	tanl
tgamma	tgammaf

tgammal	trunc
truncf	truncl
y0	y0f
y0l	y1
y1f	y1l
yn	ynf
ynl	

The following interfaces are unique to the x86 version of this library:

fegetprec	fesetprec

ACCURACY

ISO/IEC 9899:1999, also known as C99, specifies the functions listed in the following tables and states that the accuracy of these functions is "implementation-defined". The information below characterizes the accuracy of these functions as implemented in `libm.so.2`. For each function, the tables provide an upper bound on the largest error possible for any argument and the largest error actually observed among a large sample of arguments. Errors are expressed in "units in the last place", or ulps, relative to the exact function value for each argument (regarding the argument as exact). Ulps depend on the precision of the floating point format: if y is the exact function value, x and x' are adjacent floating point numbers such that $x < y < x'$, and x'' is the computed function value, then provided x, x', and x'' all lie in the same binade, the error in x'' is $|y - x''| / |x - x'|$ ulps. In particular, when the error is less than one ulp, the computed value is one of the two floating point numbers adjacent to the exact value.

The bounds and observed errors listed below apply only in the default floating point modes. Specifically, on SPARC, these bounds assume the rounding direction is round-to-nearest and non-standard mode is disabled. On x86, the bounds assume the rounding direction is round-to-nearest and the rounding precision is round-to-64-bits. Moreover, on x86, floating point function values are returned in a floating point register in extended double precision format, but the bounds below assume that the result value is then stored to memory in the format corresponding to the function's type.

The error bounds listed below are believed to be correct, but smaller bounds might be proved later. The observed errors are the largest ones currently known, but larger errors might be discovered later. Numbers in the notes column refer to the notes following the tables.

Real Functions

Single precision real functions (SPARC and x86)

function	error bound (ulps)	largest error observed (ulps)	notes
acosf	1.0	< 1	
acoshf	1.0	< 1	
asinf	1.0	< 1	
asinhf	1.0	< 1	
atanf	1.0	< 1	
atan2f	1.0	< 1	
atanhf	1.0	< 1	
cbrtf	1.0	< 1	
cosf	1.0	< 1	
coshf	1.0	< 1	
erff	1.0	< 1	
erfcf	1.0	< 1	
expf	1.0	< 1	
exp2f	1.0	< 1	
expm1f	1.0	< 1	
hypotf	1.0	< 1	
lgammaf	1.0	< 1	
logf	1.0	< 1	
log10f	1.0	< 1	
log1pf	1.0	< 1	
log2f	1.0	< 1	
powf	1.0	< 1	
sinf	1.0	< 1	
sinhf	1.0	< 1	
sqrtf	0.5	0.500	[1]
tanf	1.0	< 1	
tanhf	1.0	< 1	
tgammaf	1.0	< 1	

Double precision real functions (SPARC)

function	error bound (ulps)	largest error observed (ulps)	notes
acos	1.0	< 1	
acosh	4.0	1.878	
asin	1.0	< 1	
asinh	7.0	1.653	
atan	1.0	< 1	
atan2	2.5	1.456	
atanh	4.0	1.960	
cbrt	1.0	< 1	
cos	1.0	< 1	
cosh	3.0	1.168	
erf	4.0	0.959	
erfc	6.0	2.816	
exp	1.0	< 1	
exp2	2.0	1.050	
expm1	1.0	< 1	
hypot	1.0	< 1	
lgamma	61.5	5.629	[2]
log	1.0	< 1	
log10	3.5	1.592	
log1p	1.0	< 1	
log2	3.5	1.378	
pow	1.0	< 1	
sin	1.0	< 1	
sinh	4.0	2.078	
sqrt	0.5	0.500	[1]
tan	1.0	< 1	

function	error bound (ulps)	largest error observed (ulps)	notes
tanh	3.5	2.136	
tgamma	1.0	< 1	

Double precision real functions (x86)

function	error bound (ulps)	largest error observed (ulps)	notes
acos	1.0	< 1	
acosh	4.0	1.694	
asin	1.0	< 1	
asinh	7.0	1.493	
atan	1.0	< 1	
atan2	1.0	< 1	
atanh	4.0	1.445	
cbrt	1.0	< 1	
cos	1.0	< 1	
cosh	3.0	1.001	
erf	4.0	0.932	
erfc	6.0	2.728	
exp	1.0	< 1	
exp2	1.0	< 1	
expm1	1.0	< 1	
hypot	1.0	< 1	
lgamma	61.5	2.654	[2]
log	1.0	< 1	
log10	1.0	< 1	
log1p	1.0	< 1	
log2	1.0	< 1	
pow	1.0	< 1	

function	error bound (ulps)	largest error observed (ulps)	notes
sin	1.0	< 1	
sinh	4.0	1.458	
sqrt	0.5003	0.500	[1]
tan	1.0	< 1	
tanh	3.5	1.592	
tgamma	1.0	< 1	

Quadruple precision real functions (SPARC)

function	error bound (ulps)	largest error observed (ulps)	notes
acosl	3.5	1.771	
acoshl	8.0	1.275	
asinl	4.0	2.007	
asinhl	9.0	1.823	
atanl	1.0	< 1	
atan2l	2.5	1.102	
atanhl	4.0	1.970	
cbrtl	1.0	< 1	
cosl	1.0	< 1	
coshl	3.5	0.985	
erfl	2.0	0.779	
erfcl	68.5	13.923	
expl	1.0	< 1	
exp2l	2.0	0.714	
expm1l	2.0	1.020	
hypotl	1.0	< 1	
lgammal	18.5	2.916	[2]
logl	1.0	< 1	

function	error bound (ulps)	largest error observed (ulps)	notes
log10l	3.5	1.156	
log1pl	2.0	1.216	
log2l	3.5	1.675	
powl	1.0	< 1	
sinl	1.0	< 1	
sinhl	4.5	1.589	
sqrtl	0.5	0.500	[1]
tanl	4.5	2.380	
tanhl	4.5	1.692	
tgammal	1.0	< 1	

Extended precision real functions (x86)

function	error bound (ulps)	largest error observed (ulps)	notes
acosl	3.0	1.868	
acoshl	8.0	2.352	
asinl	3.0	1.716	
asinhl	9.0	2.346	
atanl	1.0	< 1	
atan2l	1.0	< 1	
atanhl	4.0	2.438	
cbrtl	1.0	< 1	
cosl	1.0	< 1	
coshl	3.5	1.288	
erfl	1.0	< 1	
erfcl	78.5	13.407	
expl	3.5	1.272	
exp2l	1.5	0.786	

function	error bound (ulps)	largest error observed (ulps)	notes
expm1l	4.0	1.921	
hypotl	3.5	2.087	
lgammal	22.5	3.125	[2]
logl	2.0	0.796	
log10l	2.0	1.224	
log1pl	5.0	2.370	
log2l	1.0	< 1	
powl	32770.0	3756.512	
sinl	1.0	< 1	
sinhl	4.5	2.356	
sqrtl	0.5	0.500	[1]
tanl	4.5	2.366	
tanhl	4.5	2.417	
tgammal	1.0	< 1	

Notes:

[1] On SPARC, `sqrtf`, `sqrt`, and `sqrtl` are correctly rounded in accordance with IEEE 754. On x86, `sqrtl` is correctly rounded, `sqrtf` is correctly rounded provided the result is narrowed to single precision as discussed above, but `sqrt` might not be correctly rounded due to "double rounding": when the intermediate value computed to extended precision lies exactly halfway between two representable numbers in double precision, the result of rounding the intermediate value to double precision is determined by the round-ties-to-even rule. If this rule causes the second rounding to round in the same direction as the first, the net rounding error can exceed 0.5 ulps. (The error is bounded instead by 0.5*(1 + 2^-11) ulps.)

[2] Error bounds for lgamma and lgammal apply only for positive arguments.

Complex functions

The real-valued complex functions `cabsf`, `cabs`, `cabsl`, `cargf`, `carg`, and `cargl` are equivalent to the real functions `hypotf`, `hypot`, `hypotl`, `atan2f`, `atan2`, and `atan2l`, respectively. The error bounds and observed errors given above for the latter functions also apply to the former.

The complex functions listed below are complex-valued. For each function, the error bound shown applies separately to both the real and imaginary parts of the result. (For example, both the real and imaginary parts of `cacosf`(`z`) are accurate to within 1 ulp regardless of their magnitudes.) Similarly, the largest observed error shown is the largest error found in either the real or the imaginary part of the result.

Single precision complex functions (SPARC)

function	error bound (ulps)	largest error observed (ulps)	notes
`cacosf, cacoshf`	1	< 1	[1]
`casinf, casinhf`	1	< 1	
`catanf, catanhf`	6	< 1	
`ccosf, ccoshf`	10	2.008	
`cexpf`	3	2.155	
`clogf`	3	< 1	
`cpowf`	—	< 1	[2]
`csinf, csinhf`	10	2.004	
`csqrtf`	4	< 1	
`ctanf, ctanhf`	13	6.141	

Single precision complex functions (x86)

function	error bound (ulps)	largest error observed (ulps)	notes
`cacosf, cacoshf`	1	< 1	[1]
`casinf, casinhf`	1	< 1	
`catanf, catanhf`	6	< 1	
`ccosf, ccoshf`	10	1.984	
`cexpf`	3	1.984	
`clogf`	3	< 1	
`cpowf`	—	< 1	[2]
`csinf, csinhf`	10	1.973	

function	error bound (ulps)	largest error observed (ulps)	notes
csqrtf	4	< 1	
ctanf, ctanhf	13	4.657	

Double precision complex functions (SPARC)

function	error bound (ulps)	largest error observed (ulps)	notes
cacos, cacosh	9	3.337	[1]
casin, casinh	9	3.267	
catan, catanh	6	3.737	
ccos, ccosh	10	3.635	
cexp	3	2.165	
clog	3	2.162	
cpow	-	-	[2]
csin, csinh	10	3.414	
csqrt	4	2.572	
ctan, ctanh	13	6.116	

Double precision complex functions (x86)

function	error bound (ulps)	largest error observed (ulps)	notes
cacos, cacosh	9	3.624	[1]
casin, casinh	9	3.624	
catan, catanh	6	2.500	
ccos, ccosh	10	2.929	
cexp	3	2.147	
clog	3	1.927	
cpow	-	-	[2]

function	error bound (ulps)	largest error observed (ulps)	notes
csin, csinh	10	2.918	
csqrt	4	1.914	
ctan, ctanh	13	4.630	

Quadruple precision complex functions (SPARC)

function	error bound (ulps)	largest error observed (ulps)	notes
cacosl, cacoshl	9	3	[1]
casinl, casinhl	9	3	
catanl, catanhl	6	3	
ccosl, ccoshl	10	3	
cexpl	3	2	
clogl	3	2	
cpowl	-	-	[2]
csinl, csinhl	10	3	
csqrtl	4	3	
ctanl, ctanhl	13	5	

Extended precision complex functions (x86)

function	error bound (ulps)	largest error observed (ulps)	notes
cacosl, cacoshl	9	2	[1]
casinl, casinhl	9	2	
catanl, catanhl	6	2	
ccosl, ccoshl	10	3	
cexpl	3	1	
clogl	3	1	

function	error bound (ulps)	largest error observed (ulps)	notes
cpowl	-	-	[2]
csinl, csinhl	10	3	
csqrtl	4	1	
ctanl, ctanhl	13	5	

Notes:

[1] The complex hyperbolic trigonometric functions are equivalent by symmetries to their circular trigonometric counterparts. Because the implementations of these functions exploit these symmetries, corresponding functions have the same error bounds and observed errors.

[2] For large arguments, the results computed by cpowf, cpow, and cpowl can have unbounded relative error. It might be possible to give error bounds for specific domains, but no such bounds are currently available. The observed errors shown are for the domain {(*z*,*w*) : max(|Re *z*|, |Im *z*|, |Re *w*|, |Im *w*|) <= 1}.

FILES

/lib/libm.so.2	shared object
/lib/64/libm.so.2	64-bit shared object

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWlibmsr
MT-Level	Safe with exceptions

As described on the lgamma(3M) manual page, gamma() and lgamma() and their float and long double counterparts are Unafe. All other functions in libm.so.2 are MT-Safe.

SEE ALSO

intro(3), lgamma(3M), math.h(3HEAD), attributes(5), standards(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.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.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.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

`/lib/libmd5.so.1`	shared object
`/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.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

libmlib – mediaLib library

SYNOPSIS

```
cc [ flag... ] file... -lmlib [ library... ]
 #include <mlib.h>
```

DESCRIPTION

Interfaces in this library provide functions for multimedia processing. When executed on an UltraSPARC platform, these functions take advantage of the VIS Instruction Set.

INTERFACES

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

System Functions

mlib_free
mlib_malloc
mlib_memcpy
mlib_memmove
mlib_memset
mlib_realloc
mlib_version

Algebra Functions

mlib_MatrixAdd_S16C_Mod
mlib_MatrixAdd_S16C_S16C_Mod
mlib_MatrixAdd_S16C_S16C_Sat
mlib_MatrixAdd_S16C_S8C_Mod
mlib_MatrixAdd_S16C_S8C_Sat
mlib_MatrixAdd_S16C_Sat
mlib_MatrixAdd_S16C_U8C_Mod
mlib_MatrixAdd_S16C_U8C_Sat
mlib_MatrixAdd_S16_Mod
mlib_MatrixAdd_S16_S16_Mod
mlib_MatrixAdd_S16_S16_Sat
mlib_MatrixAdd_S16_S8_Mod
mlib_MatrixAdd_S16_S8_Sat
mlib_MatrixAdd_S16_Sat
mlib_MatrixAdd_S16_U8_Mod
mlib_MatrixAdd_S16_U8_Sat
mlib_MatrixAdd_S32C_Mod
mlib_MatrixAdd_S32C_S16C_Mod
mlib_MatrixAdd_S32C_S16C_Sat
mlib_MatrixAdd_S32C_S32C_Mod
mlib_MatrixAdd_S32C_S32C_Sat
mlib_MatrixAdd_S32C_Sat
mlib_MatrixAdd_S32_Mod
mlib_MatrixAdd_S32_S16_Mod
mlib_MatrixAdd_S32_S16_Sat
mlib_MatrixAdd_S32_S32_Mod
mlib_MatrixAdd_S32_S32_Sat
mlib_MatrixAdd_S32_Sat
mlib_MatrixAdd_S8C_Mod
mlib_MatrixAdd_S8C_S8C_Mod

mlib_MatrixAdd_S8C_S8C_Sat
mlib_MatrixAdd_S8C_Sat
mlib_MatrixAdd_S8_Mod
mlib_MatrixAdd_S8_S8_Mod
mlib_MatrixAdd_S8_S8_Sat
mlib_MatrixAdd_S8_Sat
mlib_MatrixAddS_S16C_Mod
mlib_MatrixAddS_S16C_S16C_Mod
mlib_MatrixAddS_S16C_S16C_Sat
mlib_MatrixAddS_S16C_S8C_Mod
mlib_MatrixAddS_S16C_S8C_Sat
mlib_MatrixAddS_S16C_Sat
mlib_MatrixAddS_S16C_U8C_Mod
mlib_MatrixAddS_S16C_U8C_Sat
mlib_MatrixAddS_S16_Mod
mlib_MatrixAddS_S16_S16_Mod
mlib_MatrixAddS_S16_S16_Sat
mlib_MatrixAddS_S16_S8_Mod
mlib_MatrixAddS_S16_S8_Sat
mlib_MatrixAddS_S16_Sat
mlib_MatrixAddS_S16_U8_Mod
mlib_MatrixAddS_S16_U8_Sat
mlib_MatrixAddS_S32C_Mod
mlib_MatrixAddS_S32C_S16C_Mod
mlib_MatrixAddS_S32C_S16C_Sat
mlib_MatrixAddS_S32C_S32C_Mod
mlib_MatrixAddS_S32C_S32C_Sat
mlib_MatrixAddS_S32C_Sat
mlib_MatrixAddS_S32_Mod
mlib_MatrixAddS_S32_S16_Mod
mlib_MatrixAddS_S32_S16_Sat
mlib_MatrixAddS_S32_S32_Mod
mlib_MatrixAddS_S32_S32_Sat
mlib_MatrixAddS_S32_Sat
mlib_MatrixAddS_S8C_Mod
mlib_MatrixAddS_S8C_S8C_Mod
mlib_MatrixAddS_S8C_S8C_Sat
mlib_MatrixAddS_S8C_Sat
mlib_MatrixAddS_S8_Mod
mlib_MatrixAddS_S8_S8_Mod
mlib_MatrixAddS_S8_S8_Sat
mlib_MatrixAddS_S8_Sat
mlib_MatrixAddS_U8C_Mod
mlib_MatrixAddS_U8C_Sat
mlib_MatrixAddS_U8C_U8C_Mod
mlib_MatrixAddS_U8C_U8C_Sat
mlib_MatrixAddS_U8_Mod
mlib_MatrixAddS_U8_Sat

mlib_MatrixAddS_U8_U8_Mod
mlib_MatrixAddS_U8_U8_Sat
mlib_MatrixAdd_U8C_Mod
mlib_MatrixAdd_U8C_Sat
mlib_MatrixAdd_U8C_U8C_Mod
mlib_MatrixAdd_U8C_U8C_Sat
mlib_MatrixAdd_U8_Mod
mlib_MatrixAdd_U8_Sat
mlib_MatrixAdd_U8_U8_Mod
mlib_MatrixAdd_U8_U8_Sat
mlib_MatrixMaximum_D64
mlib_MatrixMaximum_F32
mlib_MatrixMaximumMag_D64C
mlib_MatrixMaximumMag_F32C
mlib_MatrixMaximumMag_S16C
mlib_MatrixMaximumMag_S32C
mlib_MatrixMaximumMag_S8C
mlib_MatrixMaximumMag_U8C
mlib_MatrixMaximum_S16
mlib_MatrixMaximum_S32
mlib_MatrixMaximum_S8
mlib_MatrixMaximum_U8
mlib_MatrixMinimum_D64
mlib_MatrixMinimum_F32
mlib_MatrixMinimumMag_D64C
mlib_MatrixMinimumMag_F32C
mlib_MatrixMinimumMag_S16C
mlib_MatrixMinimumMag_S32C
mlib_MatrixMinimumMag_S8C
mlib_MatrixMinimumMag_U8C
mlib_MatrixMinimum_S16
mlib_MatrixMinimum_S32
mlib_MatrixMinimum_S8
mlib_MatrixMinimum_U8
mlib_MatrixMul_S16C_S16C_Mod
mlib_MatrixMul_S16C_S16C_Sat
mlib_MatrixMul_S16C_S8C_Mod
mlib_MatrixMul_S16C_S8C_Sat
mlib_MatrixMul_S16C_U8C_Mod
mlib_MatrixMul_S16C_U8C_Sat
mlib_MatrixMul_S16_S16_Mod
mlib_MatrixMul_S16_S16_Sat
mlib_MatrixMul_S16_S8_Mod
mlib_MatrixMul_S16_S8_Sat
mlib_MatrixMul_S16_U8_Mod
mlib_MatrixMul_S16_U8_Sat
mlib_MatrixMul_S32C_S16C_Mod
mlib_MatrixMul_S32C_S16C_Sat

mlib_MatrixMul_S32C_S32C_Mod
mlib_MatrixMul_S32C_S32C_Sat
mlib_MatrixMul_S32_S16_Mod
mlib_MatrixMul_S32_S16_Sat
mlib_MatrixMul_S32_S32_Mod
mlib_MatrixMul_S32_S32_Sat
mlib_MatrixMul_S8C_S8C_Mod
mlib_MatrixMul_S8C_S8C_Sat
mlib_MatrixMul_S8_S8_Mod
mlib_MatrixMul_S8_S8_Sat
mlib_MatrixMulShift_S16C_S16C_Mod
mlib_MatrixMulShift_S16C_S16C_Sat
mlib_MatrixMulShift_S16_S16_Mod
mlib_MatrixMulShift_S16_S16_Sat
mlib_MatrixMulS_S16C_Mod
mlib_MatrixMulS_S16C_S16C_Mod
mlib_MatrixMulS_S16C_S16C_Sat
mlib_MatrixMulS_S16C_S8C_Mod
mlib_MatrixMulS_S16C_S8C_Sat
mlib_MatrixMulS_S16C_Sat
mlib_MatrixMulS_S16C_U8C_Mod
mlib_MatrixMulS_S16C_U8C_Sat
mlib_MatrixMulS_S16_Mod
mlib_MatrixMulS_S16_S16_Mod
mlib_MatrixMulS_S16_S16_Sat
mlib_MatrixMulS_S16_S8_Mod
mlib_MatrixMulS_S16_S8_Sat
mlib_MatrixMulS_S16_Sat
mlib_MatrixMulS_S16_U8_Mod
mlib_MatrixMulS_S16_U8_Sat
mlib_MatrixMulS_S32C_Mod
mlib_MatrixMulS_S32C_S16C_Mod
mlib_MatrixMulS_S32C_S16C_Sat
mlib_MatrixMulS_S32C_S32C_Mod
mlib_MatrixMulS_S32C_S32C_Sat
mlib_MatrixMulS_S32C_Sat
mlib_MatrixMulS_S32_Mod
mlib_MatrixMulS_S32_S16_Mod
mlib_MatrixMulS_S32_S16_Sat
mlib_MatrixMulS_S32_S32_Mod
mlib_MatrixMulS_S32_S32_Sat
mlib_MatrixMulS_S32_Sat
mlib_MatrixMulS_S8C_Mod
mlib_MatrixMulS_S8C_S8C_Mod
mlib_MatrixMulS_S8C_S8C_Sat
mlib_MatrixMulS_S8C_Sat
mlib_MatrixMulS_S8_Mod
mlib_MatrixMulS_S8_S8_Mod

mlib_MatrixMulS_S8_S8_Sat
mlib_MatrixMulS_S8_Sat
mlib_MatrixMulSShift_S16C_Mod
mlib_MatrixMulSShift_S16C_S16C_Mod
mlib_MatrixMulSShift_S16C_S16C_Sat
mlib_MatrixMulSShift_S16C_Sat
mlib_MatrixMulSShift_S16_Mod
mlib_MatrixMulSShift_S16_S16_Mod
mlib_MatrixMulSShift_S16_S16_Sat
mlib_MatrixMulSShift_S16_Sat
mlib_MatrixMulSShift_S32C_Mod
mlib_MatrixMulSShift_S32C_S32C_Mod
mlib_MatrixMulSShift_S32C_S32C_Sat
mlib_MatrixMulSShift_S32C_Sat
mlib_MatrixMulSShift_S32_Mod
mlib_MatrixMulSShift_S32_S32_Mod
mlib_MatrixMulSShift_S32_S32_Sat
mlib_MatrixMulSShift_S32_Sat
mlib_MatrixMulSShift_S8C_Mod
mlib_MatrixMulSShift_S8C_S8C_Mod
mlib_MatrixMulSShift_S8C_S8C_Sat
mlib_MatrixMulSShift_S8C_Sat
mlib_MatrixMulSShift_S8_Mod
mlib_MatrixMulSShift_S8_S8_Mod
mlib_MatrixMulSShift_S8_S8_Sat
mlib_MatrixMulSShift_S8_Sat
mlib_MatrixMulSShift_U8C_Mod
mlib_MatrixMulSShift_U8C_Sat
mlib_MatrixMulSShift_U8C_U8C_Mod
mlib_MatrixMulSShift_U8C_U8C_Sat
mlib_MatrixMulSShift_U8_Mod
mlib_MatrixMulSShift_U8_Sat
mlib_MatrixMulSShift_U8_U8_Mod
mlib_MatrixMulSShift_U8_U8_Sat
mlib_MatrixMulS_U8C_Mod
mlib_MatrixMulS_U8C_Sat
mlib_MatrixMulS_U8C_U8C_Mod
mlib_MatrixMulS_U8C_U8C_Sat
mlib_MatrixMulS_U8_Mod
mlib_MatrixMulS_U8_Sat
mlib_MatrixMulS_U8_U8_Mod
mlib_MatrixMulS_U8_U8_Sat
mlib_MatrixMul_U8C_U8C_Mod
mlib_MatrixMul_U8C_U8C_Sat
mlib_MatrixMul_U8_U8_Mod
mlib_MatrixMul_U8_U8_Sat
mlib_MatrixScale_S16C_Mod
mlib_MatrixScale_S16C_S16C_Mod

mlib_MatrixScale_S16C_S16C_Sat
mlib_MatrixScale_S16C_S8C_Mod
mlib_MatrixScale_S16C_S8C_Sat
mlib_MatrixScale_S16C_Sat
mlib_MatrixScale_S16C_U8C_Mod
mlib_MatrixScale_S16C_U8C_Sat
mlib_MatrixScale_S16_Mod
mlib_MatrixScale_S16_S16_Mod
mlib_MatrixScale_S16_S16_Sat
mlib_MatrixScale_S16_S8_Mod
mlib_MatrixScale_S16_S8_Sat
mlib_MatrixScale_S16_Sat
mlib_MatrixScale_S16_U8_Mod
mlib_MatrixScale_S16_U8_Sat
mlib_MatrixScale_S32C_Mod
mlib_MatrixScale_S32C_S16C_Mod
mlib_MatrixScale_S32C_S16C_Sat
mlib_MatrixScale_S32C_S32C_Mod
mlib_MatrixScale_S32C_S32C_Sat
mlib_MatrixScale_S32C_Sat
mlib_MatrixScale_S32_Mod
mlib_MatrixScale_S32_S16_Mod
mlib_MatrixScale_S32_S16_Sat
mlib_MatrixScale_S32_S32_Mod
mlib_MatrixScale_S32_S32_Sat
mlib_MatrixScale_S32_Sat
mlib_MatrixScale_S8C_Mod
mlib_MatrixScale_S8C_S8C_Mod
mlib_MatrixScale_S8C_S8C_Sat
mlib_MatrixScale_S8C_Sat
mlib_MatrixScale_S8_Mod
mlib_MatrixScale_S8_S8_Mod
mlib_MatrixScale_S8_S8_Sat
mlib_MatrixScale_S8_Sat
mlib_MatrixScale_U8C_Mod
mlib_MatrixScale_U8C_Sat
mlib_MatrixScale_U8C_U8C_Mod
mlib_MatrixScale_U8C_U8C_Sat
mlib_MatrixScale_U8_Mod
mlib_MatrixScale_U8_Sat
mlib_MatrixScale_U8_U8_Mod
mlib_MatrixScale_U8_U8_Sat
mlib_MatrixSub_S16C_Mod
mlib_MatrixSub_S16C_S16C_Mod
mlib_MatrixSub_S16C_S16C_Sat
mlib_MatrixSub_S16C_S8C_Mod
mlib_MatrixSub_S16C_S8C_Sat
mlib_MatrixSub_S16C_Sat

mlib_MatrixSub_S16C_U8C_Mod
mlib_MatrixSub_S16C_U8C_Sat
mlib_MatrixSub_S16_Mod
mlib_MatrixSub_S16_S16_Mod
mlib_MatrixSub_S16_S16_Sat
mlib_MatrixSub_S16_S8_Mod
mlib_MatrixSub_S16_S8_Sat
mlib_MatrixSub_S16_Sat
mlib_MatrixSub_S16_U8_Mod
mlib_MatrixSub_S16_U8_Sat
mlib_MatrixSub_S32C_Mod
mlib_MatrixSub_S32C_S16C_Mod
mlib_MatrixSub_S32C_S16C_Sat
mlib_MatrixSub_S32C_S32C_Mod
mlib_MatrixSub_S32C_S32C_Sat
mlib_MatrixSub_S32C_Sat
mlib_MatrixSub_S32_Mod
mlib_MatrixSub_S32_S16_Mod
mlib_MatrixSub_S32_S16_Sat
mlib_MatrixSub_S32_S32_Mod
mlib_MatrixSub_S32_S32_Sat
mlib_MatrixSub_S32_Sat
mlib_MatrixSub_S8C_Mod
mlib_MatrixSub_S8C_S8C_Mod
mlib_MatrixSub_S8C_S8C_Sat
mlib_MatrixSub_S8C_Sat
mlib_MatrixSub_S8_Mod
mlib_MatrixSub_S8_S8_Mod
mlib_MatrixSub_S8_S8_Sat
mlib_MatrixSub_S8_Sat
mlib_MatrixSubS_S16C_Mod
mlib_MatrixSubS_S16C_S16C_Mod
mlib_MatrixSubS_S16C_S16C_Sat
mlib_MatrixSubS_S16C_S8C_Mod
mlib_MatrixSubS_S16C_S8C_Sat
mlib_MatrixSubS_S16C_Sat
mlib_MatrixSubS_S16C_U8C_Mod
mlib_MatrixSubS_S16C_U8C_Sat
mlib_MatrixSubS_S16_Mod
mlib_MatrixSubS_S16_S16_Mod
mlib_MatrixSubS_S16_S16_Sat
mlib_MatrixSubS_S16_S8_Mod
mlib_MatrixSubS_S16_S8_Sat
mlib_MatrixSubS_S16_Sat
mlib_MatrixSubS_S16_U8_Mod
mlib_MatrixSubS_S16_U8_Sat
mlib_MatrixSubS_S32C_Mod
mlib_MatrixSubS_S32C_S16C_Mod

mlib_MatrixSubS_S32C_S16C_Sat
mlib_MatrixSubS_S32C_S32C_Mod
mlib_MatrixSubS_S32C_S32C_Sat
mlib_MatrixSubS_S32C_Sat
mlib_MatrixSubS_S32_Mod
mlib_MatrixSubS_S32_S16_Mod
mlib_MatrixSubS_S32_S16_Sat
mlib_MatrixSubS_S32_S32_Mod
mlib_MatrixSubS_S32_S32_Sat
mlib_MatrixSubS_S32_Sat
mlib_MatrixSubS_S8C_Mod
mlib_MatrixSubS_S8C_S8C_Mod
mlib_MatrixSubS_S8C_S8C_Sat
mlib_MatrixSubS_S8C_Sat
mlib_MatrixSubS_S8_Mod
mlib_MatrixSubS_S8_S8_Mod
mlib_MatrixSubS_S8_S8_Sat
mlib_MatrixSubS_S8_Sat
mlib_MatrixSubS_U8C_Mod
mlib_MatrixSubS_U8C_Sat
mlib_MatrixSubS_U8C_U8C_Mod
mlib_MatrixSubS_U8C_U8C_Sat
mlib_MatrixSubS_U8_Mod
mlib_MatrixSubS_U8_Sat
mlib_MatrixSubS_U8_U8_Mod
mlib_MatrixSubS_U8_U8_Sat
mlib_MatrixSub_U8C_Mod
mlib_MatrixSub_U8C_Sat
mlib_MatrixSub_U8C_U8C_Mod
mlib_MatrixSub_U8C_U8C_Sat
mlib_MatrixSub_U8_Mod
mlib_MatrixSub_U8_Sat
mlib_MatrixSub_U8_U8_Mod
mlib_MatrixSub_U8_U8_Sat
mlib_MatrixTranspose_S16
mlib_MatrixTranspose_S16C
mlib_MatrixTranspose_S16C_S16C
mlib_MatrixTranspose_S16_S16
mlib_MatrixTranspose_S32
mlib_MatrixTranspose_S32C
mlib_MatrixTranspose_S32C_S32C
mlib_MatrixTranspose_S32_S32
mlib_MatrixTranspose_S8
mlib_MatrixTranspose_S8C
mlib_MatrixTranspose_S8C_S8C
mlib_MatrixTranspose_S8_S8
mlib_MatrixTranspose_U8
mlib_MatrixTranspose_U8C

mlib_MatrixTranspose_U8C_U8C
mlib_MatrixTranspose_U8_U8
mlib_MatrixUnit_S16
mlib_MatrixUnit_S16C
mlib_MatrixUnit_S32
mlib_MatrixUnit_S32C
mlib_MatrixUnit_S8
mlib_MatrixUnit_S8C
mlib_MatrixUnit_U8
mlib_MatrixUnit_U8C
mlib_VectorAdd_S16C_Mod
mlib_VectorAdd_S16C_S16C_Mod
mlib_VectorAdd_S16C_S16C_Sat
mlib_VectorAdd_S16C_S8C_Mod
mlib_VectorAdd_S16C_S8C_Sat
mlib_VectorAdd_S16C_Sat
mlib_VectorAdd_S16C_U8C_Mod
mlib_VectorAdd_S16C_U8C_Sat
mlib_VectorAdd_S16_Mod
mlib_VectorAdd_S16_S16_Mod
mlib_VectorAdd_S16_S16_Sat
mlib_VectorAdd_S16_S8_Mod
mlib_VectorAdd_S16_S8_Sat
mlib_VectorAdd_S16_Sat
mlib_VectorAdd_S16_U8_Mod
mlib_VectorAdd_S16_U8_Sat
mlib_VectorAdd_S32C_Mod
mlib_VectorAdd_S32C_S16C_Mod
mlib_VectorAdd_S32C_S16C_Sat
mlib_VectorAdd_S32C_S32C_Mod
mlib_VectorAdd_S32C_S32C_Sat
mlib_VectorAdd_S32C_Sat
mlib_VectorAdd_S32_Mod
mlib_VectorAdd_S32_S16_Mod
mlib_VectorAdd_S32_S16_Sat
mlib_VectorAdd_S32_S32_Mod
mlib_VectorAdd_S32_S32_Sat
mlib_VectorAdd_S32_Sat
mlib_VectorAdd_S8C_Mod
mlib_VectorAdd_S8C_S8C_Mod
mlib_VectorAdd_S8C_S8C_Sat
mlib_VectorAdd_S8C_Sat
mlib_VectorAdd_S8_Mod
mlib_VectorAdd_S8_S8_Mod
mlib_VectorAdd_S8_S8_Sat
mlib_VectorAdd_S8_Sat
mlib_VectorAddS_S16C_Mod
mlib_VectorAddS_S16C_S16C_Mod

mlib_VectorAddS_S16C_S16C_Sat
mlib_VectorAddS_S16C_S8C_Mod
mlib_VectorAddS_S16C_S8C_Sat
mlib_VectorAddS_S16C_Sat
mlib_VectorAddS_S16C_U8C_Mod
mlib_VectorAddS_S16C_U8C_Sat
mlib_VectorAddS_S16_Mod
mlib_VectorAddS_S16_S16_Mod
mlib_VectorAddS_S16_S16_Sat
mlib_VectorAddS_S16_S8_Mod
mlib_VectorAddS_S16_S8_Sat
mlib_VectorAddS_S16_Sat
mlib_VectorAddS_S16_U8_Mod
mlib_VectorAddS_S16_U8_Sat
mlib_VectorAddS_S32C_Mod
mlib_VectorAddS_S32C_S16C_Mod
mlib_VectorAddS_S32C_S16C_Sat
mlib_VectorAddS_S32C_S32C_Mod
mlib_VectorAddS_S32C_S32C_Sat
mlib_VectorAddS_S32C_Sat
mlib_VectorAddS_S32_Mod
mlib_VectorAddS_S32_S16_Mod
mlib_VectorAddS_S32_S16_Sat
mlib_VectorAddS_S32_S32_Mod
mlib_VectorAddS_S32_S32_Sat
mlib_VectorAddS_S32_Sat
mlib_VectorAddS_S8C_Mod
mlib_VectorAddS_S8C_S8C_Mod
mlib_VectorAddS_S8C_S8C_Sat
mlib_VectorAddS_S8C_Sat
mlib_VectorAddS_S8_Mod
mlib_VectorAddS_S8_S8_Mod
mlib_VectorAddS_S8_S8_Sat
mlib_VectorAddS_S8_Sat
mlib_VectorAddS_U8C_Mod
mlib_VectorAddS_U8C_Sat
mlib_VectorAddS_U8C_U8C_Mod
mlib_VectorAddS_U8C_U8C_Sat
mlib_VectorAddS_U8_Mod
mlib_VectorAddS_U8_Sat
mlib_VectorAddS_U8_U8_Mod
mlib_VectorAddS_U8_U8_Sat
mlib_VectorAdd_U8C_Mod
mlib_VectorAdd_U8C_Sat
mlib_VectorAdd_U8C_U8C_Mod
mlib_VectorAdd_U8C_U8C_Sat
mlib_VectorAdd_U8_Mod
mlib_VectorAdd_U8_Sat

mlib_VectorAdd_U8_U8_Mod
mlib_VectorAdd_U8_U8_Sat
mlib_VectorAng_S16C
mlib_VectorAng_S32C
mlib_VectorAng_S8C
mlib_VectorAng_U8C
mlib_VectorConjRev_S16C_S16C_Sat
mlib_VectorConjRev_S32C_S32C_Sat
mlib_VectorConjRev_S8C_S8C_Sat
mlib_VectorConj_S16C_S16C_Sat
mlib_VectorConj_S16C_Sat
mlib_VectorConj_S32C_S32C_Sat
mlib_VectorConj_S32C_Sat
mlib_VectorConj_S8C_S8C_Sat
mlib_VectorConj_S8C_Sat
mlib_VectorConjSymExt_S16C_S16C_Sat
mlib_VectorConjSymExt_S32C_S32C_Sat
mlib_VectorConjSymExt_S8C_S8C_Sat
mlib_VectorConvert_S16C_S32C_Mod
mlib_VectorConvert_S16C_S32C_Sat
mlib_VectorConvert_S16C_S8C_Mod
mlib_VectorConvert_S16C_S8C_Sat
mlib_VectorConvert_S16C_U8C_Mod
mlib_VectorConvert_S16C_U8C_Sat
mlib_VectorConvert_S16_S32_Mod
mlib_VectorConvert_S16_S32_Sat
mlib_VectorConvert_S16_S8_Mod
mlib_VectorConvert_S16_S8_Sat
mlib_VectorConvert_S16_U8_Mod
mlib_VectorConvert_S16_U8_Sat
mlib_VectorConvert_S32C_S16C_Mod
mlib_VectorConvert_S32C_S16C_Sat
mlib_VectorConvert_S32C_S8C_Mod
mlib_VectorConvert_S32C_S8C_Sat
mlib_VectorConvert_S32C_U8C_Mod
mlib_VectorConvert_S32C_U8C_Sat
mlib_VectorConvert_S32_S16_Mod
mlib_VectorConvert_S32_S16_Sat
mlib_VectorConvert_S32_S8_Mod
mlib_VectorConvert_S32_S8_Sat
mlib_VectorConvert_S32_U8_Mod
mlib_VectorConvert_S32_U8_Sat
mlib_VectorConvert_S8C_S16C_Mod
mlib_VectorConvert_S8C_S16C_Sat
mlib_VectorConvert_S8C_S32C_Mod
mlib_VectorConvert_S8C_S32C_Sat
mlib_VectorConvert_S8C_U8C_Mod
mlib_VectorConvert_S8C_U8C_Sat

mlib_VectorConvert_S8_S16_Mod
mlib_VectorConvert_S8_S16_Sat
mlib_VectorConvert_S8_S32_Mod
mlib_VectorConvert_S8_S32_Sat
mlib_VectorConvert_S8_U8_Mod
mlib_VectorConvert_S8_U8_Sat
mlib_VectorConvert_U8C_S16C_Mod
mlib_VectorConvert_U8C_S16C_Sat
mlib_VectorConvert_U8C_S32C_Mod
mlib_VectorConvert_U8C_S32C_Sat
mlib_VectorConvert_U8C_S8C_Mod
mlib_VectorConvert_U8C_S8C_Sat
mlib_VectorConvert_U8_S16_Mod
mlib_VectorConvert_U8_S16_Sat
mlib_VectorConvert_U8_S32_Mod
mlib_VectorConvert_U8_S32_Sat
mlib_VectorConvert_U8_S8_Mod
mlib_VectorConvert_U8_S8_Sat
mlib_VectorCopy_S16
mlib_VectorCopy_S16C
mlib_VectorCopy_S32
mlib_VectorCopy_S32C
mlib_VectorCopy_S8
mlib_VectorCopy_S8C
mlib_VectorCopy_U8
mlib_VectorCopy_U8C
mlib_VectorDistance_S16_Sat
mlib_VectorDistance_S32_Sat
mlib_VectorDistance_S8_Sat
mlib_VectorDistance_U8_Sat
mlib_VectorDotProd_S16C_Sat
mlib_VectorDotProd_S16_Sat
mlib_VectorDotProd_S32C_Sat
mlib_VectorDotProd_S32_Sat
mlib_VectorDotProd_S8C_Sat
mlib_VectorDotProd_S8_Sat
mlib_VectorDotProd_U8C_Sat
mlib_VectorDotProd_U8_Sat
mlib_VectorMag_S16C
mlib_VectorMag_S32C
mlib_VectorMag_S8C
mlib_VectorMag_U8C
mlib_VectorMaximum_D64
mlib_VectorMaximum_F32
mlib_VectorMaximumMag_D64C
mlib_VectorMaximumMag_F32C
mlib_VectorMaximumMag_S16C
mlib_VectorMaximumMag_S32C

mlib_VectorMaximumMag_S8C
mlib_VectorMaximumMag_U8C
mlib_VectorMaximum_S16
mlib_VectorMaximum_S32
mlib_VectorMaximum_S8
mlib_VectorMaximum_U8
mlib_VectorMerge_S16C_S16
mlib_VectorMerge_S32C_S32
mlib_VectorMerge_S8C_S8
mlib_VectorMerge_U8C_U8
mlib_VectorMinimum_D64
mlib_VectorMinimum_F32
mlib_VectorMinimumMag_D64C
mlib_VectorMinimumMag_F32C
mlib_VectorMinimumMag_S16C
mlib_VectorMinimumMag_S32C
mlib_VectorMinimumMag_S8C
mlib_VectorMinimumMag_U8C
mlib_VectorMinimum_S16
mlib_VectorMinimum_S32
mlib_VectorMinimum_S8
mlib_VectorMinimum_U8
mlib_VectorMulM_S16C_S16C_Mod
mlib_VectorMulM_S16C_S16C_Sat
mlib_VectorMulM_S16C_S8C_Mod
mlib_VectorMulM_S16C_S8C_Sat
mlib_VectorMulM_S16C_U8C_Mod
mlib_VectorMulM_S16C_U8C_Sat
mlib_VectorMulM_S16_S16_Mod
mlib_VectorMulM_S16_S16_Sat
mlib_VectorMulM_S16_S8_Mod
mlib_VectorMulM_S16_S8_Sat
mlib_VectorMulM_S16_U8_Mod
mlib_VectorMulM_S16_U8_Sat
mlib_VectorMulM_S32C_S16C_Mod
mlib_VectorMulM_S32C_S16C_Sat
mlib_VectorMulM_S32C_S32C_Mod
mlib_VectorMulM_S32C_S32C_Sat
mlib_VectorMulM_S32_S16_Mod
mlib_VectorMulM_S32_S16_Sat
mlib_VectorMulM_S32_S32_Mod
mlib_VectorMulM_S32_S32_Sat
mlib_VectorMulM_S8C_S8C_Mod
mlib_VectorMulM_S8C_S8C_Sat
mlib_VectorMulM_S8_S8_Mod
mlib_VectorMulM_S8_S8_Sat
mlib_VectorMulMShift_S16C_S16C_Mod
mlib_VectorMulMShift_S16C_S16C_Sat

mlib_VectorMulMShift_S16_S16_Mod
mlib_VectorMulMShift_S16_S16_Sat
mlib_VectorMulM_U8C_U8C_Mod
mlib_VectorMulM_U8C_U8C_Sat
mlib_VectorMulM_U8_U8_Mod
mlib_VectorMulM_U8_U8_Sat
mlib_VectorMul_S16C_Mod
mlib_VectorMul_S16C_S16C_Mod
mlib_VectorMul_S16C_S16C_Sat
mlib_VectorMul_S16C_S8C_Mod
mlib_VectorMul_S16C_S8C_Sat
mlib_VectorMul_S16C_Sat
mlib_VectorMul_S16C_U8C_Mod
mlib_VectorMul_S16C_U8C_Sat
mlib_VectorMul_S16_Mod
mlib_VectorMul_S16_S16_Mod
mlib_VectorMul_S16_S16_Sat
mlib_VectorMul_S16_S8_Mod
mlib_VectorMul_S16_S8_Sat
mlib_VectorMul_S16_Sat
mlib_VectorMul_S16_U8_Mod
mlib_VectorMul_S16_U8_Sat
mlib_VectorMul_S32C_Mod
mlib_VectorMul_S32C_S16C_Mod
mlib_VectorMul_S32C_S16C_Sat
mlib_VectorMul_S32C_S32C_Mod
mlib_VectorMul_S32C_S32C_Sat
mlib_VectorMul_S32C_Sat
mlib_VectorMul_S32_Mod
mlib_VectorMul_S32_S16_Mod
mlib_VectorMul_S32_S16_Sat
mlib_VectorMul_S32_S32_Mod
mlib_VectorMul_S32_S32_Sat
mlib_VectorMul_S32_Sat
mlib_VectorMul_S8C_Mod
mlib_VectorMul_S8C_S8C_Mod
mlib_VectorMul_S8C_S8C_Sat
mlib_VectorMul_S8C_Sat
mlib_VectorMul_S8_Mod
mlib_VectorMul_S8_S8_Mod
mlib_VectorMul_S8_S8_Sat
mlib_VectorMul_S8_Sat
mlib_VectorMulSAdd_S16C_Mod
mlib_VectorMulSAdd_S16C_S16C_Mod
mlib_VectorMulSAdd_S16C_S16C_Sat
mlib_VectorMulSAdd_S16C_S8C_Mod
mlib_VectorMulSAdd_S16C_S8C_Sat
mlib_VectorMulSAdd_S16C_Sat

mlib_VectorMulSAdd_S16C_U8C_Mod
mlib_VectorMulSAdd_S16C_U8C_Sat
mlib_VectorMulSAdd_S16_Mod
mlib_VectorMulSAdd_S16_S16_Mod
mlib_VectorMulSAdd_S16_S16_Sat
mlib_VectorMulSAdd_S16_S8_Mod
mlib_VectorMulSAdd_S16_S8_Sat
mlib_VectorMulSAdd_S16_Sat
mlib_VectorMulSAdd_S16_U8_Mod
mlib_VectorMulSAdd_S16_U8_Sat
mlib_VectorMulSAdd_S32C_Mod
mlib_VectorMulSAdd_S32C_S16C_Mod
mlib_VectorMulSAdd_S32C_S16C_Sat
mlib_VectorMulSAdd_S32C_S32C_Mod
mlib_VectorMulSAdd_S32C_S32C_Sat
mlib_VectorMulSAdd_S32C_Sat
mlib_VectorMulSAdd_S32_Mod
mlib_VectorMulSAdd_S32_S16_Mod
mlib_VectorMulSAdd_S32_S16_Sat
mlib_VectorMulSAdd_S32_S32_Mod
mlib_VectorMulSAdd_S32_S32_Sat
mlib_VectorMulSAdd_S32_Sat
mlib_VectorMulSAdd_S8C_Mod
mlib_VectorMulSAdd_S8C_S8C_Mod
mlib_VectorMulSAdd_S8C_S8C_Sat
mlib_VectorMulSAdd_S8C_Sat
mlib_VectorMulSAdd_S8_Mod
mlib_VectorMulSAdd_S8_S8_Mod
mlib_VectorMulSAdd_S8_S8_Sat
mlib_VectorMulSAdd_S8_Sat
mlib_VectorMulSAdd_U8C_Mod
mlib_VectorMulSAdd_U8C_Sat
mlib_VectorMulSAdd_U8C_U8C_Mod
mlib_VectorMulSAdd_U8C_U8C_Sat
mlib_VectorMulSAdd_U8_Mod
mlib_VectorMulSAdd_U8_Sat
mlib_VectorMulSAdd_U8_U8_Mod
mlib_VectorMulSAdd_U8_U8_Sat
mlib_VectorMulShift_S16C_Mod
mlib_VectorMulShift_S16C_S16C_Mod
mlib_VectorMulShift_S16C_S16C_Sat
mlib_VectorMulShift_S16C_Sat
mlib_VectorMulShift_S16_Mod
mlib_VectorMulShift_S16_S16_Mod
mlib_VectorMulShift_S16_S16_Sat
mlib_VectorMulShift_S16_Sat
mlib_VectorMulShift_S32C_Mod
mlib_VectorMulShift_S32C_S32C_Mod

mlib_VectorMulShift_S32C_S32C_Sat
mlib_VectorMulShift_S32C_Sat
mlib_VectorMulShift_S32_Mod
mlib_VectorMulShift_S32_S32_Mod
mlib_VectorMulShift_S32_S32_Sat
mlib_VectorMulShift_S32_Sat
mlib_VectorMulShift_S8C_Mod
mlib_VectorMulShift_S8C_S8C_Mod
mlib_VectorMulShift_S8C_S8C_Sat
mlib_VectorMulShift_S8C_Sat
mlib_VectorMulShift_S8_Mod
mlib_VectorMulShift_S8_S8_Mod
mlib_VectorMulShift_S8_S8_Sat
mlib_VectorMulShift_S8_Sat
mlib_VectorMulShift_U8C_Mod
mlib_VectorMulShift_U8C_Sat
mlib_VectorMulShift_U8C_U8C_Mod
mlib_VectorMulShift_U8C_U8C_Sat
mlib_VectorMulShift_U8_Mod
mlib_VectorMulShift_U8_Sat
mlib_VectorMulShift_U8_U8_Mod
mlib_VectorMulShift_U8_U8_Sat
mlib_VectorMulS_S16C_Mod
mlib_VectorMulS_S16C_S16C_Mod
mlib_VectorMulS_S16C_S16C_Sat
mlib_VectorMulS_S16C_S8C_Mod
mlib_VectorMulS_S16C_S8C_Sat
mlib_VectorMulS_S16C_Sat
mlib_VectorMulS_S16C_U8C_Mod
mlib_VectorMulS_S16C_U8C_Sat
mlib_VectorMulS_S16_Mod
mlib_VectorMulS_S16_S16_Mod
mlib_VectorMulS_S16_S16_Sat
mlib_VectorMulS_S16_S8_Mod
mlib_VectorMulS_S16_S8_Sat
mlib_VectorMulS_S16_Sat
mlib_VectorMulS_S16_U8_Mod
mlib_VectorMulS_S16_U8_Sat
mlib_VectorMulS_S32C_Mod
mlib_VectorMulS_S32C_S16C_Mod
mlib_VectorMulS_S32C_S16C_Sat
mlib_VectorMulS_S32C_S32C_Mod
mlib_VectorMulS_S32C_S32C_Sat
mlib_VectorMulS_S32C_Sat
mlib_VectorMulS_S32_Mod
mlib_VectorMulS_S32_S16_Mod
mlib_VectorMulS_S32_S16_Sat
mlib_VectorMulS_S32_S32_Mod

mlib_VectorMulS_S32_S32_Sat
mlib_VectorMulS_S32_Sat
mlib_VectorMulS_S8C_Mod
mlib_VectorMulS_S8C_S8C_Mod
mlib_VectorMulS_S8C_S8C_Sat
mlib_VectorMulS_S8C_Sat
mlib_VectorMulS_S8_Mod
mlib_VectorMulS_S8_S8_Mod
mlib_VectorMulS_S8_S8_Sat
mlib_VectorMulS_S8_Sat
mlib_VectorMulSShift_S16C_Mod
mlib_VectorMulSShift_S16C_S16C_Mod
mlib_VectorMulSShift_S16C_S16C_Sat
mlib_VectorMulSShift_S16C_Sat
mlib_VectorMulSShift_S16_Mod
mlib_VectorMulSShift_S16_S16_Mod
mlib_VectorMulSShift_S16_S16_Sat
mlib_VectorMulSShift_S16_Sat
mlib_VectorMulSShift_S32C_Mod
mlib_VectorMulSShift_S32C_S32C_Mod
mlib_VectorMulSShift_S32C_S32C_Sat
mlib_VectorMulSShift_S32C_Sat
mlib_VectorMulSShift_S32_Mod
mlib_VectorMulSShift_S32_S32_Mod
mlib_VectorMulSShift_S32_S32_Sat
mlib_VectorMulSShift_S32_Sat
mlib_VectorMulSShift_S8C_Mod
mlib_VectorMulSShift_S8C_S8C_Mod
mlib_VectorMulSShift_S8C_S8C_Sat
mlib_VectorMulSShift_S8C_Sat
mlib_VectorMulSShift_S8_Mod
mlib_VectorMulSShift_S8_S8_Mod
mlib_VectorMulSShift_S8_S8_Sat
mlib_VectorMulSShift_S8_Sat
mlib_VectorMulSShift_U8C_Mod
mlib_VectorMulSShift_U8C_Sat
mlib_VectorMulSShift_U8C_U8C_Mod
mlib_VectorMulSShift_U8C_U8C_Sat
mlib_VectorMulSShift_U8_Mod
mlib_VectorMulSShift_U8_Sat
mlib_VectorMulSShift_U8_U8_Mod
mlib_VectorMulSShift_U8_U8_Sat
mlib_VectorMulS_U8C_Mod
mlib_VectorMulS_U8C_Sat
mlib_VectorMulS_U8C_U8C_Mod
mlib_VectorMulS_U8C_U8C_Sat
mlib_VectorMulS_U8_Mod
mlib_VectorMulS_U8_Sat

mlib_VectorMulS_U8_U8_Mod
mlib_VectorMulS_U8_U8_Sat
mlib_VectorMul_U8C_Mod
mlib_VectorMul_U8C_Sat
mlib_VectorMul_U8C_U8C_Mod
mlib_VectorMul_U8C_U8C_Sat
mlib_VectorMul_U8_Mod
mlib_VectorMul_U8_Sat
mlib_VectorMul_U8_U8_Mod
mlib_VectorMul_U8_U8_Sat
mlib_VectorNorm_S16_Sat
mlib_VectorNorm_S32_Sat
mlib_VectorNorm_S8_Sat
mlib_VectorNorm_U8_Sat
mlib_VectorReverseByteOrder
mlib_VectorReverseByteOrder_D64
mlib_VectorReverseByteOrder_D64_D64
mlib_VectorReverseByteOrder_F32
mlib_VectorReverseByteOrder_F32_F32
mlib_VectorReverseByteOrder_Inp
mlib_VectorReverseByteOrder_S16
mlib_VectorReverseByteOrder_S16_S16
mlib_VectorReverseByteOrder_S32
mlib_VectorReverseByteOrder_S32_S32
mlib_VectorReverseByteOrder_S64
mlib_VectorReverseByteOrder_S64_S64
mlib_VectorReverseByteOrder_U16
mlib_VectorReverseByteOrder_U16_U16
mlib_VectorReverseByteOrder_U32
mlib_VectorReverseByteOrder_U32_U32
mlib_VectorReverseByteOrder_U64
mlib_VectorReverseByteOrder_U64_U64
mlib_VectorScale_S16C_Mod
mlib_VectorScale_S16C_S16C_Mod
mlib_VectorScale_S16C_S16C_Sat
mlib_VectorScale_S16C_S8C_Mod
mlib_VectorScale_S16C_S8C_Sat
mlib_VectorScale_S16C_Sat
mlib_VectorScale_S16C_U8C_Mod
mlib_VectorScale_S16C_U8C_Sat
mlib_VectorScale_S16_Mod
mlib_VectorScale_S16_S16_Mod
mlib_VectorScale_S16_S16_Sat
mlib_VectorScale_S16_S8_Mod
mlib_VectorScale_S16_S8_Sat
mlib_VectorScale_S16_Sat
mlib_VectorScale_S16_U8_Mod
mlib_VectorScale_S16_U8_Sat

mlib_VectorScale_S32C_Mod
mlib_VectorScale_S32C_S16C_Mod
mlib_VectorScale_S32C_S16C_Sat
mlib_VectorScale_S32C_S32C_Mod
mlib_VectorScale_S32C_S32C_Sat
mlib_VectorScale_S32C_Sat
mlib_VectorScale_S32_Mod
mlib_VectorScale_S32_S16_Mod
mlib_VectorScale_S32_S16_Sat
mlib_VectorScale_S32_S32_Mod
mlib_VectorScale_S32_S32_Sat
mlib_VectorScale_S32_Sat
mlib_VectorScale_S8C_Mod
mlib_VectorScale_S8C_S8C_Mod
mlib_VectorScale_S8C_S8C_Sat
mlib_VectorScale_S8C_Sat
mlib_VectorScale_S8_Mod
mlib_VectorScale_S8_S8_Mod
mlib_VectorScale_S8_S8_Sat
mlib_VectorScale_S8_Sat
mlib_VectorScale_U8C_Mod
mlib_VectorScale_U8C_Sat
mlib_VectorScale_U8C_U8C_Mod
mlib_VectorScale_U8C_U8C_Sat
mlib_VectorScale_U8_Mod
mlib_VectorScale_U8_Sat
mlib_VectorScale_U8_U8_Mod
mlib_VectorScale_U8_U8_Sat
mlib_VectorSet_S16
mlib_VectorSet_S16C
mlib_VectorSet_S32
mlib_VectorSet_S32C
mlib_VectorSet_S8
mlib_VectorSet_S8C
mlib_VectorSet_U8
mlib_VectorSet_U8C
mlib_VectorSplit_S16_S16C
mlib_VectorSplit_S32_S32C
mlib_VectorSplit_S8_S8C
mlib_VectorSplit_U8_U8C
mlib_VectorSub_S16C_Mod
mlib_VectorSub_S16C_S16C_Mod
mlib_VectorSub_S16C_S16C_Sat
mlib_VectorSub_S16C_S8C_Mod
mlib_VectorSub_S16C_S8C_Sat
mlib_VectorSub_S16C_Sat
mlib_VectorSub_S16C_U8C_Mod
mlib_VectorSub_S16C_U8C_Sat

mlib_VectorSub_S16_Mod
mlib_VectorSub_S16_S16_Mod
mlib_VectorSub_S16_S16_Sat
mlib_VectorSub_S16_S8_Mod
mlib_VectorSub_S16_S8_Sat
mlib_VectorSub_S16_Sat
mlib_VectorSub_S16_U8_Mod
mlib_VectorSub_S16_U8_Sat
mlib_VectorSub_S32C_Mod
mlib_VectorSub_S32C_S16C_Mod
mlib_VectorSub_S32C_S16C_Sat
mlib_VectorSub_S32C_S32C_Mod
mlib_VectorSub_S32C_S32C_Sat
mlib_VectorSub_S32C_Sat
mlib_VectorSub_S32_Mod
mlib_VectorSub_S32_S16_Mod
mlib_VectorSub_S32_S16_Sat
mlib_VectorSub_S32_S32_Mod
mlib_VectorSub_S32_S32_Sat
mlib_VectorSub_S32_Sat
mlib_VectorSub_S8C_Mod
mlib_VectorSub_S8C_S8C_Mod
mlib_VectorSub_S8C_S8C_Sat
mlib_VectorSub_S8C_Sat
mlib_VectorSub_S8_Mod
mlib_VectorSub_S8_S8_Mod
mlib_VectorSub_S8_S8_Sat
mlib_VectorSub_S8_Sat
mlib_VectorSubS_S16C_Mod
mlib_VectorSubS_S16C_S16C_Mod
mlib_VectorSubS_S16C_S16C_Sat
mlib_VectorSubS_S16C_S8C_Mod
mlib_VectorSubS_S16C_S8C_Sat
mlib_VectorSubS_S16C_Sat
mlib_VectorSubS_S16C_U8C_Mod
mlib_VectorSubS_S16C_U8C_Sat
mlib_VectorSubS_S16_Mod
mlib_VectorSubS_S16_S16_Mod
mlib_VectorSubS_S16_S16_Sat
mlib_VectorSubS_S16_S8_Mod
mlib_VectorSubS_S16_S8_Sat
mlib_VectorSubS_S16_Sat
mlib_VectorSubS_S16_U8_Mod
mlib_VectorSubS_S16_U8_Sat
mlib_VectorSubS_S32C_Mod
mlib_VectorSubS_S32C_S16C_Mod
mlib_VectorSubS_S32C_S16C_Sat
mlib_VectorSubS_S32C_S32C_Mod

mlib_VectorSubS_S32C_S32C_Sat
mlib_VectorSubS_S32C_Sat
mlib_VectorSubS_S32_Mod
mlib_VectorSubS_S32_S16_Mod
mlib_VectorSubS_S32_S16_Sat
mlib_VectorSubS_S32_S32_Mod
mlib_VectorSubS_S32_S32_Sat
mlib_VectorSubS_S32_Sat
mlib_VectorSubS_S8C_Mod
mlib_VectorSubS_S8C_S8C_Mod
mlib_VectorSubS_S8C_S8C_Sat
mlib_VectorSubS_S8C_Sat
mlib_VectorSubS_S8_Mod
mlib_VectorSubS_S8_S8_Mod
mlib_VectorSubS_S8_S8_Sat
mlib_VectorSubS_S8_Sat
mlib_VectorSubS_U8C_Mod
mlib_VectorSubS_U8C_Sat
mlib_VectorSubS_U8C_U8C_Mod
mlib_VectorSubS_U8C_U8C_Sat
mlib_VectorSubS_U8_Mod
mlib_VectorSubS_U8_Sat
mlib_VectorSubS_U8_U8_Mod
mlib_VectorSubS_U8_U8_Sat
mlib_VectorSub_U8C_Mod
mlib_VectorSub_U8C_Sat
mlib_VectorSub_U8C_U8C_Mod
mlib_VectorSub_U8C_U8C_Sat
mlib_VectorSub_U8_Mod
mlib_VectorSub_U8_Sat
mlib_VectorSub_U8_U8_Mod
mlib_VectorSub_U8_U8_Sat
mlib_VectorSumAbsDiff_S16_Sat
mlib_VectorSumAbsDiff_S32_Sat
mlib_VectorSumAbsDiff_S8_Sat
mlib_VectorSumAbsDiff_U8_Sat
mlib_VectorSumAbs_S16_Sat
mlib_VectorSumAbs_S32_Sat
mlib_VectorSumAbs_S8_Sat
mlib_VectorSumAbs_U8_Sat
mlib_VectorZero_S16
mlib_VectorZero_S16C
mlib_VectorZero_S32
mlib_VectorZero_S32C
mlib_VectorZero_S8
mlib_VectorZero_S8C
mlib_VectorZero_U8
mlib_VectorZero_U8C

Graphics Functions

mlib_GraphicsBoundaryFill_32
mlib_GraphicsBoundaryFill_8
mlib_GraphicsDrawArc_32
mlib_GraphicsDrawArc_8
mlib_GraphicsDrawArc_A_32
mlib_GraphicsDrawArc_A_8
mlib_GraphicsDrawArc_X_32
mlib_GraphicsDrawArc_X_8
mlib_GraphicsDrawCircle_32
mlib_GraphicsDrawCircle_8
mlib_GraphicsDrawCircle_A_32
mlib_GraphicsDrawCircle_A_8
mlib_GraphicsDrawCircle_X_32
mlib_GraphicsDrawCircle_X_8
mlib_GraphicsDrawEllipse_32
mlib_GraphicsDrawEllipse_8
mlib_GraphicsDrawEllipse_A_32
mlib_GraphicsDrawEllipse_A_8
mlib_GraphicsDrawEllipse_X_32
mlib_GraphicsDrawEllipse_X_8
mlib_GraphicsDrawLine_32
mlib_GraphicsDrawLine_8
mlib_GraphicsDrawLine_A_32
mlib_GraphicsDrawLine_A_8
mlib_GraphicsDrawLine_AG_32
mlib_GraphicsDrawLine_AG_8
mlib_GraphicsDrawLine_AGZ_32
mlib_GraphicsDrawLine_AGZ_8
mlib_GraphicsDrawLine_AZ_32
mlib_GraphicsDrawLine_AZ_8
mlib_GraphicsDrawLineFanSet_32
mlib_GraphicsDrawLineFanSet_8
mlib_GraphicsDrawLineFanSet_A_32
mlib_GraphicsDrawLineFanSet_A_8
mlib_GraphicsDrawLineFanSet_AG_32
mlib_GraphicsDrawLineFanSet_AG_8
mlib_GraphicsDrawLineFanSet_AGZ_32
mlib_GraphicsDrawLineFanSet_AGZ_8
mlib_GraphicsDrawLineFanSet_AZ_32
mlib_GraphicsDrawLineFanSet_AZ_8
mlib_GraphicsDrawLineFanSet_G_32
mlib_GraphicsDrawLineFanSet_G_8
mlib_GraphicsDrawLineFanSet_GZ_32
mlib_GraphicsDrawLineFanSet_GZ_8
mlib_GraphicsDrawLineFanSet_X_32
mlib_GraphicsDrawLineFanSet_X_8
mlib_GraphicsDrawLineFanSet_Z_32
mlib_GraphicsDrawLineFanSet_Z_8

mlib_GraphicsDrawLine_G_32
mlib_GraphicsDrawLine_G_8
mlib_GraphicsDrawLine_GZ_32
mlib_GraphicsDrawLine_GZ_8
mlib_GraphicsDrawLineSet_32
mlib_GraphicsDrawLineSet_8
mlib_GraphicsDrawLineSet_A_32
mlib_GraphicsDrawLineSet_A_8
mlib_GraphicsDrawLineSet_AG_32
mlib_GraphicsDrawLineSet_AG_8
mlib_GraphicsDrawLineSet_AGZ_32
mlib_GraphicsDrawLineSet_AGZ_8
mlib_GraphicsDrawLineSet_AZ_32
mlib_GraphicsDrawLineSet_AZ_8
mlib_GraphicsDrawLineSet_G_32
mlib_GraphicsDrawLineSet_G_8
mlib_GraphicsDrawLineSet_GZ_32
mlib_GraphicsDrawLineSet_GZ_8
mlib_GraphicsDrawLineSet_X_32
mlib_GraphicsDrawLineSet_X_8
mlib_GraphicsDrawLineSet_Z_32
mlib_GraphicsDrawLineSet_Z_8
mlib_GraphicsDrawLineStripSet_32
mlib_GraphicsDrawLineStripSet_8
mlib_GraphicsDrawLineStripSet_A_32
mlib_GraphicsDrawLineStripSet_A_8
mlib_GraphicsDrawLineStripSet_AG_32
mlib_GraphicsDrawLineStripSet_AG_8
mlib_GraphicsDrawLineStripSet_AGZ_32
mlib_GraphicsDrawLineStripSet_AGZ_8
mlib_GraphicsDrawLineStripSet_AZ_32
mlib_GraphicsDrawLineStripSet_AZ_8
mlib_GraphicsDrawLineStripSet_G_32
mlib_GraphicsDrawLineStripSet_G_8
mlib_GraphicsDrawLineStripSet_GZ_32
mlib_GraphicsDrawLineStripSet_GZ_8
mlib_GraphicsDrawLineStripSet_X_32
mlib_GraphicsDrawLineStripSet_X_8
mlib_GraphicsDrawLineStripSet_Z_32
mlib_GraphicsDrawLineStripSet_Z_8
mlib_GraphicsDrawLine_X_32
mlib_GraphicsDrawLine_X_8
mlib_GraphicsDrawLine_Z_32
mlib_GraphicsDrawLine_Z_8
mlib_GraphicsDrawPoint_32
mlib_GraphicsDrawPoint_8
mlib_GraphicsDrawPointSet_32
mlib_GraphicsDrawPointSet_8

mlib_GraphicsDrawPointSet_X_32
mlib_GraphicsDrawPointSet_X_8
mlib_GraphicsDrawPoint_X_32
mlib_GraphicsDrawPoint_X_8
mlib_GraphicsDrawPolygon_32
mlib_GraphicsDrawPolygon_8
mlib_GraphicsDrawPolygon_A_32
mlib_GraphicsDrawPolygon_A_8
mlib_GraphicsDrawPolygon_AG_32
mlib_GraphicsDrawPolygon_AG_8
mlib_GraphicsDrawPolygon_AGZ_32
mlib_GraphicsDrawPolygon_AGZ_8
mlib_GraphicsDrawPolygon_AZ_32
mlib_GraphicsDrawPolygon_AZ_8
mlib_GraphicsDrawPolygon_G_32
mlib_GraphicsDrawPolygon_G_8
mlib_GraphicsDrawPolygon_GZ_32
mlib_GraphicsDrawPolygon_GZ_8
mlib_GraphicsDrawPolygon_X_32
mlib_GraphicsDrawPolygon_X_8
mlib_GraphicsDrawPolygon_Z_32
mlib_GraphicsDrawPolygon_Z_8
mlib_GraphicsDrawPolyline_32
mlib_GraphicsDrawPolyline_8
mlib_GraphicsDrawPolyline_A_32
mlib_GraphicsDrawPolyline_A_8
mlib_GraphicsDrawPolyline_AG_32
mlib_GraphicsDrawPolyline_AG_8
mlib_GraphicsDrawPolyline_AGZ_32
mlib_GraphicsDrawPolyline_AGZ_8
mlib_GraphicsDrawPolyline_AZ_32
mlib_GraphicsDrawPolyline_AZ_8
mlib_GraphicsDrawPolyline_G_32
mlib_GraphicsDrawPolyline_G_8
mlib_GraphicsDrawPolyline_GZ_32
mlib_GraphicsDrawPolyline_GZ_8
mlib_GraphicsDrawPolyline_X_32
mlib_GraphicsDrawPolyline_X_8
mlib_GraphicsDrawPolyline_Z_32
mlib_GraphicsDrawPolyline_Z_8
mlib_GraphicsDrawPolypoint_32
mlib_GraphicsDrawPolypoint_8
mlib_GraphicsDrawPolypoint_X_32
mlib_GraphicsDrawPolypoint_X_8
mlib_GraphicsDrawRectangle_32
mlib_GraphicsDrawRectangle_8
mlib_GraphicsDrawRectangle_X_32
mlib_GraphicsDrawRectangle_X_8

mlib_GraphicsDrawTriangle_32
mlib_GraphicsDrawTriangle_8
mlib_GraphicsDrawTriangle_A_32
mlib_GraphicsDrawTriangle_A_8
mlib_GraphicsDrawTriangle_AG_32
mlib_GraphicsDrawTriangle_AG_8
mlib_GraphicsDrawTriangle_AGZ_32
mlib_GraphicsDrawTriangle_AGZ_8
mlib_GraphicsDrawTriangle_AZ_32
mlib_GraphicsDrawTriangle_AZ_8
mlib_GraphicsDrawTriangleFanSet_32
mlib_GraphicsDrawTriangleFanSet_8
mlib_GraphicsDrawTriangleFanSet_A_32
mlib_GraphicsDrawTriangleFanSet_A_8
mlib_GraphicsDrawTriangleFanSet_AG_32
mlib_GraphicsDrawTriangleFanSet_AG_8
mlib_GraphicsDrawTriangleFanSet_AGZ_32
mlib_GraphicsDrawTriangleFanSet_AGZ_8
mlib_GraphicsDrawTriangleFanSet_AZ_32
mlib_GraphicsDrawTriangleFanSet_AZ_8
mlib_GraphicsDrawTriangleFanSet_G_32
mlib_GraphicsDrawTriangleFanSet_G_8
mlib_GraphicsDrawTriangleFanSet_GZ_32
mlib_GraphicsDrawTriangleFanSet_GZ_8
mlib_GraphicsDrawTriangleFanSet_X_32
mlib_GraphicsDrawTriangleFanSet_X_8
mlib_GraphicsDrawTriangleFanSet_Z_32
mlib_GraphicsDrawTriangleFanSet_Z_8
mlib_GraphicsDrawTriangle_G_32
mlib_GraphicsDrawTriangle_G_8
mlib_GraphicsDrawTriangle_GZ_32
mlib_GraphicsDrawTriangle_GZ_8
mlib_GraphicsDrawTriangleSet_32
mlib_GraphicsDrawTriangleSet_8
mlib_GraphicsDrawTriangleSet_A_32
mlib_GraphicsDrawTriangleSet_A_8
mlib_GraphicsDrawTriangleSet_AG_32
mlib_GraphicsDrawTriangleSet_AG_8
mlib_GraphicsDrawTriangleSet_AGZ_32
mlib_GraphicsDrawTriangleSet_AGZ_8
mlib_GraphicsDrawTriangleSet_AZ_32
mlib_GraphicsDrawTriangleSet_AZ_8
mlib_GraphicsDrawTriangleSet_G_32
mlib_GraphicsDrawTriangleSet_G_8
mlib_GraphicsDrawTriangleSet_GZ_32
mlib_GraphicsDrawTriangleSet_GZ_8
mlib_GraphicsDrawTriangleSet_X_32
mlib_GraphicsDrawTriangleSet_X_8

mlib_GraphicsDrawTriangleSet_Z_32
mlib_GraphicsDrawTriangleSet_Z_8
mlib_GraphicsDrawTriangleStripSet_32
mlib_GraphicsDrawTriangleStripSet_8
mlib_GraphicsDrawTriangleStripSet_A_32
mlib_GraphicsDrawTriangleStripSet_A_8
mlib_GraphicsDrawTriangleStripSet_AG_32
mlib_GraphicsDrawTriangleStripSet_AG_8
mlib_GraphicsDrawTriangleStripSet_AGZ_32
mlib_GraphicsDrawTriangleStripSet_AGZ_8
mlib_GraphicsDrawTriangleStripSet_AZ_32
mlib_GraphicsDrawTriangleStripSet_AZ_8
mlib_GraphicsDrawTriangleStripSet_G_32
mlib_GraphicsDrawTriangleStripSet_G_8
mlib_GraphicsDrawTriangleStripSet_GZ_32
mlib_GraphicsDrawTriangleStripSet_GZ_8
mlib_GraphicsDrawTriangleStripSet_X_32
mlib_GraphicsDrawTriangleStripSet_X_8
mlib_GraphicsDrawTriangleStripSet_Z_32
mlib_GraphicsDrawTriangleStripSet_Z_8
mlib_GraphicsDrawTriangle_X_32
mlib_GraphicsDrawTriangle_X_8
mlib_GraphicsDrawTriangle_Z_32
mlib_GraphicsDrawTriangle_Z_8
mlib_GraphicsFillArc_32
mlib_GraphicsFillArc_8
mlib_GraphicsFillArc_A_32
mlib_GraphicsFillArc_A_8
mlib_GraphicsFillArc_X_32
mlib_GraphicsFillArc_X_8
mlib_GraphicsFillCircle_32
mlib_GraphicsFillCircle_8
mlib_GraphicsFillCircle_A_32
mlib_GraphicsFillCircle_A_8
mlib_GraphicsFillCircle_X_32
mlib_GraphicsFillCircle_X_8
mlib_GraphicsFillEllipse_32
mlib_GraphicsFillEllipse_8
mlib_GraphicsFillEllipse_A_32
mlib_GraphicsFillEllipse_A_8
mlib_GraphicsFillEllipse_X_32
mlib_GraphicsFillEllipse_X_8
mlib_GraphicsFillPolygon_32
mlib_GraphicsFillPolygon_8
mlib_GraphicsFillPolygon_A_32
mlib_GraphicsFillPolygon_A_8
mlib_GraphicsFillPolygon_AG_32
mlib_GraphicsFillPolygon_AG_8

mlib_GraphicsFillPolygon_AGZ_32
mlib_GraphicsFillPolygon_AGZ_8
mlib_GraphicsFillPolygon_AZ_32
mlib_GraphicsFillPolygon_AZ_8
mlib_GraphicsFillPolygon_G_32
mlib_GraphicsFillPolygon_G_8
mlib_GraphicsFillPolygon_GZ_32
mlib_GraphicsFillPolygon_GZ_8
mlib_GraphicsFillPolygon_X_32
mlib_GraphicsFillPolygon_X_8
mlib_GraphicsFillPolygon_Z_32
mlib_GraphicsFillPolygon_Z_8
mlib_GraphicsFillRectangle_32
mlib_GraphicsFillRectangle_8
mlib_GraphicsFillRectangle_X_32
mlib_GraphicsFillRectangle_X_8
mlib_GraphicsFillTriangle_32
mlib_GraphicsFillTriangle_8
mlib_GraphicsFillTriangle_A_32
mlib_GraphicsFillTriangle_A_8
mlib_GraphicsFillTriangle_AG_32
mlib_GraphicsFillTriangle_AG_8
mlib_GraphicsFillTriangle_AGZ_32
mlib_GraphicsFillTriangle_AGZ_8
mlib_GraphicsFillTriangle_AZ_32
mlib_GraphicsFillTriangle_AZ_8
mlib_GraphicsFillTriangleFanSet_32
mlib_GraphicsFillTriangleFanSet_8
mlib_GraphicsFillTriangleFanSet_A_32
mlib_GraphicsFillTriangleFanSet_A_8
mlib_GraphicsFillTriangleFanSet_AG_32
mlib_GraphicsFillTriangleFanSet_AG_8
mlib_GraphicsFillTriangleFanSet_AGZ_32
mlib_GraphicsFillTriangleFanSet_AGZ_8
mlib_GraphicsFillTriangleFanSet_AZ_32
mlib_GraphicsFillTriangleFanSet_AZ_8
mlib_GraphicsFillTriangleFanSet_G_32
mlib_GraphicsFillTriangleFanSet_G_8
mlib_GraphicsFillTriangleFanSet_GZ_32
mlib_GraphicsFillTriangleFanSet_GZ_8
mlib_GraphicsFillTriangleFanSet_X_32
mlib_GraphicsFillTriangleFanSet_X_8
mlib_GraphicsFillTriangleFanSet_Z_32
mlib_GraphicsFillTriangleFanSet_Z_8
mlib_GraphicsFillTriangle_G_32
mlib_GraphicsFillTriangle_G_8
mlib_GraphicsFillTriangle_GZ_32
mlib_GraphicsFillTriangle_GZ_8

mlib_GraphicsFillTriangleSet_32
mlib_GraphicsFillTriangleSet_8
mlib_GraphicsFillTriangleSet_A_32
mlib_GraphicsFillTriangleSet_A_8
mlib_GraphicsFillTriangleSet_AG_32
mlib_GraphicsFillTriangleSet_AG_8
mlib_GraphicsFillTriangleSet_AGZ_32
mlib_GraphicsFillTriangleSet_AGZ_8
mlib_GraphicsFillTriangleSet_AZ_32
mlib_GraphicsFillTriangleSet_AZ_8
mlib_GraphicsFillTriangleSet_G_32
mlib_GraphicsFillTriangleSet_G_8
mlib_GraphicsFillTriangleSet_GZ_32
mlib_GraphicsFillTriangleSet_GZ_8
mlib_GraphicsFillTriangleSet_X_32
mlib_GraphicsFillTriangleSet_X_8
mlib_GraphicsFillTriangleSet_Z_32
mlib_GraphicsFillTriangleSet_Z_8
mlib_GraphicsFillTriangleStripSet_32
mlib_GraphicsFillTriangleStripSet_8
mlib_GraphicsFillTriangleStripSet_A_32
mlib_GraphicsFillTriangleStripSet_A_8
mlib_GraphicsFillTriangleStripSet_AG_32
mlib_GraphicsFillTriangleStripSet_AG_8
mlib_GraphicsFillTriangleStripSet_AGZ_32
mlib_GraphicsFillTriangleStripSet_AGZ_8
mlib_GraphicsFillTriangleStripSet_AZ_32
mlib_GraphicsFillTriangleStripSet_AZ_8
mlib_GraphicsFillTriangleStripSet_G_32
mlib_GraphicsFillTriangleStripSet_G_8
mlib_GraphicsFillTriangleStripSet_GZ_32
mlib_GraphicsFillTriangleStripSet_GZ_8
mlib_GraphicsFillTriangleStripSet_X_32
mlib_GraphicsFillTriangleStripSet_X_8
mlib_GraphicsFillTriangleStripSet_Z_32
mlib_GraphicsFillTriangleStripSet_Z_8
mlib_GraphicsFillTriangle_X_32
mlib_GraphicsFillTriangle_X_8
mlib_GraphicsFillTriangle_Z_32
mlib_GraphicsFillTriangle_Z_8
mlib_GraphicsFloodFill_32
mlib_GraphicsFloodFill_8

Imaging Functions

mlib_ImageAbs
mlib_ImageAbs_Fp
mlib_ImageAbs_Fp_Inp
mlib_ImageAbs_Inp
mlib_ImageAdd

mlib_ImageAdd_Fp
mlib_ImageAdd_Fp_Inp
mlib_ImageAdd_Inp
mlib_ImageAffine
mlib_ImageAffine_Fp
mlib_ImageAffineIndex
mlib_ImageAffineTable
mlib_ImageAffineTable_Fp
mlib_ImageAffineTransform
mlib_ImageAffineTransform_Fp
mlib_ImageAffineTransformIndex
mlib_ImageAnd
mlib_ImageAnd_Inp
mlib_ImageAndNot
mlib_ImageAndNot1_Inp
mlib_ImageAndNot2_Inp
mlib_ImageAutoCorrel
mlib_ImageAutoCorrel_Fp
mlib_ImageAve
mlib_ImageAve_Fp
mlib_ImageAve_Fp_Inp
mlib_ImageAve_Inp
mlib_ImageBlend
mlib_ImageBlend1_Fp_Inp
mlib_ImageBlend1_Inp
mlib_ImageBlend2_Fp_Inp
mlib_ImageBlend2_Inp
mlib_ImageBlendColor
mlib_ImageBlendColor_Fp
mlib_ImageBlendColor_Fp_Inp
mlib_ImageBlendColor_Inp
mlib_ImageBlend_DA_DA
mlib_ImageBlend_DA_DA_Inp
mlib_ImageBlend_DA_DC
mlib_ImageBlend_DA_DC_Inp
mlib_ImageBlend_DA_OMDA
mlib_ImageBlend_DA_OMDA_Inp
mlib_ImageBlend_DA_OMDC
mlib_ImageBlend_DA_OMDC_Inp
mlib_ImageBlend_DA_OMSA
mlib_ImageBlend_DA_OMSA_Inp
mlib_ImageBlend_DA_ONE
mlib_ImageBlend_DA_ONE_Inp
mlib_ImageBlend_DA_SA
mlib_ImageBlend_DA_SA_Inp
mlib_ImageBlend_DA_SAS
mlib_ImageBlend_DA_SAS_Inp
mlib_ImageBlend_DA_ZERO

mlib_ImageBlend_DA_ZERO_Inp
mlib_ImageBlend_Fp
mlib_ImageBlendMulti
mlib_ImageBlendMulti_Fp
mlib_ImageBlend_OMDA_DA
mlib_ImageBlend_OMDA_DA_Inp
mlib_ImageBlend_OMDA_DC
mlib_ImageBlend_OMDA_DC_Inp
mlib_ImageBlend_OMDA_OMDA
mlib_ImageBlend_OMDA_OMDA_Inp
mlib_ImageBlend_OMDA_OMDC
mlib_ImageBlend_OMDA_OMDC_Inp
mlib_ImageBlend_OMDA_OMSA
mlib_ImageBlend_OMDA_OMSA_Inp
mlib_ImageBlend_OMDA_ONE
mlib_ImageBlend_OMDA_ONE_Inp
mlib_ImageBlend_OMDA_SA
mlib_ImageBlend_OMDA_SA_Inp
mlib_ImageBlend_OMDA_SAS
mlib_ImageBlend_OMDA_SAS_Inp
mlib_ImageBlend_OMDA_ZERO
mlib_ImageBlend_OMDA_ZERO_Inp
mlib_ImageBlend_OMSA_DA
mlib_ImageBlend_OMSA_DA_Inp
mlib_ImageBlend_OMSA_DC
mlib_ImageBlend_OMSA_DC_Inp
mlib_ImageBlend_OMSA_OMDA
mlib_ImageBlend_OMSA_OMDA_Inp
mlib_ImageBlend_OMSA_OMDC
mlib_ImageBlend_OMSA_OMDC_Inp
mlib_ImageBlend_OMSA_OMSA
mlib_ImageBlend_OMSA_OMSA_Inp
mlib_ImageBlend_OMSA_ONE
mlib_ImageBlend_OMSA_ONE_Inp
mlib_ImageBlend_OMSA_SA
mlib_ImageBlend_OMSA_SA_Inp
mlib_ImageBlend_OMSA_SAS
mlib_ImageBlend_OMSA_SAS_Inp
mlib_ImageBlend_OMSA_ZERO
mlib_ImageBlend_OMSA_ZERO_Inp
mlib_ImageBlend_OMSC_DA
mlib_ImageBlend_OMSC_DA_Inp
mlib_ImageBlend_OMSC_DC
mlib_ImageBlend_OMSC_DC_Inp
mlib_ImageBlend_OMSC_OMDA
mlib_ImageBlend_OMSC_OMDA_Inp
mlib_ImageBlend_OMSC_OMDC
mlib_ImageBlend_OMSC_OMDC_Inp

mlib_ImageBlend_OMSC_OMSA
mlib_ImageBlend_OMSC_OMSA_Inp
mlib_ImageBlend_OMSC_ONE
mlib_ImageBlend_OMSC_ONE_Inp
mlib_ImageBlend_OMSC_SA
mlib_ImageBlend_OMSC_SA_Inp
mlib_ImageBlend_OMSC_SAS
mlib_ImageBlend_OMSC_SAS_Inp
mlib_ImageBlend_OMSC_ZERO
mlib_ImageBlend_OMSC_ZERO_Inp
mlib_ImageBlend_ONE_DA
mlib_ImageBlend_ONE_DA_Inp
mlib_ImageBlend_ONE_DC
mlib_ImageBlend_ONE_DC_Inp
mlib_ImageBlend_ONE_OMDA
mlib_ImageBlend_ONE_OMDA_Inp
mlib_ImageBlend_ONE_OMDC
mlib_ImageBlend_ONE_OMDC_Inp
mlib_ImageBlend_ONE_OMSA
mlib_ImageBlend_ONE_OMSA_Inp
mlib_ImageBlend_ONE_ONE
mlib_ImageBlend_ONE_ONE_Inp
mlib_ImageBlend_ONE_SA
mlib_ImageBlend_ONE_SA_Inp
mlib_ImageBlend_ONE_SAS
mlib_ImageBlend_ONE_SAS_Inp
mlib_ImageBlend_ONE_ZERO
mlib_ImageBlend_ONE_ZERO_Inp
mlib_ImageBlendRGBA2ARGB
mlib_ImageBlendRGBA2BGRA
mlib_ImageBlend_SA_DA
mlib_ImageBlend_SA_DA_Inp
mlib_ImageBlend_SA_DC
mlib_ImageBlend_SA_DC_Inp
mlib_ImageBlend_SA_OMDA
mlib_ImageBlend_SA_OMDA_Inp
mlib_ImageBlend_SA_OMDC
mlib_ImageBlend_SA_OMDC_Inp
mlib_ImageBlend_SA_OMSA
mlib_ImageBlend_SA_OMSA_Inp
mlib_ImageBlend_SA_ONE
mlib_ImageBlend_SA_ONE_Inp
mlib_ImageBlend_SA_SA
mlib_ImageBlend_SA_SA_Inp
mlib_ImageBlend_SA_SAS
mlib_ImageBlend_SA_SAS_Inp
mlib_ImageBlend_SA_ZERO
mlib_ImageBlend_SA_ZERO_Inp

mlib_ImageBlend_SC_DA
mlib_ImageBlend_SC_DA_Inp
mlib_ImageBlend_SC_DC
mlib_ImageBlend_SC_DC_Inp
mlib_ImageBlend_SC_OMDA
mlib_ImageBlend_SC_OMDA_Inp
mlib_ImageBlend_SC_OMDC
mlib_ImageBlend_SC_OMDC_Inp
mlib_ImageBlend_SC_OMSA
mlib_ImageBlend_SC_OMSA_Inp
mlib_ImageBlend_SC_ONE
mlib_ImageBlend_SC_ONE_Inp
mlib_ImageBlend_SC_SA
mlib_ImageBlend_SC_SA_Inp
mlib_ImageBlend_SC_SAS
mlib_ImageBlend_SC_SAS_Inp
mlib_ImageBlend_SC_ZERO
mlib_ImageBlend_SC_ZERO_Inp
mlib_ImageBlend_ZERO_DA
mlib_ImageBlend_ZERO_DA_Inp
mlib_ImageBlend_ZERO_DC
mlib_ImageBlend_ZERO_DC_Inp
mlib_ImageBlend_ZERO_OMDA
mlib_ImageBlend_ZERO_OMDA_Inp
mlib_ImageBlend_ZERO_OMDC
mlib_ImageBlend_ZERO_OMDC_Inp
mlib_ImageBlend_ZERO_OMSA
mlib_ImageBlend_ZERO_OMSA_Inp
mlib_ImageBlend_ZERO_ONE
mlib_ImageBlend_ZERO_ONE_Inp
mlib_ImageBlend_ZERO_SA
mlib_ImageBlend_ZERO_SA_Inp
mlib_ImageBlend_ZERO_SAS
mlib_ImageBlend_ZERO_SAS_Inp
mlib_ImageBlend_ZERO_ZERO
mlib_ImageBlend_ZERO_ZERO_Inp
mlib_ImageChannelCopy
mlib_ImageChannelExtract
mlib_ImageChannelInsert
mlib_ImageChannelMerge
mlib_ImageChannelSplit
mlib_ImageClear
mlib_ImageClearEdge
mlib_ImageClearEdge_Fp
mlib_ImageClear_Fp
mlib_ImageColorConvert1
mlib_ImageColorConvert1_Fp
mlib_ImageColorConvert2

mlib_ImageColorConvert2_Fp
mlib_ImageColorDitherFree
mlib_ImageColorDitherInit
mlib_ImageColorErrorDiffusion3x3
mlib_ImageColorErrorDiffusionMxN
mlib_ImageColorHSL2RGB
mlib_ImageColorHSL2RGB_Fp
mlib_ImageColorHSV2RGB
mlib_ImageColorHSV2RGB_Fp
mlib_ImageColorOrderedDither8x8
mlib_ImageColorOrderedDitherMxN
mlib_ImageColorRGB2CIEMono
mlib_ImageColorRGB2CIEMono_Fp
mlib_ImageColorRGB2HSL
mlib_ImageColorRGB2HSL_Fp
mlib_ImageColorRGB2HSV
mlib_ImageColorRGB2HSV_Fp
mlib_ImageColorRGB2Mono
mlib_ImageColorRGB2Mono_Fp
mlib_ImageColorRGB2XYZ
mlib_ImageColorRGB2XYZ_Fp
mlib_ImageColorRGB2YCC
mlib_ImageColorRGB2YCC_Fp
mlib_ImageColorTrue2Index
mlib_ImageColorTrue2IndexFree
mlib_ImageColorTrue2IndexInit
mlib_ImageColorXYZ2RGB
mlib_ImageColorXYZ2RGB_Fp
mlib_ImageColorYCC2RGB
mlib_ImageColorYCC2RGB_Fp
mlib_ImageComposite
mlib_ImageComposite_Inp
mlib_ImageConstAdd
mlib_ImageConstAdd_Fp
mlib_ImageConstAdd_Fp_Inp
mlib_ImageConstAdd_Inp
mlib_ImageConstAnd
mlib_ImageConstAnd_Inp
mlib_ImageConstAndNot
mlib_ImageConstAndNot_Inp
mlib_ImageConstDiv
mlib_ImageConstDiv_Fp
mlib_ImageConstDiv_Fp_Inp
mlib_ImageConstDiv_Inp
mlib_ImageConstDivShift
mlib_ImageConstDivShift_Inp
mlib_ImageConstMul
mlib_ImageConstMul_Fp

mlib_ImageConstMul_Fp_Inp
mlib_ImageConstMul_Inp
mlib_ImageConstMulShift
mlib_ImageConstMulShift_Inp
mlib_ImageConstNotAnd
mlib_ImageConstNotAnd_Inp
mlib_ImageConstNotOr
mlib_ImageConstNotOr_Inp
mlib_ImageConstNotXor
mlib_ImageConstNotXor_Inp
mlib_ImageConstOr
mlib_ImageConstOr_Inp
mlib_ImageConstOrNot
mlib_ImageConstOrNot_Inp
mlib_ImageConstSub
mlib_ImageConstSub_Fp
mlib_ImageConstSub_Fp_Inp
mlib_ImageConstSub_Inp
mlib_ImageConstXor
mlib_ImageConstXor_Inp
mlib_ImageConv2x2
mlib_ImageConv2x2_Fp
mlib_ImageConv2x2Index
mlib_ImageConv3x3
mlib_ImageConv3x3_Fp
mlib_ImageConv3x3Index
mlib_ImageConv4x4
mlib_ImageConv4x4_Fp
mlib_ImageConv4x4Index
mlib_ImageConv5x5
mlib_ImageConv5x5_Fp
mlib_ImageConv5x5Index
mlib_ImageConv7x7
mlib_ImageConv7x7_Fp
mlib_ImageConv7x7Index
mlib_ImageConvKernelConvert
mlib_ImageConvMxN
mlib_ImageConvMxN_Fp
mlib_ImageConvMxNIndex
mlib_ImageConvolveMxN
mlib_ImageConvolveMxN_Fp
mlib_ImageCopy
mlib_ImageCopyArea
mlib_ImageCopyMask
mlib_ImageCopyMask_Fp
mlib_ImageCopySubimage
mlib_ImageCreate
mlib_ImageCreateStruct

mlib_ImageCreateSubimage
mlib_ImageCrossCorrel
mlib_ImageCrossCorrel_Fp
mlib_ImageDataTypeConvert
mlib_ImageDelete
mlib_ImageDilate4
mlib_ImageDilate4_Fp
mlib_ImageDilate8
mlib_ImageDilate8_Fp
mlib_ImageDiv1_Fp_Inp
mlib_ImageDiv2_Fp_Inp
mlib_ImageDivAlpha
mlib_ImageDivAlpha_Fp
mlib_ImageDivAlpha_Fp_Inp
mlib_ImageDivAlpha_Inp
mlib_ImageDivConstShift
mlib_ImageDivConstShift_Inp
mlib_ImageDiv_Fp
mlib_ImageDivShift
mlib_ImageDivShift1_Inp
mlib_ImageDivShift2_Inp
mlib_ImageErode4
mlib_ImageErode4_Fp
mlib_ImageErode8
mlib_ImageErode8_Fp
mlib_ImageExp
mlib_ImageExp_Fp
mlib_ImageExp_Fp_Inp
mlib_ImageExp_Inp
mlib_ImageExtrema2
mlib_ImageExtrema2_Fp
mlib_ImageExtremaLocations
mlib_ImageExtremaLocations_Fp
mlib_ImageFilteredSubsample
mlib_ImageFilteredSubsample_Fp
mlib_ImageFlipAntiDiag
mlib_ImageFlipAntiDiag_Fp
mlib_ImageFlipMainDiag
mlib_ImageFlipMainDiag_Fp
mlib_ImageFlipX
mlib_ImageFlipX_Fp
mlib_ImageFlipY
mlib_ImageFlipY_Fp
mlib_ImageFourierTransform
mlib_ImageGetBitOffset
mlib_ImageGetChannels
mlib_ImageGetData
mlib_ImageGetFlags

mlib_ImageGetFormat
mlib_ImageGetHeight
mlib_ImageGetPaddings
mlib_ImageGetStride
mlib_ImageGetType
mlib_ImageGetWidth
mlib_ImageGradient3x3
mlib_ImageGradient3x3_Fp
mlib_ImageGradientMxN
mlib_ImageGradientMxN_Fp
mlib_ImageGridWarp
mlib_ImageGridWarp_Fp
mlib_ImageGridWarpTable
mlib_ImageGridWarpTable_Fp
mlib_ImageHistogram
mlib_ImageHistogram2
mlib_ImageInterpTableCreate
mlib_ImageInterpTableDelete
mlib_ImageInvert
mlib_ImageInvert_Fp
mlib_ImageInvert_Fp_Inp
mlib_ImageInvert_Inp
mlib_ImageIsNotAligned2
mlib_ImageIsNotAligned4
mlib_ImageIsNotAligned64
mlib_ImageIsNotAligned8
mlib_ImageIsNotHeight2X
mlib_ImageIsNotHeight4X
mlib_ImageIsNotHeight8X
mlib_ImageIsNotOneDvector
mlib_ImageIsNotStride8X
mlib_ImageIsNotWidth2X
mlib_ImageIsNotWidth4X
mlib_ImageIsNotWidth8X
mlib_ImageIsUserAllocated
mlib_ImageLog
mlib_ImageLog_Fp
mlib_ImageLog_Fp_Inp
mlib_ImageLog_Inp
mlib_ImageLookUp
mlib_ImageLookUp2
mlib_ImageLookUp_Inp
mlib_ImageLookUpMask
mlib_ImageMax
mlib_ImageMaxFilter3x3
mlib_ImageMaxFilter3x3_Fp
mlib_ImageMaxFilter5x5
mlib_ImageMaxFilter5x5_Fp

mlib_ImageMaxFilter7x7
mlib_ImageMaxFilter7x7_Fp
mlib_ImageMax_Fp
mlib_ImageMax_Fp_Inp
mlib_ImageMaximum
mlib_ImageMaximum_Fp
mlib_ImageMax_Inp
mlib_ImageMean
mlib_ImageMean_Fp
mlib_ImageMedianFilter3x3
mlib_ImageMedianFilter3x3_Fp
mlib_ImageMedianFilter3x3_US
mlib_ImageMedianFilter5x5
mlib_ImageMedianFilter5x5_Fp
mlib_ImageMedianFilter5x5_US
mlib_ImageMedianFilter7x7
mlib_ImageMedianFilter7x7_Fp
mlib_ImageMedianFilter7x7_US
mlib_ImageMedianFilterMxN
mlib_ImageMedianFilterMxN_Fp
mlib_ImageMedianFilterMxN_US
mlib_ImageMin
mlib_ImageMinFilter3x3
mlib_ImageMinFilter3x3_Fp
mlib_ImageMinFilter5x5
mlib_ImageMinFilter5x5_Fp
mlib_ImageMinFilter7x7
mlib_ImageMinFilter7x7_Fp
mlib_ImageMin_Fp
mlib_ImageMin_Fp_Inp
mlib_ImageMinimum
mlib_ImageMinimum_Fp
mlib_ImageMin_Inp
mlib_ImageMoment2
mlib_ImageMoment2_Fp
mlib_ImageMulAlpha
mlib_ImageMulAlpha_Fp
mlib_ImageMulAlpha_Fp_Inp
mlib_ImageMulAlpha_Inp
mlib_ImageMul_Fp
mlib_ImageMul_Fp_Inp
mlib_ImageMulShift
mlib_ImageMulShift_Inp
mlib_ImageNot
mlib_ImageNotAnd
mlib_ImageNotAnd_Inp
mlib_ImageNot_Inp
mlib_ImageNotOr

mlib_ImageNotOr_Inp
mlib_ImageNotXor
mlib_ImageNotXor_Inp
mlib_ImageOr
mlib_ImageOr_Inp
mlib_ImageOrNot
mlib_ImageOrNot1_Inp
mlib_ImageOrNot2_Inp
mlib_ImagePolynomialWarp
mlib_ImagePolynomialWarp_Fp
mlib_ImagePolynomialWarpTable
mlib_ImagePolynomialWarpTable_Fp
mlib_ImageRankFilter3x3
mlib_ImageRankFilter3x3_Fp
mlib_ImageRankFilter3x3_US
mlib_ImageRankFilter5x5
mlib_ImageRankFilter5x5_Fp
mlib_ImageRankFilter5x5_US
mlib_ImageRankFilter7x7
mlib_ImageRankFilter7x7_Fp
mlib_ImageRankFilter7x7_US
mlib_ImageRankFilterMxN
mlib_ImageRankFilterMxN_Fp
mlib_ImageRankFilterMxN_US
mlib_ImageReformat
mlib_ImageReplaceColor
mlib_ImageReplaceColor_Fp
mlib_ImageReplaceColor_Fp_Inp
mlib_ImageReplaceColor_Inp
mlib_ImageRotate
mlib_ImageRotate180
mlib_ImageRotate180_Fp
mlib_ImageRotate270
mlib_ImageRotate270_Fp
mlib_ImageRotate90
mlib_ImageRotate90_Fp
mlib_ImageRotate_Fp
mlib_ImageRotateIndex
mlib_ImageScalarBlend
mlib_ImageScalarBlend_Fp
mlib_ImageScalarBlend_Fp_Inp
mlib_ImageScalarBlend_Inp
mlib_ImageScale
mlib_ImageScale2
mlib_ImageScale2_Inp
mlib_ImageScale_Fp
mlib_ImageScale_Fp_Inp
mlib_ImageScale_Inp

mlib_ImageSConv3x3
mlib_ImageSConv3x3_Fp
mlib_ImageSConv5x5
mlib_ImageSConv5x5_Fp
mlib_ImageSConv7x7
mlib_ImageSConv7x7_Fp
mlib_ImageSConvKernelConvert
mlib_ImageSetFormat
mlib_ImageSetPaddings
mlib_ImageSobel
mlib_ImageSobel_Fp
mlib_ImageSqr_Fp
mlib_ImageSqr_Fp_Inp
mlib_ImageSqrShift
mlib_ImageSqrShift_Inp
mlib_ImageStdDev
mlib_ImageStdDev_Fp
mlib_ImageSub
mlib_ImageSub1_Fp_Inp
mlib_ImageSub1_Inp
mlib_ImageSub2_Fp_Inp
mlib_ImageSub2_Inp
mlib_ImageSub_Fp
mlib_ImageSubsampleAverage
mlib_ImageSubsampleAverage_Fp
mlib_ImageSubsampleBinaryToGray
mlib_ImageTestFlags
mlib_ImageThresh1
mlib_ImageThresh1_Fp
mlib_ImageThresh1_Fp_Inp
mlib_ImageThresh1_Inp
mlib_ImageThresh2
mlib_ImageThresh2_Fp
mlib_ImageThresh2_Fp_Inp
mlib_ImageThresh2_Inp
mlib_ImageThresh3
mlib_ImageThresh3_Fp
mlib_ImageThresh3_Fp_Inp
mlib_ImageThresh3_Inp
mlib_ImageThresh4
mlib_ImageThresh4_Fp
mlib_ImageThresh4_Fp_Inp
mlib_ImageThresh4_Inp
mlib_ImageThresh5
mlib_ImageThresh5_Fp
mlib_ImageThresh5_Fp_Inp
mlib_ImageThresh5_Inp
mlib_ImageXor

mlib_ImageXor_Inp
mlib_ImageXProj
mlib_ImageXProj_Fp
mlib_ImageYProj
mlib_ImageYProj_Fp
mlib_ImageZoom
mlib_ImageZoomBlend
mlib_ImageZoom_Fp
mlib_ImageZoomIn2X
mlib_ImageZoomIn2X_Fp
mlib_ImageZoomIn2XIndex
mlib_ImageZoomIndex
mlib_ImageZoomOut2X
mlib_ImageZoomOut2X_Fp
mlib_ImageZoomOut2XIndex
mlib_ImageZoomTranslate
mlib_ImageZoomTranslateBlend
mlib_ImageZoomTranslate_Fp
mlib_ImageZoomTranslateTable
mlib_ImageZoomTranslateTableBlend
mlib_ImageZoomTranslateTable_Fp
mlib_ImageZoomTranslateToGray

Signal Processing Functions

mlib_SignalADPCM2Bits2Linear
mlib_SignalADPCM3Bits2Linear
mlib_SignalADPCM4Bits2Linear
mlib_SignalADPCM5Bits2Linear
mlib_SignalADPCMFree
mlib_SignalADPCMInit
mlib_SignalALaw2Linear
mlib_SignalALaw2uLaw
mlib_SignalAutoCorrel_F32
mlib_SignalAutoCorrel_F32S
mlib_SignalAutoCorrel_S16
mlib_SignalAutoCorrel_S16S
mlib_SignalCepstral_F32
mlib_SignalCepstralFree_F32
mlib_SignalCepstralFree_S16
mlib_SignalCepstralInit_F32
mlib_SignalCepstralInit_S16
mlib_SignalCepstral_S16
mlib_SignalCepstral_S16_Adp
mlib_SignalConvertShift_F32_S16
mlib_SignalConvertShift_F32_S32
mlib_SignalConvertShift_F32_S8
mlib_SignalConvertShift_F32S_S16S
mlib_SignalConvertShift_F32S_S32S
mlib_SignalConvertShift_F32S_S8S

mlib_SignalConvertShift_F32S_U8S
mlib_SignalConvertShift_F32_U8
mlib_SignalConvertShift_S16_F32_Sat
mlib_SignalConvertShift_S16_S32_Sat
mlib_SignalConvertShift_S16_S8_Sat
mlib_SignalConvertShift_S16S_F32S_Sat
mlib_SignalConvertShift_S16S_S32S_Sat
mlib_SignalConvertShift_S16S_S8S_Sat
mlib_SignalConvertShift_S16S_U8S_Sat
mlib_SignalConvertShift_S16_U8_Sat
mlib_SignalConvertShift_S32_F32_Sat
mlib_SignalConvertShift_S32_S16_Sat
mlib_SignalConvertShift_S32_S8_Sat
mlib_SignalConvertShift_S32S_F32S_Sat
mlib_SignalConvertShift_S32S_S16S_Sat
mlib_SignalConvertShift_S32S_S8S_Sat
mlib_SignalConvertShift_S32S_U8S_Sat
mlib_SignalConvertShift_S32_U8_Sat
mlib_SignalConvertShift_S8_F32_Sat
mlib_SignalConvertShift_S8_S16_Sat
mlib_SignalConvertShift_S8_S32_Sat
mlib_SignalConvertShift_S8S_F32S_Sat
mlib_SignalConvertShift_S8S_S16S_Sat
mlib_SignalConvertShift_S8S_S32S_Sat
mlib_SignalConvertShift_S8S_U8S_Sat
mlib_SignalConvertShift_S8_U8_Sat
mlib_SignalConvertShift_U8_F32_Sat
mlib_SignalConvertShift_U8_S16_Sat
mlib_SignalConvertShift_U8_S32_Sat
mlib_SignalConvertShift_U8_S8_Sat
mlib_SignalConvertShift_U8S_F32S_Sat
mlib_SignalConvertShift_U8S_S16S_Sat
mlib_SignalConvertShift_U8S_S32S_Sat
mlib_SignalConvertShift_U8S_S8S_Sat
mlib_SignalConv_F32_F32
mlib_SignalConv_F32S_F32S
mlib_SignalConv_S16_S16_Sat
mlib_SignalConv_S16S_S16S_Sat
mlib_SignalCrossCorrel_F32
mlib_SignalCrossCorrel_F32S
mlib_SignalCrossCorrel_S16
mlib_SignalCrossCorrel_S16S
mlib_SignalDownSample_F32_F32
mlib_SignalDownSample_F32S_F32S
mlib_SignalDownSample_S16_S16
mlib_SignalDownSample_S16S_S16S
mlib_SignalDTWKScalar_F32
mlib_SignalDTWKScalarFree_F32

mlib_SignalDTWKScalarFree_S16
mlib_SignalDTWKScalarInit_F32
mlib_SignalDTWKScalarInit_S16
mlib_SignalDTWKScalarPath_F32
mlib_SignalDTWKScalarPath_S16
mlib_SignalDTWKScalar_S16
mlib_SignalDTWKVector_F32
mlib_SignalDTWKVectorFree_F32
mlib_SignalDTWKVectorFree_S16
mlib_SignalDTWKVectorInit_F32
mlib_SignalDTWKVectorInit_S16
mlib_SignalDTWKVectorPath_F32
mlib_SignalDTWKVectorPath_S16
mlib_SignalDTWKVector_S16
mlib_SignalDTWScalar_F32
mlib_SignalDTWScalarFree_F32
mlib_SignalDTWScalarFree_S16
mlib_SignalDTWScalarInit_F32
mlib_SignalDTWScalarInit_S16
mlib_SignalDTWScalarPath_F32
mlib_SignalDTWScalarPath_S16
mlib_SignalDTWScalar_S16
mlib_SignalDTWVector_F32
mlib_SignalDTWVectorFree_F32
mlib_SignalDTWVectorFree_S16
mlib_SignalDTWVectorInit_F32
mlib_SignalDTWVectorInit_S16
mlib_SignalDTWVectorPath_F32
mlib_SignalDTWVectorPath_S16
mlib_SignalDTWVector_S16
mlib_SignalEmphasize_F32_F32
mlib_SignalEmphasize_F32S_F32S
mlib_SignalEmphasizeFree_F32_F32
mlib_SignalEmphasizeFree_F32S_F32S
mlib_SignalEmphasizeFree_S16_S16
mlib_SignalEmphasizeFree_S16S_S16S
mlib_SignalEmphasizeInit_F32_F32
mlib_SignalEmphasizeInit_F32S_F32S
mlib_SignalEmphasizeInit_S16_S16
mlib_SignalEmphasizeInit_S16S_S16S
mlib_SignalEmphasize_S16_S16_Sat
mlib_SignalEmphasize_S16S_S16S_Sat
mlib_SignalFFT_1_D64
mlib_SignalFFT_1_D64C
mlib_SignalFFT_1_D64C_D64
mlib_SignalFFT_1_D64C_D64C
mlib_SignalFFT_1_D64_D64
mlib_SignalFFT_1_F32

mlib_SignalFFT_1_F32C
mlib_SignalFFT_1_F32C_F32
mlib_SignalFFT_1_F32C_F32C
mlib_SignalFFT_1_F32_F32
mlib_SignalFFT_1_S16C_Mod
mlib_SignalFFT_1_S16C_S16C_Mod
mlib_SignalFFT_1_S16C_S16_Mod
mlib_SignalFFT_1_S16_Mod
mlib_SignalFFT_1_S16_S16_Mod
mlib_SignalFFT_2_D64
mlib_SignalFFT_2_D64C
mlib_SignalFFT_2_D64C_D64
mlib_SignalFFT_2_D64C_D64C
mlib_SignalFFT_2_D64_D64
mlib_SignalFFT_2_F32
mlib_SignalFFT_2_F32C
mlib_SignalFFT_2_F32C_F32
mlib_SignalFFT_2_F32C_F32C
mlib_SignalFFT_2_F32_F32
mlib_SignalFFT_2_S16
mlib_SignalFFT_2_S16C
mlib_SignalFFT_2_S16C_S16
mlib_SignalFFT_2_S16C_S16C
mlib_SignalFFT_2_S16_S16
mlib_SignalFFT_3_D64
mlib_SignalFFT_3_D64C
mlib_SignalFFT_3_D64C_D64
mlib_SignalFFT_3_D64C_D64C
mlib_SignalFFT_3_D64_D64
mlib_SignalFFT_3_F32
mlib_SignalFFT_3_F32C
mlib_SignalFFT_3_F32C_F32
mlib_SignalFFT_3_F32C_F32C
mlib_SignalFFT_3_F32_F32
mlib_SignalFFT_3_S16C_Mod
mlib_SignalFFT_3_S16C_S16C_Mod
mlib_SignalFFT_3_S16C_S16_Mod
mlib_SignalFFT_3_S16_Mod
mlib_SignalFFT_3_S16_S16_Mod
mlib_SignalFFT_4_S16
mlib_SignalFFT_4_S16C
mlib_SignalFFT_4_S16C_S16
mlib_SignalFFT_4_S16C_S16C
mlib_SignalFFT_4_S16_S16
mlib_SignalFFTW_1_F32
mlib_SignalFFTW_1_F32C
mlib_SignalFFTW_1_F32C_F32
mlib_SignalFFTW_1_F32C_F32C

mlib_SignalFFTW_1_F32_F32
mlib_SignalFFTW_1_S16C_Mod
mlib_SignalFFTW_1_S16C_S16C_Mod
mlib_SignalFFTW_1_S16C_S16_Mod
mlib_SignalFFTW_1_S16_Mod
mlib_SignalFFTW_1_S16_S16_Mod
mlib_SignalFFTW_2_F32
mlib_SignalFFTW_2_F32C
mlib_SignalFFTW_2_F32C_F32
mlib_SignalFFTW_2_F32C_F32C
mlib_SignalFFTW_2_F32_F32
mlib_SignalFFTW_2_S16
mlib_SignalFFTW_2_S16C
mlib_SignalFFTW_2_S16C_S16
mlib_SignalFFTW_2_S16C_S16C
mlib_SignalFFTW_2_S16_S16
mlib_SignalFFTW_3_F32
mlib_SignalFFTW_3_F32C
mlib_SignalFFTW_3_F32C_F32
mlib_SignalFFTW_3_F32C_F32C
mlib_SignalFFTW_3_F32_F32
mlib_SignalFFTW_3_S16C_Mod
mlib_SignalFFTW_3_S16C_S16C_Mod
mlib_SignalFFTW_3_S16C_S16_Mod
mlib_SignalFFTW_3_S16_Mod
mlib_SignalFFTW_3_S16_S16_Mod
mlib_SignalFFTW_4_S16
mlib_SignalFFTW_4_S16C
mlib_SignalFFTW_4_S16C_S16
mlib_SignalFFTW_4_S16C_S16C
mlib_SignalFFTW_4_S16_S16
mlib_SignalFIR_F32_F32
mlib_SignalFIR_F32S_F32S
mlib_SignalFIRFree_F32_F32
mlib_SignalFIRFree_F32S_F32S
mlib_SignalFIRFree_S16_S16
mlib_SignalFIRFree_S16S_S16S
mlib_SignalFIRInit_F32_F32
mlib_SignalFIRInit_F32S_F32S
mlib_SignalFIRInit_S16_S16
mlib_SignalFIRInit_S16S_S16S
mlib_SignalFIR_S16_S16_Sat
mlib_SignalFIR_S16S_S16S_Sat
mlib_SignalGaussNoise_F32
mlib_SignalGaussNoiseFree_F32
mlib_SignalGaussNoiseFree_S16
mlib_SignalGaussNoiseInit_F32
mlib_SignalGaussNoiseInit_S16

mlib_SignalGaussNoise_S16
mlib_SignalGenBartlett_F32
mlib_SignalGenBartlett_S16
mlib_SignalGenBlackman_F32
mlib_SignalGenBlackman_S16
mlib_SignalGenHamming_F32
mlib_SignalGenHamming_S16
mlib_SignalGenHanning_F32
mlib_SignalGenHanning_S16
mlib_SignalGenKaiser_F32
mlib_SignalGenKaiser_S16
mlib_SignalIFFT_1_D64
mlib_SignalIFFT_1_D64C
mlib_SignalIFFT_1_D64C_D64C
mlib_SignalIFFT_1_D64_D64
mlib_SignalIFFT_1_D64_D64C
mlib_SignalIFFT_1_F32
mlib_SignalIFFT_1_F32C
mlib_SignalIFFT_1_F32C_F32C
mlib_SignalIFFT_1_F32_F32
mlib_SignalIFFT_1_F32_F32C
mlib_SignalIFFT_1_S16
mlib_SignalIFFT_1_S16C
mlib_SignalIFFT_1_S16C_S16C
mlib_SignalIFFT_1_S16_S16
mlib_SignalIFFT_1_S16_S16C
mlib_SignalIFFT_2_D64
mlib_SignalIFFT_2_D64C
mlib_SignalIFFT_2_D64C_D64C
mlib_SignalIFFT_2_D64_D64
mlib_SignalIFFT_2_D64_D64C
mlib_SignalIFFT_2_F32
mlib_SignalIFFT_2_F32C
mlib_SignalIFFT_2_F32C_F32C
mlib_SignalIFFT_2_F32_F32
mlib_SignalIFFT_2_F32_F32C
mlib_SignalIFFT_2_S16C_Mod
mlib_SignalIFFT_2_S16C_S16C_Mod
mlib_SignalIFFT_2_S16_Mod
mlib_SignalIFFT_2_S16_S16C_Mod
mlib_SignalIFFT_2_S16_S16_Mod
mlib_SignalIFFT_3_D64
mlib_SignalIFFT_3_D64C
mlib_SignalIFFT_3_D64C_D64C
mlib_SignalIFFT_3_D64_D64
mlib_SignalIFFT_3_D64_D64C
mlib_SignalIFFT_3_F32
mlib_SignalIFFT_3_F32C

mlib_SignalIFFT_3_F32C_F32C
mlib_SignalIFFT_3_F32_F32
mlib_SignalIFFT_3_F32_F32C
mlib_SignalIFFT_3_S16C_Mod
mlib_SignalIFFT_3_S16C_S16C_Mod
mlib_SignalIFFT_3_S16_Mod
mlib_SignalIFFT_3_S16_S16C_Mod
mlib_SignalIFFT_3_S16_S16_Mod
mlib_SignalIFFT_4_S16
mlib_SignalIFFT_4_S16C
mlib_SignalIFFT_4_S16C_S16C
mlib_SignalIFFT_4_S16_S16
mlib_SignalIFFT_4_S16_S16C
mlib_SignalIFFTW_1_F32
mlib_SignalIFFTW_1_F32C
mlib_SignalIFFTW_1_F32C_F32C
mlib_SignalIFFTW_1_F32_F32
mlib_SignalIFFTW_1_F32_F32C
mlib_SignalIFFTW_1_S16
mlib_SignalIFFTW_1_S16C
mlib_SignalIFFTW_1_S16C_S16C
mlib_SignalIFFTW_1_S16_S16
mlib_SignalIFFTW_1_S16_S16C
mlib_SignalIFFTW_2_F32
mlib_SignalIFFTW_2_F32C
mlib_SignalIFFTW_2_F32C_F32C
mlib_SignalIFFTW_2_F32_F32
mlib_SignalIFFTW_2_F32_F32C
mlib_SignalIFFTW_2_S16C_Mod
mlib_SignalIFFTW_2_S16C_S16C_Mod
mlib_SignalIFFTW_2_S16_Mod
mlib_SignalIFFTW_2_S16_S16C_Mod
mlib_SignalIFFTW_2_S16_S16_Mod
mlib_SignalIFFTW_3_F32
mlib_SignalIFFTW_3_F32C
mlib_SignalIFFTW_3_F32C_F32C
mlib_SignalIFFTW_3_F32_F32
mlib_SignalIFFTW_3_F32_F32C
mlib_SignalIFFTW_3_S16C_Mod
mlib_SignalIFFTW_3_S16C_S16C_Mod
mlib_SignalIFFTW_3_S16_Mod
mlib_SignalIFFTW_3_S16_S16C_Mod
mlib_SignalIFFTW_3_S16_S16_Mod
mlib_SignalIFFTW_4_S16
mlib_SignalIFFTW_4_S16C
mlib_SignalIFFTW_4_S16C_S16C
mlib_SignalIFFTW_4_S16_S16
mlib_SignalIFFTW_4_S16_S16C

mlib_SignalIIR_Biquad_F32_F32
mlib_SignalIIR_Biquad_F32S_F32S
mlib_SignalIIR_Biquad_S16_S16_Sat
mlib_SignalIIR_Biquad_S16S_S16S_Sat
mlib_SignalIIRFree_Biquad_F32_F32
mlib_SignalIIRFree_Biquad_F32S_F32S
mlib_SignalIIRFree_Biquad_S16_S16
mlib_SignalIIRFree_Biquad_S16S_S16S
mlib_SignalIIRFree_P4_F32_F32
mlib_SignalIIRFree_P4_F32S_F32S
mlib_SignalIIRFree_P4_S16_S16
mlib_SignalIIRFree_P4_S16S_S16S
mlib_SignalIIRInit_Biquad_F32_F32
mlib_SignalIIRInit_Biquad_F32S_F32S
mlib_SignalIIRInit_Biquad_S16_S16
mlib_SignalIIRInit_Biquad_S16S_S16S
mlib_SignalIIRInit_P4_F32_F32
mlib_SignalIIRInit_P4_F32S_F32S
mlib_SignalIIRInit_P4_S16_S16
mlib_SignalIIRInit_P4_S16S_S16S
mlib_SignalIIR_P4_F32_F32
mlib_SignalIIR_P4_F32S_F32S
mlib_SignalIIR_P4_S16_S16_Sat
mlib_SignalIIR_P4_S16S_S16S_Sat
mlib_SignalIMDCT_D64
mlib_SignalIMDCT_F32
mlib_SignalIMDCTSplit_D64
mlib_SignalIMDCTSplit_F32
mlib_SignalLimit_F32
mlib_SignalLimit_F32_F32
mlib_SignalLimit_F32S
mlib_SignalLimit_F32S_F32S
mlib_SignalLimit_S16
mlib_SignalLimit_S16S
mlib_SignalLimit_S16_S16
mlib_SignalLimit_S16S_S16S
mlib_SignalLinear2ADPCM2Bits
mlib_SignalLinear2ADPCM3Bits
mlib_SignalLinear2ADPCM4Bits
mlib_SignalLinear2ADPCM5Bits
mlib_SignalLinear2ALaw
mlib_SignalLinear2uLaw
mlib_SignalLMSFilter_F32_F32
mlib_SignalLMSFilter_F32S_F32S
mlib_SignalLMSFilterFree_F32_F32
mlib_SignalLMSFilterFree_F32S_F32S
mlib_SignalLMSFilterFree_S16_S16
mlib_SignalLMSFilterFree_S16S_S16S

mlib_SignalLMSFilterInit_F32_F32
mlib_SignalLMSFilterInit_F32S_F32S
mlib_SignalLMSFilterInit_S16_S16
mlib_SignalLMSFilterInit_S16S_S16S
mlib_SignalLMSFilter_S16_S16_Sat
mlib_SignalLMSFilter_S16S_S16S_Sat
mlib_SignalLPC2Cepstral_F32
mlib_SignalLPC2Cepstral_S16
mlib_SignalLPC2Cepstral_S16_Adp
mlib_SignalLPC2LSP_F32
mlib_SignalLPC2LSP_S16
mlib_SignalLPCAutoCorrel_F32
mlib_SignalLPCAutoCorrelFree_F32
mlib_SignalLPCAutoCorrelFree_S16
mlib_SignalLPCAutoCorrelGetEnergy_F32
mlib_SignalLPCAutoCorrelGetEnergy_S16
mlib_SignalLPCAutoCorrelGetEnergy_S16_Adp
mlib_SignalLPCAutoCorrelGetPARCOR_F32
mlib_SignalLPCAutoCorrelGetPARCOR_S16
mlib_SignalLPCAutoCorrelGetPARCOR_S16_Adp
mlib_SignalLPCAutoCorrelInit_F32
mlib_SignalLPCAutoCorrelInit_S16
mlib_SignalLPCAutoCorrel_S16
mlib_SignalLPCAutoCorrel_S16_Adp
mlib_SignalLPCCovariance_F32
mlib_SignalLPCCovarianceFree_F32
mlib_SignalLPCCovarianceFree_S16
mlib_SignalLPCCovarianceInit_F32
mlib_SignalLPCCovarianceInit_S16
mlib_SignalLPCCovariance_S16
mlib_SignalLPCCovariance_S16_Adp
mlib_SignalLPCPerceptWeight_F32
mlib_SignalLPCPerceptWeightFree_F32
mlib_SignalLPCPerceptWeightFree_S16
mlib_SignalLPCPerceptWeightInit_F32
mlib_SignalLPCPerceptWeightInit_S16
mlib_SignalLPCPerceptWeight_S16
mlib_SignalLPCPitchAnalyze_F32
mlib_SignalLPCPitchAnalyze_S16
mlib_SignalLSP2LPC_F32
mlib_SignalLSP2LPC_S16
mlib_SignalLSP2LPC_S16_Adp
mlib_SignalMelCepstral_F32
mlib_SignalMelCepstralFree_F32
mlib_SignalMelCepstralFree_S16
mlib_SignalMelCepstralInit_F32
mlib_SignalMelCepstralInit_S16
mlib_SignalMelCepstral_S16

mlib_SignalMelCepstral_S16_Adp
mlib_SignalMerge_F32S_F32
mlib_SignalMerge_S16S_S16
mlib_SignalMulBartlett_F32
mlib_SignalMulBartlett_F32_F32
mlib_SignalMulBartlett_F32S
mlib_SignalMulBartlett_F32S_F32S
mlib_SignalMulBartlett_S16
mlib_SignalMulBartlett_S16S
mlib_SignalMulBartlett_S16_S16
mlib_SignalMulBartlett_S16S_S16S
mlib_SignalMulBlackman_F32
mlib_SignalMulBlackman_F32_F32
mlib_SignalMulBlackman_F32S
mlib_SignalMulBlackman_F32S_F32S
mlib_SignalMulBlackman_S16
mlib_SignalMulBlackman_S16S
mlib_SignalMulBlackman_S16_S16
mlib_SignalMulBlackman_S16S_S16S
mlib_SignalMul_F32
mlib_SignalMul_F32_F32
mlib_SignalMul_F32S
mlib_SignalMul_F32S_F32S
mlib_SignalMulHamming_F32
mlib_SignalMulHamming_F32_F32
mlib_SignalMulHamming_F32S
mlib_SignalMulHamming_F32S_F32S
mlib_SignalMulHamming_S16
mlib_SignalMulHamming_S16S
mlib_SignalMulHamming_S16_S16
mlib_SignalMulHamming_S16S_S16S
mlib_SignalMulHanning_F32
mlib_SignalMulHanning_F32_F32
mlib_SignalMulHanning_F32S
mlib_SignalMulHanning_F32S_F32S
mlib_SignalMulHanning_S16
mlib_SignalMulHanning_S16S
mlib_SignalMulHanning_S16_S16
mlib_SignalMulHanning_S16S_S16S
mlib_SignalMulKaiser_F32
mlib_SignalMulKaiser_F32_F32
mlib_SignalMulKaiser_F32S
mlib_SignalMulKaiser_F32S_F32S
mlib_SignalMulKaiser_S16
mlib_SignalMulKaiser_S16S
mlib_SignalMulKaiser_S16_S16
mlib_SignalMulKaiser_S16S_S16S
mlib_SignalMulRectangular_F32

mlib_SignalMulRectangular_F32_F32
mlib_SignalMulRectangular_F32S
mlib_SignalMulRectangular_F32S_F32S
mlib_SignalMulRectangular_S16
mlib_SignalMulRectangular_S16S
mlib_SignalMulRectangular_S16_S16
mlib_SignalMulRectangular_S16S_S16S
mlib_SignalMul_S16_S16_Sat
mlib_SignalMul_S16_Sat
mlib_SignalMul_S16S_S16S_Sat
mlib_SignalMul_S16S_Sat
mlib_SignalMulSAdd_F32
mlib_SignalMulSAdd_F32_F32
mlib_SignalMulSAdd_F32S
mlib_SignalMulSAdd_F32S_F32S
mlib_SignalMulSAdd_S16_S16_Sat
mlib_SignalMulSAdd_S16_Sat
mlib_SignalMulSAdd_S16S_S16S_Sat
mlib_SignalMulSAdd_S16S_Sat
mlib_SignalMulS_F32
mlib_SignalMulS_F32_F32
mlib_SignalMulS_F32S
mlib_SignalMulS_F32S_F32S
mlib_SignalMulShift_S16_S16_Sat
mlib_SignalMulShift_S16_Sat
mlib_SignalMulShift_S16S_S16S_Sat
mlib_SignalMulShift_S16S_Sat
mlib_SignalMulS_S16_S16_Sat
mlib_SignalMulS_S16_Sat
mlib_SignalMulS_S16S_S16S_Sat
mlib_SignalMulS_S16S_Sat
mlib_SignalMulSShiftAdd_S16_S16_Sat
mlib_SignalMulSShiftAdd_S16_Sat
mlib_SignalMulSShiftAdd_S16S_S16S_Sat
mlib_SignalMulSShiftAdd_S16S_Sat
mlib_SignalMulSShift_S16_S16_Sat
mlib_SignalMulSShift_S16_Sat
mlib_SignalMulSShift_S16S_S16S_Sat
mlib_SignalMulSShift_S16S_Sat
mlib_SignalMulWindow_F32
mlib_SignalMulWindow_F32_F32
mlib_SignalMulWindow_F32S
mlib_SignalMulWindow_F32S_F32S
mlib_SignalMulWindow_S16
mlib_SignalMulWindow_S16S
mlib_SignalMulWindow_S16_S16
mlib_SignalMulWindow_S16S_S16S
mlib_SignalQuant2_S16_F32

mlib_SignalQuant2_S16S_F32S
mlib_SignalQuant_S16_F32
mlib_SignalQuant_S16S_F32S
mlib_SignalQuant_U8_F32
mlib_SignalQuant_U8_S16
mlib_SignalQuant_U8S_F32S
mlib_SignalQuant_U8S_S16S
mlib_SignalReSampleFIR_F32_F32
mlib_SignalReSampleFIR_F32S_F32S
mlib_SignalReSampleFIRFree_F32_F32
mlib_SignalReSampleFIRFree_F32S_F32S
mlib_SignalReSampleFIRFree_S16_S16
mlib_SignalReSampleFIRFree_S16S_S16S
mlib_SignalReSampleFIRInit_F32_F32
mlib_SignalReSampleFIRInit_F32S_F32S
mlib_SignalReSampleFIRInit_S16_S16
mlib_SignalReSampleFIRInit_S16S_S16S
mlib_SignalReSampleFIR_S16_S16_Sat
mlib_SignalReSampleFIR_S16S_S16S_Sat
mlib_SignalSineWave_F32
mlib_SignalSineWaveFree_F32
mlib_SignalSineWaveFree_S16
mlib_SignalSineWaveInit_F32
mlib_SignalSineWaveInit_S16
mlib_SignalSineWave_S16
mlib_SignalSplit_F32_F32S
mlib_SignalSplit_S16_S16S
mlib_SignaluLaw2ALaw
mlib_SignaluLaw2Linear
mlib_SignalUpSample_F32_F32
mlib_SignalUpSample_F32S_F32S
mlib_SignalUpSampleFIR_F32_F32
mlib_SignalUpSampleFIR_F32S_F32S
mlib_SignalUpSampleFIRFree_F32_F32
mlib_SignalUpSampleFIRFree_F32S_F32S
mlib_SignalUpSampleFIRFree_S16_S16
mlib_SignalUpSampleFIRFree_S16S_S16S
mlib_SignalUpSampleFIRInit_F32_F32
mlib_SignalUpSampleFIRInit_F32S_F32S
mlib_SignalUpSampleFIRInit_S16_S16
mlib_SignalUpSampleFIRInit_S16S_S16S
mlib_SignalUpSampleFIR_S16_S16_Sat
mlib_SignalUpSampleFIR_S16S_S16S_Sat
mlib_SignalUpSample_S16_S16
mlib_SignalUpSample_S16S_S16S
mlib_SignalWhiteNoise_F32

mlib_SignalWhiteNoiseFree_F32
mlib_SignalWhiteNoiseFree_S16
mlib_SignalWhiteNoiseInit_F32
mlib_SignalWhiteNoiseInit_S16
mlib_SignalWhiteNoise_S16

Video Processing Functions

mlib_VideoAddBlock_U8_S16
mlib_VideoColorABGR2JFIFYCC420
mlib_VideoColorABGR2JFIFYCC422
mlib_VideoColorABGR2JFIFYCC444
mlib_VideoColorABGR2RGB
mlib_VideoColorABGRint_to_ARGBint
mlib_VideoColorARGB2JFIFYCC420
mlib_VideoColorARGB2JFIFYCC422
mlib_VideoColorARGB2JFIFYCC444
mlib_VideoColorARGB2RGB
mlib_VideoColorBGRAint_to_ABGRint
mlib_VideoColorBGRint_to_ABGRint
mlib_VideoColorBlendABGR
mlib_VideoColorBlendABGR_Inp
mlib_VideoColorBlendABGR_ResetAlpha
mlib_VideoColorBlendABGR_ResetAlpha_Inp
mlib_VideoColorCMYK2JFIFYCCK444
mlib_VideoColorJFIFYCC2ABGR444
mlib_VideoColorJFIFYCC2ARGB444
mlib_VideoColorJFIFYCC2RGB420
mlib_VideoColorJFIFYCC2RGB420_Nearest
mlib_VideoColorJFIFYCC2RGB422
mlib_VideoColorJFIFYCC2RGB422_Nearest
mlib_VideoColorJFIFYCC2RGB444
mlib_VideoColorJFIFYCC2RGB444_S16
mlib_VideoColorJFIFYCCK2CMYK444
mlib_VideoColorMerge2
mlib_VideoColorMerge2_S16
mlib_VideoColorMerge3
mlib_VideoColorMerge3_S16
mlib_VideoColorMerge4
mlib_VideoColorMerge4_S16
mlib_VideoColorResizeABGR
mlib_VideoColorRGB2ABGR
mlib_VideoColorRGB2ARGB
mlib_VideoColorRGB2JFIFYCC420
mlib_VideoColorRGB2JFIFYCC422
mlib_VideoColorRGB2JFIFYCC444
mlib_VideoColorRGB2JFIFYCC444_S16
mlib_VideoColorRGBAint_to_ABGRint
mlib_VideoColorRGBint_to_ABGRint
mlib_VideoColorRGBseq_to_ABGRint

mlib_VideoColorRGBXint_to_ABGRint
mlib_VideoColorRGBXint_to_ARGBint
mlib_VideoColorSplit2
mlib_VideoColorSplit2_S16
mlib_VideoColorSplit3
mlib_VideoColorSplit3_S16
mlib_VideoColorSplit4
mlib_VideoColorSplit4_S16
mlib_VideoColorUYV444int_to_ABGRint
mlib_VideoColorUYV444int_to_ARGBint
mlib_VideoColorUYV444int_to_UYVY422int
mlib_VideoColorUYV444int_to_YUYV422int
mlib_VideoColorUYVY422int_to_ABGRint
mlib_VideoColorUYVY422int_to_ARGBint
mlib_VideoColorXRGBint_to_ABGRint
mlib_VideoColorXRGBint_to_ARGBint
mlib_VideoColorYUV2ABGR411
mlib_VideoColorYUV2ABGR420
mlib_VideoColorYUV2ABGR420_W
mlib_VideoColorYUV2ABGR420_WX2
mlib_VideoColorYUV2ABGR420_WX3
mlib_VideoColorYUV2ABGR420_X2
mlib_VideoColorYUV2ABGR420_X3
mlib_VideoColorYUV2ABGR422
mlib_VideoColorYUV2ABGR444
mlib_VideoColorYUV2ARGB411
mlib_VideoColorYUV2ARGB420
mlib_VideoColorYUV2ARGB422
mlib_VideoColorYUV2ARGB444
mlib_VideoColorYUV2RGB411
mlib_VideoColorYUV2RGB420
mlib_VideoColorYUV2RGB422
mlib_VideoColorYUV2RGB444
mlib_VideoColorYUV411seq_to_ABGRint
mlib_VideoColorYUV411seq_to_ARGBint
mlib_VideoColorYUV411seq_to_UYVY422int
mlib_VideoColorYUV411seq_to_YUYV422int
mlib_VideoColorYUV420seq_to_ABGRint
mlib_VideoColorYUV420seq_to_ARGBint
mlib_VideoColorYUV420seq_to_UYVY422int
mlib_VideoColorYUV420seq_to_YUYV422int
mlib_VideoColorYUV422seq_to_ABGRint
mlib_VideoColorYUV422seq_to_ARGBint
mlib_VideoColorYUV422seq_to_UYVY422int
mlib_VideoColorYUV422seq_to_YUYV422int
mlib_VideoColorYUV444int_to_ABGRint
mlib_VideoColorYUV444int_to_ARGBint
mlib_VideoColorYUV444int_to_UYVY422int

mlib_VideoColorYUV444int_to_YUYV422int
mlib_VideoColorYUV444seq_to_ABGRint
mlib_VideoColorYUV444seq_to_ARGBint
mlib_VideoColorYUV444seq_to_UYVY422int
mlib_VideoColorYUV444seq_to_YUYV422int
mlib_VideoColorYUYV422int_to_ABGRint
mlib_VideoColorYUYV422int_to_ARGBint
mlib_VideoCopyRefAve_U8_U8
mlib_VideoCopyRefAve_U8_U8_16x16
mlib_VideoCopyRefAve_U8_U8_16x8
mlib_VideoCopyRefAve_U8_U8_8x16
mlib_VideoCopyRefAve_U8_U8_8x4
mlib_VideoCopyRefAve_U8_U8_8x8
mlib_VideoCopyRef_S16_U8
mlib_VideoCopyRef_S16_U8_16x16
mlib_VideoCopyRef_S16_U8_16x8
mlib_VideoCopyRef_S16_U8_8x16
mlib_VideoCopyRef_S16_U8_8x4
mlib_VideoCopyRef_S16_U8_8x8
mlib_VideoCopyRef_U8_U8
mlib_VideoCopyRef_U8_U8_16x16
mlib_VideoCopyRef_U8_U8_16x8
mlib_VideoCopyRef_U8_U8_8x16
mlib_VideoCopyRef_U8_U8_8x4
mlib_VideoCopyRef_U8_U8_8x8
mlib_VideoDCT16x16_S16_S16
mlib_VideoDCT16x16_S16_S16_B10
mlib_VideoDCT2x2_S16_S16
mlib_VideoDCT4x4_S16_S16
mlib_VideoDCT8x8_S16_S16
mlib_VideoDCT8x8_S16_S16_B12
mlib_VideoDCT8x8_S16_S16_NA
mlib_VideoDCT8x8_S16_U8
mlib_VideoDCT8x8_S16_U8_NA
mlib_VideoDeQuantizeInit_S16
mlib_VideoDeQuantize_S16
mlib_VideoDownSample420
mlib_VideoDownSample420_S16
mlib_VideoDownSample422
mlib_VideoDownSample422_S16
mlib_VideoH263OverlappedMC_S16_U8
mlib_VideoH263OverlappedMC_U8_U8
mlib_VideoIDCT8x8_S16_S16
mlib_VideoIDCT8x8_S16_S16_DC
mlib_VideoIDCT8x8_S16_S16_NA
mlib_VideoIDCT8x8_S16_S16_Q1
mlib_VideoIDCT8x8_S16_S16_Q1_Mismatch
mlib_VideoIDCT8x8_U8_S16

mlib_VideoIDCT8x8_U8_S16_DC
mlib_VideoIDCT8x8_U8_S16_NA
mlib_VideoIDCT8x8_U8_S16_Q1
mlib_VideoIDCT_IEEE_S16_S16
mlib_VideoInterpAveX_U8_U8
mlib_VideoInterpAveX_U8_U8_16x16
mlib_VideoInterpAveX_U8_U8_16x8
mlib_VideoInterpAveX_U8_U8_8x16
mlib_VideoInterpAveX_U8_U8_8x4
mlib_VideoInterpAveX_U8_U8_8x8
mlib_VideoInterpAveXY_U8_U8
mlib_VideoInterpAveXY_U8_U8_16x16
mlib_VideoInterpAveXY_U8_U8_16x8
mlib_VideoInterpAveXY_U8_U8_8x16
mlib_VideoInterpAveXY_U8_U8_8x4
mlib_VideoInterpAveXY_U8_U8_8x8
mlib_VideoInterpAveY_U8_U8
mlib_VideoInterpAveY_U8_U8_16x16
mlib_VideoInterpAveY_U8_U8_16x8
mlib_VideoInterpAveY_U8_U8_8x16
mlib_VideoInterpAveY_U8_U8_8x4
mlib_VideoInterpAveY_U8_U8_8x8
mlib_VideoInterpX_S16_U8
mlib_VideoInterpX_S16_U8_16x16
mlib_VideoInterpX_S16_U8_16x8
mlib_VideoInterpX_S16_U8_8x16
mlib_VideoInterpX_S16_U8_8x4
mlib_VideoInterpX_S16_U8_8x8
mlib_VideoInterpX_U8_U8
mlib_VideoInterpX_U8_U8_16x16
mlib_VideoInterpX_U8_U8_16x8
mlib_VideoInterpX_U8_U8_8x16
mlib_VideoInterpX_U8_U8_8x4
mlib_VideoInterpX_U8_U8_8x8
mlib_VideoInterpXY_S16_U8
mlib_VideoInterpXY_S16_U8_16x16
mlib_VideoInterpXY_S16_U8_16x8
mlib_VideoInterpXY_S16_U8_8x16
mlib_VideoInterpXY_S16_U8_8x4
mlib_VideoInterpXY_S16_U8_8x8
mlib_VideoInterpXY_U8_U8
mlib_VideoInterpXY_U8_U8_16x16
mlib_VideoInterpXY_U8_U8_16x8
mlib_VideoInterpXY_U8_U8_8x16
mlib_VideoInterpXY_U8_U8_8x4
mlib_VideoInterpXY_U8_U8_8x8
mlib_VideoInterpX_Y_XY_U8_U8
mlib_VideoInterpY_S16_U8

mlib_VideoInterpY_S16_U8_16x16
mlib_VideoInterpY_S16_U8_16x8
mlib_VideoInterpY_S16_U8_8x16
mlib_VideoInterpY_S16_U8_8x4
mlib_VideoInterpY_S16_U8_8x8
mlib_VideoInterpY_U8_U8
mlib_VideoInterpY_U8_U8_16x16
mlib_VideoInterpY_U8_U8_16x8
mlib_VideoInterpY_U8_U8_8x16
mlib_VideoInterpY_U8_U8_8x4
mlib_VideoInterpY_U8_U8_8x8
mlib_VideoP64Decimate_U8_U8
mlib_VideoP64Loop_S16_U8
mlib_VideoP64Loop_U8_U8
mlib_VideoQuantizeInit_S16
mlib_VideoQuantize_S16
mlib_VideoReversibleColorRGB2YUV_S16_S16
mlib_VideoReversibleColorRGB2YUV_S16_U8
mlib_VideoReversibleColorRGB2YUV_S32_S16
mlib_VideoReversibleColorRGB2YUV_U8_U8
mlib_VideoReversibleColorYUV2RGB_S16_S16
mlib_VideoReversibleColorYUV2RGB_S16_S32
mlib_VideoReversibleColorYUV2RGB_U8_S16
mlib_VideoReversibleColorYUV2RGB_U8_U8
mlib_VideoSignMagnitudeConvert_S16
mlib_VideoSignMagnitudeConvert_S16_S16
mlib_VideoSignMagnitudeConvert_S32
mlib_VideoSignMagnitudeConvert_S32_S32
mlib_VideoSumAbsDiff
mlib_VideoUpSample420
mlib_VideoUpSample420_Nearest
mlib_VideoUpSample420_Nearest_S16
mlib_VideoUpSample420_S16
mlib_VideoUpSample422
mlib_VideoUpSample422_Nearest
mlib_VideoUpSample422_Nearest_S16
mlib_VideoUpSample422_S16
mlib_VideoWaveletForwardTwoTenTrans_S16_S16
mlib_VideoWaveletForwardTwoTenTrans_S16_U8
mlib_VideoWaveletForwardTwoTenTrans_S32_S16
mlib_VideoWaveletForwardTwoTenTrans_S32_S32
mlib_VideoWaveletInverseTwoTenTrans_S16_S16
mlib_VideoWaveletInverseTwoTenTrans_S16_S32
mlib_VideoWaveletInverseTwoTenTrans_S32_S32
mlib_VideoWaveletInverseTwoTenTrans_U8_S16

Volume Imaging Functions

mlib_VolumeFindMaxBMask_S16
mlib_VolumeFindMaxBMask_U8
mlib_VolumeFindMaxCMask_S16

mlib_VolumeFindMaxCMask_U8
mlib_VolumeFindMax_S16
mlib_VolumeFindMax_U8
mlib_VolumeRayCast_Blocked_Divergent_Nearest_S16_S16
mlib_VolumeRayCast_Blocked_Divergent_Nearest_U8_U8
mlib_VolumeRayCast_Blocked_Divergent_Trilinear_S16_S16
mlib_VolumeRayCast_Blocked_Divergent_Trilinear_U8_U8
mlib_VolumeRayCast_Blocked_Parallel_Nearest_S16_S16
mlib_VolumeRayCast_Blocked_Parallel_Nearest_U8_U8
mlib_VolumeRayCast_Blocked_Parallel_Trilinear_S16_S16
mlib_VolumeRayCast_Blocked_Parallel_Trilinear_U8_U8
mlib_VolumeRayCast_General_Divergent_Nearest_S16_S16
mlib_VolumeRayCast_General_Divergent_Nearest_U8_Bit
mlib_VolumeRayCast_General_Divergent_Nearest_U8_U8
mlib_VolumeRayCast_General_Divergent_Trilinear_S16_S16
mlib_VolumeRayCast_General_Divergent_Trilinear_U8_U8
mlib_VolumeRayCast_General_Parallel_Nearest_S16_S16
mlib_VolumeRayCast_General_Parallel_Nearest_U8_Bit
mlib_VolumeRayCast_General_Parallel_Nearest_U8_U8
mlib_VolumeRayCast_General_Parallel_Trilinear_S16_S16
mlib_VolumeRayCast_General_Parallel_Trilinear_U8_U8
mlib_VolumeWindowLevel

FILES

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

ATTRIBUTES

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

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

SEE ALSO

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

mediaLib Reference Manual

mediaLib User's Manual

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

`/lib/libmp.so.1`	shared object for binary compatibility only
`/lib/libmp.so.2`	shared object
`/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 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`	`memalign`
`realloc`	`valloc`

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

libmvec – vector math library

SYNOPSIS

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

DESCRIPTION

This library contains function to evaluate common mathematical functions for several arguments at once. The argument values are specified by one or more vectors (arrays) of data, and the corresponding result values are stored in another vector.

INTERFACES

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

`vatan_`	`vatanf_`
`vatan2_`	`vatan2f_`
`vc_abs_`	`vc_exp_`
`vc_log_`	`vc_pow_`
`vcos_`	`vcosf_`
`vexp_`	`vexpf_`
`vhypot_`	`vhypotf_`
`vlog_`	`vlogf_`
`vpow_`	`vpowf_`
`vrhypot_`	`vrhypotf_`
`vrsqrt_`	`vrsqrtf_`
`vsin_`	`vsinf_`
`vsincos_`	`vsincosf_`
`vsqrt_`	`vsqrtf_`
`vz_abs_`	`vz_exp_`
`vz_log_`	`vz_pow_`

FILES

`/lib/libmvec.so.1`	shared object
`/lib/64/libmvec.so.1`	64-bit shared object

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWlibmsr
MT-Level	MT-Safe

SEE ALSO

`intro`(3), `vatan_`(3MVEC), `vc_abs_`(3MVEC), `attributes`(5)

NAME

libnls – network listener service library

SYNOPSIS

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

DESCRIPTION

The functions in this library interact with the network listener daemon, `listen`(1M). The functions are provided for services invoked by the listener daemon and for clients that connect to the services using `listen`.

INTERFACES

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

`nlsgetcall` `nlsprovider`

`nlsrequest`

FILES

`/usr/lib/libnls.so.1`	shared object
`/usr/lib/64/libnls.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

`listen`(1M), `intro`(3), `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.

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	getipsecalgbyname
getipsecalgbynum	getipsecprotobyname
getipsecprotobynum	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

`/lib/libnsl.so.1`	shared object
`/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 (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

The `libnvpair` library exports a set of functions for managing name-value pairs.

The library defines four opaque handles:

`nvpair_t`	handle to a name-value pair
`nvlist_t`	handle to a list of name-value pairs
`nv_alloc_t`	handle to a pluggable allocator
`nv_alloc_ops_t`	handle to pluggable allocator operations

The library supports the following operations:

- Allocate and free an `nvlist_t`.
- Specify the allocater to be used when manipulating an `nvlist_t`.
- Add and remove an `nvpair_t` from a list.
- Search `nvlist_t` for a specified name pair.
- Pack an `nvlist_t` into a contiguous buffer.
- Expand a packed nvlist into a searchable `nvlist_t`.

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_boolean_value`
`nvlist_add_boolean_array`	`nvlist_add_byte`
`nvlist_add_byte_array`	`nvlist_add_int8`
`nvlist_add_int8_array`	`nvlist_add_int16`
`nvlist_add_int16_array`	`nvlist_add_int32`
`nvlist_add_int32_array`	`nvlist_add_int64`
`nvlist_add_int64_array`	`nvlist_add_nvlist`
`nvlist_add_nvlist_array`	`nvlist_add_nvpair`
`nvlist_add_string`	`nvlist_add_string_array`
`nvlist_add_uint8`	`nvlist_add_uint8_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_boolean_value	nvlist_lookup_boolean_array
nvlist_lookup_byte	nvlist_lookup_byte_array
nvlist_lookup_int8	nvlist_lookup_int8_array
nvlist_lookup_int16	nvlist_lookup_int16_array
nvlist_lookup_int32	nvlist_lookup_int32_array
nvlist_lookup_int64	nvlist_lookup_int64_array
nvlist_lookup_nvlist	nvlist_lookup_nvlist_array
nvlist_lookup_nv_alloc	nvlist_lookup_pairs
nvlist_lookup_string	nvlist_lookup_string_array
nvlist_lookup_uint8	nvlist_lookup_uint8_array
nvlist_lookup_uint16	nvlist_lookup_uint16_array
nvlist_lookup_uint32	nvlist_lookup_uint32_array
nvlist_lookup_uint64	nvlist_lookup_uint64_array
nvlist_merge	nvlist_next_nvpair
nvlist_pack	nvlist_remove
nvlist_remove_all	nvlist_size
nvlist_unpack	nvlist_xalloc
nvlist_xdup	nvlist_xpack
nvlist_xunpack	nvpair_name
nvpair_type	nvpair_value_boolean_array
nvpair_value_boolean_value	nvpair_value_byte
nvpair_value_byte_array	nvpair_value_int8
nvpair_value_int8_array	nvpair_value_int16
nvpair_value_int16_array	nvpair_value_int32
nvpair_value_int32_array	nvpair_value_int64
nvpair_value_int64_array	nvpair_value_nvlist
nvpair_value_nvlist_array	nvpair_value_string
nvpair_value_string_array	nvpair_value_uint8

nvpair_value_uint8_array	nvpair_value_uint16
nvpair_value_uint16_array	nvpair_value_uint32
nvpair_value_uint32_array	nvpair_value_uint64
nvpair_value_uint64_array	nv_alloc_init
nv_alloc_fini	nv_alloc_reset

FILES

/lib/libnvpair.so.1	shared object
/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)
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO

intro(3), 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

`/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

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsl
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_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 is no longer supported. 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.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 (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_find_node`	`picl_get_first_prop`
`picl_get_frutree_parent`	`picl_get_next_by_col`
`picl_get_next_by_row`	`picl_get_next_prop`
`picl_get_node_by_path`	`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_frutree_parent`
`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
MT-Level	MT-Safe

SEE ALSO

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

NAME

libpkcs11 – PKCS#11 Cryptographic Framework library

SYNOPSIS

```
cc [ flag... ] file... -lpkcs11 [ library... ]
#include <security/pkcs11.h>
#include <security/cryptki.h>
```

DESCRIPTION

The `libpkcs11` library implements the RSA Security Inc. PKCS#11 Cryptographic Token Interface (Cryptoki), v2.11 specification by using plug-ins to provide the slots.

Each plug-in, which also implements RSA PKCS#11 v2.11, represents one or more slots. If more than one plug-in is available, a meta slot is created that provides the mechanisms of all plug-ins in a single slot. When available, the meta slot is always the first slot provided by `libpkcs11`. The meta slot does not provide token-based objects.

This library filters the list of mechanisms available from plug-ins based on the policy set by `cryptoadm`(1M).

This library provides entry points for all PKCS#11 v2.11 functions. See the RSA PKCS#11 v2.11 specification at http://www.rsasecurity/rsalabs/pkcs/pkcs-11.

Plug-ins are added to `libpkcs11` by the `pkcs11conf` class action script during execution of `pkgadd`(1M). The available mechanisms are administered by the `cryptoadm`(1M) utility.

Plug-ins must have all of their library dependancies specified, including `libc`(3LIB). Libraries that have unresolved symbols, including those from `libc`, will be rejected and a message will be sent to `syslog`(3C) for such plug-ins.

Due to U.S. Export regulations, all plug-ins are required to be cryptographically signed using the `elfsign` utility.

Any plug-in that is not signed or is not a compatible version of PKCS#11 will be dropped by `libpkcs11`. When a plug-in is dropped, the administrator is alerted by the `syslog`(3C) utility.

The `<security/pkcs11f.h>` header contains function definitions. The `<security/pkcs11t.h>` header contains type definitions. Applications can include either of these headers in place of `<security/pkcs11.h>`, which contains both function and type definitions.

INTERFACES

The shared object `libpkcs11.so.1` provides the public interfaces defined below. See `intro`(3) for additional information on shared object interfaces.

PKCS#11 Standard

`C_CloseAllSessions`	`C_CloseSession`
`C_CopyObject`	`C_CreateObject`
`C_Decrypt`	`C_DecryptDigestUpdate`
`C_DecryptFinal`	`C_DecryptInit`

C_DecryptUpdate	C_DecryptVerifyUpdate
C_DeriveKey	C_DestroyObject
C_Digest	C_DigestEncryptUpdate
C_DigestFinal	C_DigestInit
C_DigestKey	C_DigestUpdate
C_Encrypt	C_EncryptFinal
C_EncryptInit	C_EncryptUpdate
C_Finalize	C_FindObjects
C_FindObjectsFinal	C_FindObjectsInit
C_GenerateKey	C_GenerateKeyPair
C_GenerateRandom	C_GetAttributeValue
C_GetFunctionList	C_GetInfo
C_GetMechanismInfo	C_GetMechanismList
C_GetObjectSize	C_GetOperationState
C_GetSessionInfo	C_GetSlotInfo
C_GetSlotList	C_GetTokenInfo
C_InitPIN	C_InitToken
C_Initialize	C_Login
C_Logout	C_OpenSession
C_SeedRandom	C_SetAttributeValue
C_SetOperationState	C_SetPIN
C_Sign	C_SignEncryptUpdate
C_SignFinal	C_SignInit
C_SignRecover	C_SignRecoverInit
C_SignUpdate	C_UnwrapKey
C_Verify	C_VerifyFinal
C_VerifyInit	C_VerifyRecover
C_VerifyRecoverInit	C_VerifyUpdate
C_WaitForSlotEvent	C_WrapKey

SUNW Extensions

`SUNW_C_GetMechSession` `SUNW_C_KeyToObject`

FILES

`/usr/lib/libpkcs11.so.1`	shared object
`/usr/lib/64/libpkcs11.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	See below.
MT-Level	See below.

The SUNW Extension functions are Evolving. The PKCS#11 Standard functions are Standard: PKCS#11 v2.11.

The SUNW Extension functions are MT-Safe. The PKCS#11 Standard functions are MT-Safe with exceptions. See Section 6.5.2 of RSA PKCS#11 v2.11.

SEE ALSO

`cryptoadm`(1M), `pkgadd`(1M), `intro`(3), `SUNW_C_GetMechSession`(3EXT), `syslog`(3C), `attributes`(5) , `pkcs11_kernel`(5), `pkcs11_softtoken`(5)

RSA PKCS#11 v2.11 http://www.rsasecurity/rsalabs/pkcs/pkcs-11

NOTES

If an application calls `C_WaitForSlotEvent()` without the `CKF_DONT_BLOCK` flag set, `libpkcs11` must create threads internally. If, however, `CKF_LIBRARY_CANT_CREATE_OS_THREADS` is set, `C_WaitForSlotEvent()` returns `CKR_FUNCTION_FAILED`.

The PKCS#11 library does not work with Netscape 4.*x* but does work with more recent versions of Netscape and Mozilla.

Because `C_Initalize()` might have been called by both an application and a library, it is not safe for a library or its plugins to call `C_Finalize()`. A library can be finished calling functions from `libpkcs11`, while an application might not.

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`
`erase`	`label`
`line`	`linemod`
`move`	`openpl`
`openvt`	`point`
`space`	

FILES

`/usr/lib/libplot.so.1`	shared object
`/usr/lib/64/libplot.so.1`	64-bit shared object
`/usr/lib/lib300.so.1`	shared object
`/usr/lib/64/lib300.so.1`	64-bit shared object
`/usr/lib/lib300s.so.1`	shared object
`/usr/lib/64/lib300s.so.1`	64-bit shared object
`/usr/lib/lib4014.so.1`	shared object
`/usr/lib/64/lib4014.so.1`	64-bit shared object
`/usr/lib/lib450.so.1`	shared object
`/usr/lib/64/lib450.so.1`	64-bit shared object
`/usr/lib/libvt0.so.1`	shared object
`/usr/lib/64/libvt0.so.1`	64-bit shared object

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsl (32–bit)

ATTRIBUTE TYPE	ATTRIBUTE VALUE
	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.

When the pools facility is enabled on a system, the behavior of the following functions is modified.

System Call	Error Value
`pset_assign`(pset !=`PS_QUERY`)	`ENOTSUP`
`pset_bind`(pset !=`PS_QUERY`)	`ENOTSUP`
`pset_create()`	`ENOTSUP`
`pset_destroy()`	`ENOTSUP`
`pset_setattr()`	`ENOTSUP`

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 can 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.allocate-method	string	Allocation method to use when this configuration is instantiated
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.name	string	User name for the configuration
system.version	int	libpool version required to manipulate this configuration
system.poold.log-level	string	poold logging level
system.poold.log-location	string	poold logging location
system.poold.history-file	string	poold decision history location
system.poold.monitor-interval	uint	poold monitoring sample interval
system.poold.objectives	string	poold objectives for a system.

The system.allocate-method, system.bind-default, system.comment, system.name, system.poold.log-level, system.poold.log-location, system.poold.history-file, system.poold.monitor-interval, and system.poold.objectives properties are writable; the system.version property is not.

The system.allocate-method property accepts only two values, "importance based" and "surplus to default". The default value for this property is "importance based". The property is optional and if it is not present the library will allocate resources as though it were present and had the default value. These strings are defined in <pool.h> as POA_IMPORTANCE and POA_SURPLUS_TO_DEFAULT.

If "importance based" allocation is defined, then during a commit the library will allocate resources to pools using an algorithm that observes minimum and maximum constraints for resources but favors those resources with greater importance.

If "surplus to default" is defined, then during a commit the library will allocate minimum resources to all resource sets apart from default which will receive any surplus.

The system.bind-default property defaults to true. This property interacts with the project.pool resource control to specify the binding behavior for processes associated with a project. If project.pool is not specified, then this property has no effect. If project.pool is specified and the specified pool exists, this property has no effect. If the specified pool does not exist, perhaps because of a reconfiguration, then this property controls the binding behavior for the project member. If system.bind-default is true, then the project member is bound to the default pool

(identified as the pool for which `pool.default` is true); otherise the project member is refused access to the system. Care should be taken with the pools configuration if this property is set to false, so as to avoid denying users access to the system.

The various `poold` properties are used to configure the operation of `poold`(1M).

The `system.poold.log-level` property is used to specify the level of detail provided in log messages. Valid values are: `ALERT`, `CRIT`, `ERR`, `WARNING`, `NOTICE`, `INFO`, and `DEBUG`.

`ALERT` provides the least level of detail, `DEBUG` the greatest. See `syslog`(3C) for more information about the meaning of these debug levels. If this property is not specified, the default value `NOTICE` is used.

The `system.poold.log-location` property is used to specify the location of the logfiles generated by `poold`. The special value of "syslog" indicates that logged messages should be written to `syslog()`. If this property is not specified, the default location `/var/log/pool` is used.

The `system.poold.history-file` specifies the location of the decision history file which is used by `poold` to improve the quality of its decision making over time. If this property is not specified, the default location `/var/adm/pool` is used.

The `system.poold.monitor-interval` property specifies the monitoring interval (in milliseconds) to be used by `poold` when sampling utilization statistics. If this property is not specified, the default value of 15 seconds is used.

The `system.poold.objectives` property specifies any system wide objectives. An objectives property has the following syntax:

```
objectives = objective [; objective]*
objective = [n:] keyword [op] [value]
```

All objectives are prefixed with an optional importance. The importance acts as a multiplier for the objective and thus increases the significance of its contribution to the objective function evaluation. If no importance is specified, the default value is 1.

The "wt-load" objective is the only objective to which a system element can be set. This objective favors configurations that match resource allocations to resource utilization. A resource set that uses more resources will be given more resources when this objective is active. An administrator should use this objective when he is relatively satisfied with the constraints established using the minimum and maximum properties and would like the DRP to manipulate resources freely within those constraints.

Pools

Property name	Type	Description
`pool.active`	boolean	Mark this pool as active, if true.

Property name	Type	Description
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.
pool.sys_id	int	System-assigned pool ID.

The pool.default and pool.sys_id properties are not writable; all other listed properties are writable.

If pool.scheduler is specified, it must be set to the name of a valid scheduling class for the system. See the -c option for priocntl(1) for a list of valid class names.

Processor Sets

Property name	Type	Description
pset.comment	string	User description of resource.
pset.default	boolean	Marks default processor set.
pset.load	uint	The load for this processor set.
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.

Property name	Type	Description
`pset.poold.objectives`	string	Specifies the poold objectives for a pset.

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

The `pset.load` property represents the load on a processor set. The lowest value for this property is 0. The value of `pset.load` increases in a linear fashion with the load on the set, as measured by the number of jobs in the system run queue.

The `pset.poold.objectives` property specifies an objective which is specific to a particular `pset`. See the `system.poold.objectives` entry for the specification of this property's syntax.

There are two types of objectives that can be set on a `pset`:

`locality` This objective influences the impact that locality, as measured by lgroup data, has upon the chosen configuration. This objective can take one of three values:

- `tight` If set, configurations that maximize resource locality are favored.
- `loose` If set, configurations that minimize resource locality are favored.
- `none` This is the default value for this objective. If set, configuration favorability is uninfluenced by resource locality.

`utilization` This objective favors configurations that allocate resources to partitions that are failing to preserve the specified utilization objective.

These objectives are specified in terms of an operator and a value. The operators are

`<` The "less than" operator is used to indicate that the specified value should be treated as a maximum target value.

`>` The "greater than" operator is used to indicate that the specified value should be treated as a minimum target value.

`~` The "about" operator is used to indicate that the specified value should be treated as a target value about which some fluctuation is acceptable.

Only one objective of each type of operator can be set. For example, if the `~` operator is set, the `<` and `>` operators cannot be set. It is possible to set a `<` and a `>` operator together; the values will be validated to ensure that they do not overlap.

Processors

Property name	Type	Description
`cpu.comment`	string	User description of CPU.
`cpu.pinned`	boolean	CPU pinned to this processor set.
`cpu.status`	int	Processor status, on-line, offline or interrupts disabled.
`cpu.sys_id`	int	System-assigned processor ID.

The `cpu.comment`, `cpu.pinned`, and `cpu.status` properties are writeable.

The `cpu.status` property can be set only to the following values:

`off-line` Set the CPU offline.

`on-line` Set the CPU online.

`no-intr` Disable interrupt processing on the CPU.

These values are defined in `<sys/processor.h>` as the `PS_OFFLINE`, `PS_ONLINE`, and `PS_NOINTR` macros.

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_update`
`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_get_status	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_type_list
pool_resource_xtransfer	pool_rm_property
pool_set_binding	pool_set_status
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/64/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	Unstable

ATTRIBUTE TYPE	ATTRIBUTE VALUE
MT-Level	Safe

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 can be used to manipulate static configurations even when the pools facility is not enabled. See pooladm(1M) and pool_set_status(3POOL) for more information about enabling the pools facility. The pools facility must be enabled, however, to modify the dynamic configuration.

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

Historically, functions in this library provided POSIX threading support. See `standards`(5). This functionality now resides in `libc`(3LIB).

This library is maintained to provide backward compatibility for both runtime and compilation environments. The shared object is implemented as a filter on `libc.so.1`. New application development needs to specify `-lpthread` only to obtain POSIX semantics for `fork`(2) that assumes the behavior of `fork1`(2) rather than the default behavior that forks all threads.

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

/lib/libpthread.so.1	a filter on /lib/libc.so.1
/lib/64/libpthread.so.1	a filter on /lib/64/libc.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), libc(3LIB), libc_db(3LIB), libthread(3LIB), attributes(5), standards(5), threads(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_unset`	`xdrrec_create`
`xdrrec_endofrecord`	`xdrrec_eof`
`xdrrec_readbytes`	`xdrrec_skiprecord`

FILES

`/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 (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

/lib/libresolv.so.1	shared object for backward compatibility only
/lib/64/libresolv.so.1	64–bit shared object for backward compatibility only
/lib/libresolv.so.2	shared object
/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)
Interface Stability	Standard: BIND 8.2.4
MT-Level	See resolver(3RESOLV)

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 with 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

`/lib/librpcsvc.so.1`	shared object
`/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 (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_intr_signal_wait_pollfd`
`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... ]
```

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_nanosleep`	`clock_settime`
`close`	`fdatasync`
`fork`	`lio_listio`
`mq_close`	`mq_getattr`
`mq_notify`	`mq_open`
`mq_receive`	`mq_reltimedreceive_np`
`mq_reltimedsend_np`	`mq_send`
`mq_setattr`	`mq_timedreceive`
`mq_timedsend`	`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_reltimedwait_np
sem_timedwait	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_waitn64	aio_write64
lio_listio64	

FILES

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

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

libsasl – Simple Authenti cation and Security Layer library

SYNOPSIS

```
cc [ flag... ] file... -lsasl [ library... ]
#include <sasl/sasl.h>
#include <sasl/prop.h>
#include sasl/saslutil.h
```

DESCRIPTION

SASL is a security framework used by connection-oriented network applications primarily for authentication. Another way to describe SASL is that it is a glue layer between a network application and some security mechanisms that allow applications to authenticate each other and provide additional security services such as data encryption. As a glue layer, SASL hides the interface specifics of the security mechanism from the application, which allows greater portability and flexibility as new security mechanisms are implemented. SASL is similar to the GSS-API in that it provides a layer of abstraction between an application and one or more security mechanisms.

`libsasl` provides both an API for applications and an SPI for various plug-ins. To link with this library, specify `-lsasl` on the `cc` command line.

INTERFACES

The shared object `libsasl.so.1` and associated include files provide the public interfaces defined below. The `*_t` interfaces are function prototypes for callbacks that are defined in the public SASL header files. While `libsasl` provides default versions for some of the callbacks, this structure allows an application to define its own version of the some of the callback functions.

See intro(3) for additional information on shared object interfaces.

`prop_clear`	`prop_dispose`
`prop_dup`	`prop_erase`
`prop_format`	`prop_get`
`prop_getnames`	`prop_new`
`prop_request`	`prop_set`
`prop_setvals`	`sasl_authorize_t`
`sasl_auxprop`	`sasl_auxprop_add_plugin`
`sasl_auxprop_getctx`	`sasl_auxprop_request`
`sasl_canon_user_t`	`sasl_canonuser_add_plugin`
`sasl_chalprompt_t`	`sasl_checkapop`
`sasl_checkpass`	`sasl_client_add_plugin`
`sasl_client_init`	`sasl_client_new`
`sasl_client_plug_init_t`	`sasl_client_start`

sasl_client_step	sasl_decode
sasl_decode64	sasl_dispose
sasl_done	sasl_encode
sasl_encode64	sasl_encodev
sasl_erasebuffer	sasl_errdetail
sasl_errors	sasl_errstring
sasl_getcallback_t	sasl_getopt_t
sasl_getpath_t	sasl_getprop
sasl_getrealm_t	sasl_getsecret_t
sasl_getsimple_t	sasl_global_listmech
sasl_idle	sasl_listmech
sasl_log_t	sasl_server_add_plugin
sasl_server_init	sasl_server_new
sasl_server_plug_init_t	sasl_server_start
sasl_server_step	sasl_server_userdb_checkpass_t
sasl_server_userdb_setpass_t	sasl_set_alloc
sasl_set_mutex	sasl_seterror
sasl_setpass	sasl_setprop
sasl_utf8verify	sasl_verifyfile_t
sasl_version	

FILES

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

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWlibsasl
Interface Stability	Evolving

SEE ALSO

intro(3), attributes(5),

NAME

libscf – service configuration facility library

SYNOPSIS

```
cc [ flag... ] file... -lscf [ library... ]
#include <libscf.h>
```

DESCRIPTION

Functions in this library define the interface for reading, writing, and manipulating service configurations.

INTERFACES

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

scf_entry_add_value	scf_entry_create
scf_entry_destroy	scf_entry_destroy_children
scf_entry_handle	scf_entry_reset
scf_error	scf_handle_bind
scf_handle_create	scf_handle_decode_fmri
scf_handle_decorate	scf_handle_destroy
scf_handle_get_scope	scf_handle_unbind
scf_instance_add_pg	scf_instance_create
scf_instance_delete	scf_instance_destroy
scf_instance_get_name	scf_instance_get_parent
scf_instance_get_pg	scf_instance_get_pg_composed
scf_instance_get_snapshot	scf_instance_handle
scf_instance_to_fmri	scf_iter_create
scf_iter_destroy	scf_iter_handle
scf_iter_handle_scopes	scf_iter_instance_pgs
scf_iter_instance_pgs_composed	scf_iter_instance_pgs_typed_composed
scf_iter_instance_pgs_typed	scf_iter_instance_snapshots
scf_iter_next_instance	scf_iter_next_pg
scf_iter_next_property	scf_iter_next_scope
scf_iter_next_service	scf_iter_next_snapshot
scf_iter_next_value	scf_iter_pg_properties
scf_iter_property_values	scf_iter_reset
scf_iter_scope_services	scf_iter_service_instances

scf_iter_service_pgs	scf_iter_service_pgs_typed
scf_iter_snaplevel_pgs	scf_iter_snaplevel_pgs_typed
scf_limit	scf_myname
scf_pg_create	scf_pg_delete
scf_pg_destroy	scf_pg_get_flags
scf_pg_get_name	scf_pg_get_parent_instance
scf_pg_get_parent_service	scf_pg_get_parent_snaplevel
scf_pg_get_property	scf_pg_get_type
scf_pg_get_underlying_pg	scf_pg_handle
scf_pg_to_fmri	scf_pg_update
scf_property_create	scf_property_destroy
scf_property_get_name	scf_property_get_value
scf_property_handle	scf_property_is_type
scf_property_to_fmri	scf_property_type
scf_scope_add_service	scf_scope_create
scf_scope_destroy	scf_scope_get_name
scf_scope_get_service	scf_scope_handle
scf_scope_to_fmri	scf_service_add_instance
scf_service_add_pg	scf_service_create
scf_service_delete	scf_service_destroy
scf_service_get_instance	scf_service_get_name
scf_service_get_parent	scf_service_get_pg
scf_service_handle	scf_service_to_fmri
scf_simple_app_props_free	scf_simple_app_props_get
scf_simple_app_props_next	scf_simple_app_props_search
scf_simple_prop_free	scf_simple_prop_get
scf_simple_prop_name	scf_simple_prop_next_astring
scf_simple_prop_next_boolean	scf_simple_prop_next_count
scf_simple_prop_next_integer	scf_simple_prop_next_opaque
scf_simple_prop_next_reset	scf_simple_prop_next_time

scf_simple_prop_next_ustring	scf_simple_prop_numvalues
scf_simple_prop_pgname	scf_simple_prop_type
scf_simple_walk_instances	scf_snaplevel_create
scf_snaplevel_destroy	
scf_snaplevel_get_instance_name	scf_snaplevel_get_next_snaplevel
scf_snaplevel_get_parent	scf_snaplevel_get_pg
scf_snaplevel_get_scope_name	scf_snaplevel_get_service_name
scf_snaplevel_handle	scf_snapshot_create
scf_snapshot_destroy	scf_snapshot_get_base_snaplevel
scf_snapshot_get_name	scf_snapshot_get_parent
scf_snapshot_handle	scf_strerror
scf_transaction_add	scf_transaction_commit
scf_transaction_create	scf_transaction_destroy
scf_transaction_destroy_children	scf_transaction_handle
scf_transaction_property_change	scf_transaction_property_change_type
scf_transaction_property_delete	scf_transaction_property_new
scf_transaction_reset	scf_transaction_reset_all
scf_transaction_start	scf_type_base_type
scf_value_base_type	scf_value_create
scf_value_destroy	scf_value_get_as_string
scf_value_get_as_string_typed	scf_value_get_astring
scf_value_get_boolean	scf_value_get_count
scf_value_get_integer	scf_value_get_opaque
scf_value_get_time	scf_value_get_ustring
scf_value_handle	scf_value_is_type
scf_value_reset	scf_value_set_astring
scf_value_set_boolean	scf_value_set_count
scf_value_set_from_string	scf_value_set_integer
scf_value_set_opaque	scf_value_set_time
scf_value_set_ustring	scf_value_type

`smf_degrade_instance`	`smf_disable_instance`
`smf_enable_instance`	`smf_get_state`
`smf_maintain_instance`	`smf_refresh_instance`
`smf_restart_instance`	`smf_restore_instance`

FILES

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

ATTRIBUTES

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

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

SEE ALSO

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

NAME

libsctp – SCTP sockets library

SYNOPSIS

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

DESCRIPTION

Functions in this library provide the SCTP socket interface.

INTERFACES

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

`sctp_bindx`	`sctp_freeladdrs`
`sctp_freepaddrs`	`sctp_getladdrs`
`sctp_getpaddrs`	`sctp_opt_info`
`sctp_peeloff`	`sctp_recvmsg`
`sctp_send`	`sctp_sendmsg`

FILES

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

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	Safe

SEE ALSO

`intro`(2), `intro`(3), `attributes`(5), `sctp`(7P)

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

`/lib/libsec.so.1`	shared object
`/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 (32–bit) SUNWcslx (64–bit)
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

`/lib/libsecdb.so.1`	shared object
`/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) SUNWcslx (64–bit)

ATTRIBUTE TYPE	ATTRIBUTE VALUE
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

`/lib/libsendfile.so.1`	shared object
`/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

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

libsmartcard – smartcard library

SYNOPSIS

```
cc [ flag... ] file... -lsmartcard [ library...]
#include <smartcard/scf.h>
```

DESCRIPTION

Functions in this library allow an application to select a smartcard terminal, determine when cards are inserted or removed, and exchange data with the card.

INTERFACES

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

`SCF_Card_close`	`SCF_Card_exchangeAPDU`
`SCF_Card_freeInfo`	`SCF_Card_getInfo`
`SCF_Card_lock`	`SCF_Card_reset`
`SCF_Card_unlock`	`SCF_Card_waitForCardRemoved`
`SCF_Session_close`	`SCF_Session_freeInfo`
`SCF_Session_getInfo`	`SCF_Session_getSession`
`SCF_Session_getTerminal`	`SCF_Terminal_addEventListener`
`SCF_Terminal_close`	`SCF_Terminal_freeInfo`
`SCF_Terminal_getCard`	`SCF_Terminal_getInfo`
`SCF_Terminal_removeEventListener`	`SCF_Terminal_updateEventListener`
`SCF_Terminal_waitForCardAbsent`	`SCF_Terminal_waitForCardPresent`
`SCF_strerror`	

FILES

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

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWocf
Availability	SUNWocf (32–bit) SUNWocfx (64–bit)
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO

`smartcard`(1M), `intro(3)`, `attributes`(5), `smartcard`(5)

NAME

libsocket – sockets library

SYNOPSIS

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

DESCRIPTION

Functions in this library 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`	`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

/lib/libsocket.so.1	shared object
/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 (32–bit) SUNWcslx (64–bit)
MT-Level	See the manual page for each interface.

SEE ALSO

pvs(1), intro(2), intro(3), socket.h(3HEAD), 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 x86 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

libtecla – interactive command line input library

SYNOPSIS

```
cc [ flag... ] file... -ltecla [ library... ]
#include <libtecla.h>
```

DESCRIPTION

This library provides programs with interactive command line editing facilities, similar to those of the UNIX `tcsh` shell. In addition to simple command-line editing, it supports recall of previously entered command lines, TAB completion of file names or other tokens, and in-line wildcard expansion of filenames. The internal functions that perform file-name completion and wild-card expansion are also available externally for optional use by the calling program.

Thread Safety

The `terminfo` functions `setupterm`(3CURSES), `tigetstr`(3CURSES), `tigetnum`(3CURSES), and `tputs`(3CURSES) are not reentrant. This condition, however, should not prevent use of this library in threaded applications, since few applications will want to interact with multiple terminals.

INTERFACES

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

`cfc_file_start`	`cfc_literal_escapes`
`cfc_set_check_fn`	`cpl_add_completion`
`cpl_check_exe`	`cpl_complete_word`
`cpl_file_completions`	`cpl_last_error`
`cpl_list_completions`	`cpl_recall_matches`
`cpl_record_error`	`del_CplFileConf`
`del_ExpandFile`	`del_GetLine`
`del_PathCache`	`del_PcaPathConf`
`del_WordCompletion`	`ef_expand_file`
`ef_last_error`	`ef_list_expansions`
`gl_abandon_line`	`gl_append_history`
`gl_automatic_history`	`gl_bind_keyseq`
`gl_catch_blocked`	`gl_change_terminal`
`gl_clear_history`	`gl_completion_action`
`gl_configure_getline`	`gl_customize_completion`
`gl_display_text`	`gl_echo_mode`
`gl_erase_terminal`	`gl_error_message`
`gl_get_line`	`gl_group_history`

gl_handle_signal	gl_ignore_signal
gl_inactivity_timeout	gl_io_mode
gl_last_signal	gl_limit_history
gl_list_signals	gl_load_history
gl_lookup_history	gl_normal_io
gl_pending_io	gl_prompt_style
gl_query_char	gl_range_of_history
gl_raw_io	gl_read_char
gl_register_action	gl_replace_prompt
gl_resize_history	gl_return_status
gl_save_history	gl_set_term_size
gl_show_history	gl_size_of_history
gl_state_of_history	gl_terminal_size
gl_toggle_history	gl_trap_signal
gl_tty_signals	gl_watch_fd
libtecla_version	new_CplFileConf
new_ExpandFile	new_GetLine
new_PathCache	new_PcaPathConf
new_WordCompletion	pca_last_error
pca_lookup_file	pca_path_completions
pca_scan_path	pca_set_check_fn
ppc_file_start	ppc_literal_escapes

FILES

/usr/lib/libtecla.so.1	shared object
/usr/lib/64/libtecla.so.1	64–bit shared object

ATTRIBUTES

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

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

SEE ALSO | enhance(1), intro(3), cpl_complete_word(3TECLA), ef_expand_file(3TECLA), gl_get_line(3TECLA), gl_io_mode(3TECLA), pca_lookup_file(3TECLA), attributes(5), tecla(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.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

Historically, functions in `libthread` provided threading support. This functionality now resides in `libc`(3LIB).

This library is maintained to provide backward compatibility for both runtime and compilation environments. The shared object is implemented as a filter on `libc.so.1`. New application development need not specify `-lthread`.

INTERFACES

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

`cond_broadcast`	`cond_destroy`
`cond_init`	`cond_reltimedwait`
`cond_signal`	`cond_timedwait`
`cond_wait`	`mutex_destroy`
`mutex_init`	`mutex_lock`
`mutex_trylock`	`mutex_unlock`
`rw_rdlock`	`rw_tryrdlock`
`rw_trywrlock`	`rw_unlock`
`rw_wrlock`	`rwlock_destroy`
`rwlock_init`	`sema_destroy`
`sema_init`	`sema_post`
`sema_trywait`	`sema_wait`
`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

FILES

/lib/libthread.so.1	a filter on libc.so.1
/lib/64/libthread.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 (32–bit) SUNWcslx (64–bit)
MT-Level	Safe

SEE ALSO

pvs(1), intro(2), intro(3), libc(3LIB), libc_db(3LIB), libpthread(3LIB), attributes(5), threads(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)
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/ucblib -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.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

libumem – object-caching memory allocation library

SYNOPSIS

```
cc [ flag... ] file... -lumem [ library... ]
#include <umem.h>
```

DESCRIPTION

Functions in this library provide fast, scalable object-caching memory allocation with multithreaded application support. In addition to the standard `malloc`(3C) family of functions and the more flexible `umem_alloc`(3MALLOC) family, `libumem` provides powerful object-caching services as described in `umem_cache_create`(3MALLOC).

The `libumem` library also provides extensive debugging support, including detection of memory leaks, buffer overruns, multiple frees, use of uninitialized data, use of freed data, and many other common programming errors. See `umem_debug`(3MALLOC).

INTERFACES

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

`calloc`	`free`
`malloc`	`memalign`
`realloc`	`umem_alloc`
`umem_cache_alloc`	`umem_cache_create`
`umem_cache_destroy`	`umem_cache_free`
`umem_free`	`umem_nofail_callback`
`umem_zalloc`	`valloc`

FILES

`/usr/lib/libumem.so.1`	shared object
`/usr/lib/64/libumem.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

`intro(3)`, `malloc`(3C), `umem_alloc`(3MALLOC), `umem_cache_create`(3MALLOC), `umem_debug`(3MALLOC), `attributes`(5)

NAME

libusb – user-space USB device management library

SYNOPSIS

```
cc [ flag... ] -I/usr/sfw/include file... -L/usr/sfw/lib  -R /usr/sfw/lib \
 -lusb   [ library... ]
#include <usb.h>
```

DESCRIPTION

The `libusb` library contains interfaces for managing USB devices without a kernel driver. It is an open-source API supported on Linux, MacOS X, and NetBSD. See http://libusb.sourceforge.net.

The current implementation is version 0.1.8 of the `libusb` API.

Complete documentation for this library can be found at `/usr/sfw/share/doc/libusb/libusb.txt`.

INTERFACES

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

`usb_bulk_read`	`usb_bulk_write`
`usb_claim_interface`	`usb_clear_halt`
`usb_close`	`usb_control_msg`
`usb_find_busses`	`usb_find_devices`
`usb_get_busses`	`usb_get_descriptor_by_endpoint`
`usb_get_descriptor`	`usb_get_string`
`usb_get_string_simple`	`usb_init`
`usb_interrupt_read`	`usb_interrupt_write`
`usb_open`	`usb_release_interface`
`usb_reset`	`usb_resetep`
`usb_set_altinterface`	`usb_set_configuration`
`usb_set_debug`	`usb_strerror`

FILES

`/usr/sfw/lib/libusb.so.1`	shared object
`/usr/sfw/lib/libusb_plugins`	implementation-specific `libusb` modules
`/usr/sfw/bin/libusb-config`	script to determine linking environment

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWlibusb, SUNWlibusbut, SUNWlibugenusb

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	External
MT-Level	Unsafe

SEE ALSO intro(3), attributes(5)

http://libusb.sourceforge.net

NAME

libuuid – UUID library

SYNOPSIS

```
cc [ flag... ] file... -luuid [ library... ]
#include <uuid/uuid.h>
```

DESCRIPTION

The functions in this library perfrom operations on a universally unique identifier (UUID).

INTERFACES

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

`uuid_clear`	`uuid_compare`
`uuid_copy`	`uuid_generate`
`uuid_generate_random`	`uuid_generate_time`
`uuid_is_null`	`uuid_parse`
`uuid_time`	`uuid_unparse`

FILES

`/lib/libuuid.so.1`	shared object
`/lib/64/libuuid.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

`intro`(3), `uuid_clear`(3UUID), `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.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 (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 is implemented as a filter on `libc.so.1`. New application development need not specify `-lw`.

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

/lib/libw.so.1	a filter on libc.so.1
/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 (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

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	freeaddrinfo
gai_strerror	getaddrinfo
gethostbyaddr	gethostbyname
gethostent	gethostname
getnameinfo	getnetbyaddr
gethostname	getnetbyaddr
getnetbyname	getnetent
getpeername	getprotobyname
getprotobynumber	getprotoent
getservbyname	getservbyport
getservent	getsockname
getsockopt	h_errno
htonl	htons
if_freenameindex	if_indextoname
if_nameindex	if_nametoindex
inet_addr	inet_lnaof
inet_makeaddr	inet_netof
inet_network	inet_ntoa
inet_ntop	inet_pton
listen	ntohl
ntohs	recv
recvfrom	recvmsg
send	sendmsg
sendto	sethostent
setnetent	setprotoent
setservent	setsockopt
shutdown	sockatmark
socket	socketpair

`t_errno`

FILES

`/lib/libxnet.so.1`	shared object
`/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)
Interface Stability	Standard
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.

`main`	`yyerror`

FILES

`/usr/lib/liby.so.1`	shared object
`/usr/lib/64/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

limits.h, limits – implementation-defined constants

SYNOPSIS

```
#include <limits.h>
```

DESCRIPTION

The <limits.h> header defines various symbolic names. Different categories of names are described below.

The names represent various limits on resources that the implementation imposes on applications. Symbolic constant names beginning with _POSIX can be found in unistd.h(3HEAD).

Applications should not assume any particular value for a limit. An application wishing to avail itself of the full amount of a resource available on an implementation can make use of the value given in limits.h on that particular implementation by using the symbolic names listed below. Many of the listed limits are not invariant, and at runtime, the value of the limit might differ from those given in this header, for the following reasons:

- The limit is pathname-dependent.
- The limit differs between the compile and runtime machines.

For these reasons, an application can use the fpathconf(2), pathconf(2), and sysconf(3C) functions to determine the actual value of a limit at runtime.

Runtime Invariant Values (Possibly Indeterminate)

AIO_LISTIO_MAX
: Maximum number of I/O operations in a single list I/O call supported by the implementation.

AIO_MAX
: Maximum number of outstanding asynchronous I/O operations supported by the implementation.

AIO_PRIO_DELTA_MAX
: The maximum amount by which a process can decrease its asynchronous I/O priority level from its own scheduling priority.

ARG_MAX
: Maximum length of argument to the exec(2) functions including environment data.

ATEXIT_MAX
: Maximum number of functions that can be registered with atexit(3C).

CHILD_MAX
: Maximum number of simultaneous processes per real user ID.

CLK_TCK
: Number of clock ticks per second returned by the times(2) function.

DELAYTIMER_MAX
: Maximum number of timer expiration overruns.

HOST_NAME_MAX
: Maximum length of a host name (not including the terminating null) as returned from the gethostname(3C) function.

`IOV_MAX`
Maximum number of `iovec` structures that one process has available for use with `readv`(2) or `writev`(2).

`LOGIN_NAME_MAX`
Maximum length of a login name.

`MQ_OPEN_MAX`
The maximum number of open message queue descriptors a process is allowed to hold.

`LOGIN_NAME_MAX`
Maximum length of a login name.

`MQ_OPEN_MAX`
The maximum number of open message queue descriptors a process is allowed to hold.

`MQ_PRIO_MAX`
The maximum number of message priorities supported by the implementation.

`OPEN_MAX`
Maximum number of files that one process can have open at any one time.

`PAGESIZE`
Size in bytes of a page.

`PAGE_SIZE`
Equivalent to `PAGESIZE`. If either `PAGESIZE` or `PAGE_SIZE` is defined, the other is defined with the same value.

`PASS_MAX`
The maximum number of significant bytes in a password, not including the terminating null.

`PTHREAD_DESTRUCTOR_ITERATIONS`
Maximum number of attempts made to destroy a thread's thread-specific data values on thread exit.

`PTHREAD_KEYS_MAX`
Maximum number of data keys that can be created by a process.

`PTHREAD_STACK_MIN`
Minimum size in bytes of thread stack storage.

`PTHREAD_THREADS_MAX`
Maximum number of threads that can be created per process.

`RE_DUP_MAX`
The number of repeated occurrences of a BRE permitted by the `regexec`(3C) and `regcomp`(3C) functions when using the interval notation {\(*m*,*n*\}.

`RTSIG_MAX`
Maximum number of realtime signals reserved for application use in this implementation.

SEM_NSEMS_MAX
Maximum number of semaphores that a process can have.

SEM_VALUE_MAX
The maximum value a semaphore can have.

SIGQUEUE_MAX
Maximum number of queued signals that a process can send and have pending at the receiver(s) at any time.

SS_REPL_MAX
The maximum number of replenishment operations that may be simultaneously pending for a particular sporadic server scheduler.

STREAM_MAX
The number of streams that one process can have open at one time. If defined, it has the same value as FOPEN_MAX.

SYMLOOP_MAX
Maximum number of symbolic links that can be reliably traversed in the resolution of a pathname in the absence of a loop.

TIMER_MAX
Maximum number of timers per process supported by the implementation.

TRACE_EVENT_NAME_MAX
Maximum length of the trace event name.

TRACE_NAME_MAX
Maximum length of the trace generation version string or of the trace stream name.

TRACE_SYS_MAX
Maximum number of trace streams that may simultaneously exist in the system.

TRACE_USER_EVENT_MAX
Maximum number of user trace event type identifiers that may simultaneously exist in a traced process, including the predefined user trace event POSIX_TRACE_UNNAMED_USER_EVENT.

TTY_NAME_MAX
Maximum length of terminal device name.

TZNAME_MAX
Maximum number of bytes supported for the name of a timezone (not of the TZ variable).

Pathname Variable Values

The values in the following list can be constants within an implementation or can vary from one pathname to another. For example, file systems or directories can have different characteristics. The value supported for a specific pathname is provided by the pathconf(2) function.

FILESIZEBITS
Minimum number of bits needed to represent, as a signed integer value, the maximum size of a regular file allowed in the specified directory.

`LINK_MAX`
Maximum number of links to a single file.

`MAX_CANON`
Maximum number of bytes in a terminal canonical input line.

`MAX_INPUT`
Minimum number of bytes for which space is available in a terminal input queue; therefore, the maximum number of bytes a conforming application may require to be typed as input before reading them.

`NAME_MAX`
Maximum number of bytes in a filename (not including terminating null).

`PATH_MAX`
Maximum number of bytes in a pathname, including the terminating null character.

`PIPE_BUF`
Maximum number of bytes that is guaranteed to be atomic when writing to a pipe.

`POSIX_ALLOC_SIZE_MIN`
Minimum number of bytes of storage actually allocated for any portion of a file.

`POSIX_REC_INCR_XFER_SIZE`
Recommended increment for file transfer sizes between the `POSIX_REC_MIN_XFER_SIZE` and `POSIX_REC_MAX_XFER_SIZE` values.

`POSIX_REC_MAX_XFER_SIZE`
Maximum recommended file transfer size.

`POSIX_REC_MIN_XFER_SIZE`
Minimum recommended file transfer size.

`POSIX_REC_XFER_ALIGN`
Recommended file transfer buffer alignment.

`SYMLINK_MAX`
Maximum number of bytes in a symbolic link.

Runtime Increasable Values

The magnitude limitations in the following list are fixed by specific implementations. An application should assume that the value supplied by `<limits.h>` in a specific implementation is the minimum that pertains whenever the application is run under that implementation. A specific instance of a specific implementation can increase the value relative to that supplied by `<limits.h>` for that implementation. The actual value supported by a specific instance is provided by the `sysconf`(3C) function.

`BC_BASE_MAX`
Maximum obase values allowed by the `bc`(1) utility.

`BC_DIM_MAX`
Maximum number of elements permitted in an array by the `bc` utility.

`BC_SCALE_MAX`
Maximum scale value allowed by the `bc` utility.

BC_STRING_MAX
Maximum length of a string constant accepted by the bc utility.

CHARCLASS_NAME_MAX
Maximum number of bytes in a character class name.

COLL_WEIGHTS_MAX
Maximum number of weights that can be assigned to an entry of the LC_COLLATE order keyword in the locale definition file.

EXPR_NEST_MAX
Maximum number of expressions that can be nested within parentheses by the expr(1) utility.

EXPR_NEST_MAX
Maximum number of expressions that can be nested within parentheses by the expr utility.

LINE_MAX
Unless otherwise noted, the maximum length, in bytes, of a utility's input line (either standard input or another file), when the utility is described as processing text files. The length includes room for the trailing <newline>.

NGROUPS_MAX
Maximum number of simultaneous supplementary group IDs per process.

RE_DUP_MAX
Maximum number of repeated occurrences of a regular expression permitted when using the interval notation \{*m*,*n*\}.

Maximum Values

The symbolic constants in the following list are symbolic names for the most restrictive value for certain features on an implementation supporting the POSIX Timers option.

_POSIX_CLOCKRES_MIN
The resolution of the CLOCK_REALTIME clock, in nanoseconds.

Minimum Values

The symbolic constants in the following list are symbolic names for the most restrictive value for certain features on an implementation conforming to various POSIX and Single Unix Specification requirements. See standards(5).

_POSIX_AIO_LISTIO_MAX
The number of I/O operations that can be specified in a list I/O call.

_POSIX_AIO_MAX
The number of outstanding asynchronous I/O operations.

_POSIX_ARG_MAX
Maximum length of argument to the exec(2) functions including environment data.

_POSIX_CHILD_MAX
Maximum number of simultaneous processes per real user ID.

_POSIX_DELAYTIMER_MAX
The number of timer expiration overruns.

_POSIX_HOST_NAME_MAX
Maximum length of a host name (not including the terminating null) as returned from the gethostname(3C) function.

_POSIX_LINK_MAX
Maximum number of links to a single file.

_POSIX_LOGIN_NAME_MAX
The size of the storage required for a login name, in bytes, including the terminating null.

_POSIX_MAX_CANON
Maximum number of bytes in a terminal canonical input queue.

_POSIX_MAX_INPUT
Maximum number of bytes allowed in a terminal input queue.

_POSIX_MQ_OPEN_MAX
The number of message queues that can be open for a single process.

_POSIX_MQ_PRIO_MAX
The maximum number of message priorities supported by the implementation.

_POSIX_NAME_MAX
Maximum number of bytes in a filename (not including terminating null).

_POSIX_NGROUPS_MAX
Maximum number of simultaneous supplementary group IDs per process.

_POSIX_OPEN_MAX
Maximum number of files that one process can have open at any one time.

_POSIX_PATH_MAX
Maximum number of bytes in a pathname.

_POSIX_PIPE_BUF
Maximum number of bytes that is guaranteed to be atomic when writing to a pipe.

_POSIX_RE_DUP_MAX
The number of repeated occurrences of a BRE permitted by the regexec() and regcomp() functions when using the interval notation {\(*m*,*n*\}

_POSIX_RTSIG_MAX
The number of realtime signal numbers reserved for application use.

_POSIX_SEM_NSEMS_MAX
The number of semaphores that a process can have.

_POSIX_SEM_VALUE_MAX
The maximum value a semaphore can have.

_POSIX_SIGQUEUE_MAX
The number of queued signals that a process can send and have pending at the receiver(s) at any time.

_POSIX_SSIZE_MAX
The value that can be stored in an object of type ssize_t.

_POSIX_STREAM_MAX
The number of streams that one process can have open at one time.

_POSIX_SS_REPL_MAX
The number of replenishment operations that can be simultaneously pending for a particular sporadic server scheduler.

_POSIX_SYMLINK_MAX
The number of bytes in a symbolic link.

_POSIX_SYMLOOP_MAX
The number of symbolic links that can be traversed in the resolution of a pathname in the absence of a loop.

_POSIX_THREAD_DESTRUCTOR_ITERATIONS
The number of attempts made to destroy a thread's thread-specific data values on thread exit.

_POSIX_THREAD_KEYS_MAX
The number of data keys per process.

_POSIX_THREAD_THREADS_MAX
The number of threads per process.

_POSIX_TIMER_MAX
The per-process number of timers.

_POSIX_TRACE_EVENT_NAME_MAX
The length in bytes of a trace event name.

_POSIX_TRACE_NAME_MAX
The length in bytes of a trace generation version string or a trace stream name.

_POSIX_TRACE_SYS_MAX
The number of trace streams that can simultaneously exist in the system.

_POSIX_TRACE_USER_EVENT_MAX
The number of user trace event type identifiers that may simultaneously exist in a traced process, including the predefined user trace event POSIX_TRACE_UNNAMED_USER_EVENT.

_POSIX_TTY_NAME_MAX
The size of the storage required for a terminal device name, in bytes, including the terminating null.

_POSIX_TZNAME_MAX
Maximum number of bytes supported for the name of a timezone (not of the TZ variable).

_POSIX2_BC_BASE_MAX
Maximum obase values allowed by the bc utility.

_POSIX2_BC_DIM_MAX
Maximum number of elements permitted in an array by the bc utility.

_POSIX2_BC_SCALE_MAX
Maximum scale value allowed by the bc utility.

_POSIX2_BC_STRING_MAX
Maximum length of a string constant accepted by the bc utility.

_POSIX2_CHARCLASS_NAME_MAX
Maximum number of bytes in a character class name.

_POSIX2_COLL_WEIGHTS_MAX
Maximum number of weights that can be assigned to an entry of the LC_COLLATE order keyword in the locale definition file.

_POSIX2_EXPR_NEST_MAX
Maximum number of expressions that can be nested within parentheses by the expr utility.

_POSIX2_LINE_MAX
Unless otherwise noted, the maximum length, in bytes, of a utility's input line (either standard input or another file), when the utility is described as processing text files. The length includes room for the trailing <newline>.

_POSIX2_RE_DUP_MAX
Maximum number of repeated occurrences of a regular expression permitted when using the interval notation \{*m*,*n*\}.

_XOPEN_IOV_MAX
Maximum number of iovec structures that one process has available for use with readv(2) or writev(2).

_XOPEN_NAME_MAX
Maximum number of bytes in a filename (not including the terminating null).

_XOPEN_PATH_MAX
Maximum number of bytes in a pathname.

Numerical Limits

The values in the following lists shall be defined in <limits.h> and are constant expressions suitable for use in #if preprocessing directives. Moreover, except for CHAR_BIT, DBL_DIG, DBL_MAX, FLT_DIG, FLT_MAX, LONG_BIT, WORD_BIT, and MB_LEN_MAX, the symbolic names are defined as expressions of the correct type.

If the value of an object of type char is treated as a signed integer when used in an expression, the value of CHAR_MIN is the same as that of SCHAR_MIN and the value of CHAR_MAX is the same as that of SCHAR_MAX. Otherwise, the value of CHAR_MIN is 0 and the value of CHAR_MAX is the same as that of UCHAR_MAX.

CHAR_BIT
Number of bits in a type char.

CHAR_MAX
Maximum value of type char.

`CHAR_MIN`
Minimum value of type `char`.

`DBL_DIG`
Digits of precision of type `double`.

`DBL_MAX`
Maximum decimal value of a `double`.

`DBL_MIN`
Minimum decimal value of a `double`.

`FLT_DIG`
Digits of precision of type `float`.

`FLT_MAX`
Maximum decimal value of a `float`.

`FLT_MIN`
Minimum decimal value of a `float`.

`INT_MIN`
Minimum value of type `int`.

`INT_MAX`
Maximum value of an `int`.

`LLONG_MIN`
Minimum value of type `long long`.

`LLONG_MAX`
Maximum value of type `long long`.

`LONG_BIT`
Number of bits in a `long`.

`LONG_MIN`
Minimum value of type `long`.

`LONG_MAX`
Maximum value of a `long`.

`MB_LEN_MAX`
Maximum number of bytes in a character, for any supported locale.

`SCHAR_MIN`
Minimum value of type `signed char`.

`SCHAR_MAX`
Maximum value of type `signed char`.

`SHRT_MIN`
Minimum value of type `short`.

`SHRT_MAX`
Maximum value of type `short`.

SSIZE_MAX
Maximum value of an object of type ssize_t.

TMP_MAX
Minimum number of unique filename generated by tmpnam(3C). Maximum number of times an application can call tmpnam() reliably.

UCHAR_MAX
Maximum value of type unsigned char.

UINT_MAX
Maximum value of type unsigned.

ULLONG_MAX
Maximum value of type unsigned long long.

ULONG_MAX
Maximum value of type unsigned long.

USHRT_MAX
Maximum value for a type unsigned short.

WORD_BIT
Number of bits in a word or type int.

Other Invariant Values

The following constants are defined in <limits.h>.

CHARCLASS_NAME_MAX
Maximum number of bytes in a character class name.

LOGNAME_MAX
The maximum number of bytes supported in a user's login name.

NL_ARGMAX
Maximum value of digit in calls to the printf(3C) and scanf(3C) functions.

NL_LANGMAX
Maximum number of bytes in a LANG name.

NL_MSGMAX
Maximum message number.

NL_NMAX
Maximum number of bytes in an N-to-1 collation mapping.

NL_SETMAX
Maximum set number.

NL_TEXTMAX
Maximum number of bytes in a message string.

NZERO
Default process priority.

SEE ALSO

fpathconf(2), pathconf(2), sysconf(3C), standards(5)

NAME

locale.h, locale – category macros

SYNOPSIS

```
#include <locale.h>
```

DESCRIPTION

The <locale.h> header provides a definition for the lconv structure, which includes the following members. (See the definition of LC_MONETARY in locale(5).)

```
char        *currency_symbol
char        *decimal_point
char        frac_digits
char        *grouping
char        *int_curr_symbol
char        int_frac_digits
char        int_n_cs_precedes
char        int_n_sep_by_space
char        int_n_sign_posn
char        int_p_cs_precedes
char        int_p_sep_by_space
char        int_p_sign_posn
char        *mon_decimal_point
char        *mon_grouping
char        *mon_thousands_sep
char        *negative_sign
char        n_cs_precedes
char        n_sep_by_space
char        n_sign_posn
char        *positive_sign
char        p_cs_precedes
char        p_sep_by_space
char        p_sign_posn
char        *thousands_sep
```

The <locale.h> header defines NULL (as defined in <stddef.h>) and the following as macros:

```
LC_ALL
LC_COLLATE
LC_CTYPE
LC_MESSAGES
LC_MONETARY
LC_NUMERIC
LC_TIME
```

The preceding expand to distinct integer constant expressions, for use as the first argument to the setlocale() function. See setlocale(3C).

Additional macro definitions, beginning with the characters LC_ and an uppercase letter, can also be specified here.

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Standard

SEE ALSO | `setlocale`(3C), `localeconv`(3C), `stddef.h`(3HEAD), `attributes`(5), `locale`(5), `standards`(5)

NAME

math.h, math – mathematical declarations

SYNOPSIS

```
#include <math.h>
```

DESCRIPTION

The <math.h> header includes definitions for the following types:

float_t	A real-floating type at least as wide as float.
double_t	A real-floating type at least as wide as double, and at least as wide as float_t.

If FLT_EVAL_METHOD equals 0, float_t and double_t are float and double, respectively. If FLT_EVAL_METHOD equals 1, they are both double. If FLT_EVAL_METHOD equals 2, they are both be long double. Other values of FLT_EVAL_METHOD are implementation-defined.

The <math.h> header provides the following constants. The values are of type double and are accurate within the precision of the double type.

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	π, the ratio of the circumference of a circle to its diameter.
M_PI_2	π/2.
M_PI_4	π/4.
M_1_PI	1/π.
M_2_PI	2/π.
M_2_SQRTPI	2 over the square root of π.
M_SQRT2	The positive square root of 2.
M_SQRT1_2	The positive square root of 1/2.

The <math.h> header defines the following symbolic constants:

MAXFLOAT	The maximum value of a non-infinite single-precision floating point number.
HUGE_VAL	A positive double expression, not necessarily representable as a float. Used as an error value returned by the mathematics library. HUGE_VAL evaluates to +infinity on systems supporting IEEE Std 754-1985.

HUGE_VALF	A positive float constant expression. Used as an error value returned by the mathematics library. HUGE_VALF evaluates to +infinity on systems supporting IEEE Std 754-1985.
HUGE_VALL	A positive long double constant expression. Used as an error value returned by the mathematics library. HUGE_VALL evaluates to +infinity on systems supporting IEEE Std 754-1985.
INFINITY	A constant expression of type float representing positive or unsigned infinity, if available; else a positive constant of type float that overflows at translation time.
NAN	A constant expression of type float representing a quiet NaN. This symbolic constant is only defined if the implementation supports quiet NaNs for the float type.

The following macros are defined for number classification. They represent the mutually-exclusive kinds of floating-point values. They expand to integer constant expressions with distinct values

```
FP_INFINITE
FP_NAN
FP_NORMAL
FP_SUBNORMAL
FP_ZERO
```

The following optional macros indicate whether the fma() family of functions are fast compared with direct code:

```
FP_FAST_FMA
FP_FAST_FMAF
FP_FAST_FMAL
```

The FP_FAST_FMA macro is defined to indicate that the fma() function generally executes about as fast as, or faster than, a multiply and an add of double operands. The other macros have the equivalent meaning for the float and long double versions.

The following macros expand to integer constant expressions whose values are returned by ilogb(*x*) if *x* is zero or NaN, respectively. The value of FP_ILOGB0 is either {INT_MIN} or -{INT_MAX}. The value of FP_ILOGBNAN is either {INT_MAX} or {INT_MIN}.

```
FP_ILOGB0
FP_ILOGBNAN
```

The following macros expand to the integer constants 1 and 2, respectively:

```
MATH_ERRNO
MATH_ERREXCEPT
```

The following macro expands to an expression that has type int and the value MATH_ERREXCEPT:

`math_errhandling`

The value of the macro `math_errhandling` is constant for the duration of the program. If a macro definition is suppressed or a program defines an identifier with the name `math_errhandling`, the behavior is undefined.

The `<math.h>` header defines he following external variable:

```
extern int signgam;
```

The `<math.h>` header defines the structure and constants used by the `matherr`(3M) error-handling mechanisms.

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Standard

SEE ALSO

`intro`(3), `fenv.h`(3HEAD), `libm`(3LIB), `limits.h`(3HEAD), `matherr`(3M), `attributes`(5), `standards`(5)

NAME

mman.h, mman – memory management declarations

SYNOPSIS

```
#include <sys/mman.h>
```

DESCRIPTION

The `<sys/mman.h>` header supports the following options:

- the Memory Mapped Files option
- the Shared Memory Objects option
- the Process Memory Locking option
- the Memory Protection option
- the Synchronized Input and Output option

For Memory Mapped Files and Shared Memory Objects options, the following protection options are defined:

`PROT_READ`	Page can be read.
`PROT_WRITE`	Page can be written.
`PROT_EXEC`	Page can be executed.
`PROT_NONE`	Page cannot be accessed.

The following *flag* options are defined:

`MAP_SHARED`	Share changes.
`MAP_PRIVATE`	Changes are private.
`MAP_FIXED`	Interpret `addr` exactly.

The flags immediately following are defined for `msync()`. See `msync`(3C).

`MS_ASYNC`	Perform asynchronous writes.
`MS_SYNC`	Perform synchronous writes.
`MS_INVALIDATE`	Invalidate mappings.

The symbolic constants immediately following are defined for the `mlockall()` function. See `mlockall`(3C).

`MCL_CURRENT`	Lock currently mapped pages.
`MCL_FUTURE`	Lock pages that become mapped.

The symbolic constant `MAP_FAILED` is defined to indicate a failure from the `mmap()` function. See `mmap`(2).

The `mode_t`, `off_t`, and `size_t` types are be defined as described in `<sys/types.h>`. See types(3HEAD).

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Standard

SEE ALSO

`mmap`(2), `mprotect`(2), `munmap`(2), `madvise`(3C), `mlock`(3C), `mlockall`(3C), `msync`(3C), `shm_open`(3RT), `shm_unlink`(3RT), `attributes`(5), `standards`(5)

NAME

monetary.h, monetary – monetary types

SYNOPSIS

```
#include <monetary.h>
```

DESCRIPTION

The <monetary.h> header defines the following types:

`size_t` As described in stddef.h(3HEAD).

`ssize_t` As described in types.h(3HEAD).

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Standard

SEE ALSO

stddef.h(3HEAD), strfmon(3C), types.h(3HEAD), attributes(5), standards(5)

NAME

mqueue.h, 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.h(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>.

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Standard

SEE ALSO

fcntl.h(3HEAD), signal.h(3HEAD), time.h(3HEAD), types.h(3HEAD), attributes(5), standards(5)

NAME

msg.h, msg – message queue structures

SYNOPSIS

```
#include <sys/msg.h>
```

DESCRIPTION

The <sys/msg.h> header defines the following data types through typedef:

msgqnum_t used for the number of messages in the message queue

msglen_t used for the number of bytes allowed in the message queue

These types are unsigned integer types that are able to store values at least as large as a type unsigned short.

The <sys/msg.h> header defines the following constant as a message operation flag:

MSG_NOERROR no error if big message

The msqid_ds structure contains the following members:

```
struct ipc_perm    msg_perm      Operation permission structure.
msgqnum_t          msg_qnum      Number of messages currently on queue.
msglen_t           msg_qbytes    Maximum number of bytes allowed on queue.
pid_t              msg_lspid     Process ID of last msgsnd(2).
pid_t              msg_lrpid     Process ID of last msgrcv(2).
time_t             msg_stime     Time of last msgsnd().
time_t             msg_rtime     Time of last msgrcv().
time_t             msg_ctime     Time of last change.
```

The pid_t, time_t, key_t, size_t, and ssize_t types are defined as described in <sys/types.h>. See types(3HEAD).

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Standard

SEE ALSO

msgctl(2), msgget(2), msgrcv(2), msgsnd(2), ipc.h(3HEAD), types.h(3HEAD), attributes(5), standards(5)

NAME

ndbm.h, 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.

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Standard

SEE ALSO

ndbm(3C), attributes(5), standards(5)

NAME

netdb.h, 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.h(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     /* pointer to an array of pointers to alternative */
                     /* host names, terminated by a null pointer */
int  h_addrtype      /* address type */
int  h_length        /* length, in bytes, of the address */
char **h_addr_list   /* 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     /* 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     /* pointer to an array of pointers to alternative */
                     /* protocol names, terminated by a null pointer */
int  p_proto         /* protocol number */
```

The <netdb.h> header defines the servent structure that includes the following members:

```
char *s_name         /* fficial name of the service */
char **s_aliases     /* pointer to an array of pointers to alternative */
                     /* service names, terminated by a null pointer */
int  s_port          /* port number at which the service resides, */
                     /* in network byte order */
char *s_proto        /* 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.h(3HEAD).

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Standard

SEE ALSO

intro(3), endhostent(3NSL), endhostent(3XNET), endnetent(3SOCKET), endnetent(3XNET), endprotoent(3SOCKET), endprotoent(3XNET), endservent(3SOCKET), endservent(3XNET), in.h(3HEAD), attributes(5), standards(5)

NAME

nl_types.h, 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.h`(3HEAD)

NAME

poll.h, poll – definitions for the `poll()` function

SYNOPSIS

```
#include <poll.h>
```

DESCRIPTION

The `<poll.h>` header defines the `pollfd` structure, which includes the following members:

`int fd`	the following descriptor being polled
`short events`	the input event flags (see below)
`short revents`	the output event flags (see below)

The `<poll.h>` header defines the following type through `typedef`:

`nfds_t`	an unsigned integer type used for the number of file descriptors

The implementation supports one or more programming environments in which the width of `nfds_t` is no greater than the width of type long. The names of these programming environments can be obtained using the `confstr()` function or the `getconf` utility. See `confstr`(3C) and `getconf`(1).

The following symbolic constants are defined, zero or more of which can be OR'ed together to form the `events` or `revents` members in the `pollfd` structure:

`POLLIN`	Data other than high-priority data can be read without blocking.
`POLLRDNORM`	Normal data can be read without blocking.
`POLLRDBAND`	Priority data can be read without blocking.
`POLLPRI`	High priority data can be read without blocking.
`POLLOUT`	Normal data can be written without blocking.
`POLLWRNORM`	Equivalent to `POLLOUT`.
`POLLWRBAND`	Priority data can be written.
`POLLERR`	An error has occurred (`revents` only).
`POLLHUP`	Device has been disconnected (`revents` only).
`POLLNVAL`	Invalid `fd` member (`revents` only).

The significance and semantics of normal, priority, and high-priority data are file and device-specific.

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Standard

SEE ALSO

`getconf`(1), `poll`(2), `confstr`(3C), `attributes`(5), `standards`(5)

NAME

pthread.h, pthread – threads

SYNOPSIS

```
#include <pthread.h>
```

DESCRIPTION

The `<pthread.h>` header defines the following symbols:

```
PTHREAD_BARRIER_SERIAL_THREAD
PTHREAD_CANCEL_ASYNCHRONOUS
PTHREAD_CANCEL_ENABLE
PTHREAD_CANCEL_DEFERRED
PTHREAD_CANCEL_DISABLE
PTHREAD_CANCELED
PTHREAD_COND_INITIALIZER
PTHREAD_CREATE_DETACHED
PTHREAD_CREATE_JOINABLE
PTHREAD_EXPLICIT_SCHED
PTHREAD_INHERIT_SCHED
PTHREAD_MUTEX_DEFAULT
PTHREAD_MUTEX_ERRORCHECK
PTHREAD_MUTEX_INITIALIZER
PTHREAD_MUTEX_NORMAL
PTHREAD_MUTEX_RECURSIVE
PTHREAD_ONCE_INIT
PTHREAD_PRIO_INHERIT
PTHREAD_PRIO_NONE
PTHREAD_PRIO_PROTECT
PTHREAD_PROCESS_SHARED
PTHREAD_PROCESS_PRIVATE
PTHREAD_SCOPE_PROCESS
PTHREAD_SCOPE_SYSTEM
```

The types listed below are defined as described in `<sys/types.h>`. See `types.h`(3HEAD).

```
pthread_attr_t
pthread_barrier_t
pthread_barrierattr_t
pthread_cond_t
pthread_condattr_t
pthread_key_t
pthread_mutex_t
pthread_mutexattr_t
pthread_once_t
pthread_rwlock_t
pthread_rwlockattr_t
pthread_spinlock_t
pthread_t
```

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Standard

SEE ALSO

`sched.h`(3HEAD), `time.h`(3HEAD), `types.h`(3HEAD), `pthread_attr_getguardsize`(3C), `pthread_attr_init`(3C), `pthread_attr_setscope`(3C), `pthread_cancel`(3C),

pthread_cleanup_pop(3C), pthread_cond_init(3C), pthread_cond_signal(3C), pthread_cond_wait(3C), pthread_condattr_init(3C), pthread_create(3C), pthread_detach(3C), pthread_equal(3C), pthread_exit(3C), pthread_getconcurrency(3C), pthread_getschedparam(3C), pthread_join(3C), pthread_key_create(3C), pthread_key_delete(3C), pthread_mutex_init(3C), pthread_mutex_lock(3C), pthread_mutex_setprioceiling(3C), pthread_mutexattr_init(3C), pthread_mutexattr_init(3C), pthread_mutexattr_init(3C), pthread_mutexattr_gettype(3C), pthread_mutexattr_getprotocol(3C), pthread_once(3C), pthread_rwlock_init(3C), pthread_rwlock_rdlock(3C), pthread_rwlock_unlock(3C), pthread_rwlock_wrlock(3C), pthread_rwlockattr_getpshared(3C), pthread_rwlockattr_init(3C), pthread_self(3C), pthread_setcancelstate(3C), pthread_setspecific(3C), attributes(5), standards(5)

NAME

pwd.h, pwd – password structure

SYNOPSIS

```
#include <pwd.h>
```

DESCRIPTION

The `<pwd.h>` header provides a definition for `struct passwd`, which includes the following members:

```
char  *pw_name     user's login name
uid_t pw_uid       numerical user ID
gid_t pw_gid       numerical group ID
char  *pw_dir      initial working di rectory
char  *pw_shell    program to use as shell
```

The `gid_t` and `uid_t` types are defined as described in `<sys/types.h>`. See `types.h`(3HEAD).

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Standard

SEE ALSO

`getpwnam`(3C), `types.h`(3HEAD), `attributes`(5), `standards`(5)

NAME

regex.h, regex – regular expression matching types

SYNOPSIS

```
#include <regex.h>
```

DESCRIPTION

The `<regex.h>` header defines the structures and symbolic constants used by the `regcomp()`, `regexec()`, `regerror()`, and `regfree()` functions. See `regcomp`(3C).

The structure type `regex_t` contains the following member:

```
size_t re_nsub     number of parenthesized subexpressions
```

The type `size_t` is defined as described in `<sys/types.h>`. See `types.h`(3HEAD).

The type `regoff_t` is defined as a signed integer type that can hold the largest value that can be stored in either a type `off_t` or type `ssize_t`. The structure type `regmatch_t` contains the following members:

```
regoff_t rm_so     byte offset from start of string to start of substring
regoff_t rm_eo     byte offset from start of string of the first character after the end
                   of substring
```

Values for the *cflags* parameter to the `regcomp` function are as follows:

`REG_EXTENDED`	use extended regular expressions
`REG_ICASE`	ignore case in match
`REG_NOSUB`	report only success or fail in `regexec()`
`REG_NEWLINE`	change the handling of `NEWLINE` character

Values for the *eflags* parameter to the `regexec()` function are as follows:

`REG_NOTBOL`	The circumflex character (^), when taken as a special character, does not match the beginning of string.
`REG_NOTEOL`	The dollar sign ($), when taken as a special character, does not match the end of string.

The following constants are defined as error return values:

`REG_NOMATCH`	`regexec()` failed to match.
`REG_BADPAT`	Invalid regular expression.
`REG_ECOLLATE`	Invalid collating element referenced.
`REG_ECTYPE`	Invalid character class type referenced.
`REG_EESCAPE`	Trailing ’\’ in pattern.
`REG_ESUBREG`	Number in `\\`*digit* invalid or in error.
`REG_EBRACK`	“]&]” imbalance.
`REG_EPAREN`	“\(\)” or “()” imbalance.

REG_EBRACE	"\\{\\}" imbalance.
REG_BADBR	Content of "\\{\\}" invalid: not a number, number too large, more than two numbers, first larger than second.
REG_ERANGE	Invalid endpoint in range expression.
REG_ESPACE	Out of memory.
REG_BADRPT	'?', '*', or '+' not preceded by valid regular expression.
REG_ENOSYS	Reserved.

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Standard

SEE ALSO

regcomp(3C), types.h(3HEAD), attributes(5), standards(5)

NAME

resource.h, resource – definitions for resource operations

SYNOPSIS

```
#include <sys/resource.h>
```

DESCRIPTION

The <sys/resource.h> header defines the symbolic constants listed below as possible values of the *which* argument of getpriority() and setpriority(). See getpriority(3C).

PRIO_PROCESS	identifies the *who* argument as a process ID
PRIO_PGRP	identifies the *who* argument as a process group ID
PRIO_USER	identifies the *who* argument as a user ID

The following type is defined through typedef:

rlim_t	unsigned integer type used for limit values

The following symbolic constants are defined:

RLIM_INFINITY	a value of rlim_t indicating no limit
RLIM_SAVED_MAX	a value of type rlim_t indicating an unrepresentable saved hard limit
RLIM_SAVED_CUR	a value of type rlim_t indicating an unrepresentable saved soft limit

The symbolic constants listed below are defined as possible values of the *who* parameter of getrusage(). See getrusage(3C).

RUSAGE_SELF	returns information about the current process
RUSAGE_CHILDREN	returns information about children of the current process

The <sys/resource.h> header defines the rlimit structure, which includes the following members:

```
rlim_t rlim_cur     /* the current (soft) limit */
rlim_t rlim_max     /* the hard limit */
```

The <sys/resource.h> header defines the rusage structure, which includes the following members:

```
struct timeval ru_utime     /* user time used */
struct timeval ru_stime     /* system time used */
```

The timeval structure is defined as described in <sys/time.h>.

The symbolic constants listed below are defined as possible values for the *resource* argument of getrlimit() and setrlimit(). See getrlimit(2).

RLIMIT_CORE	limit on size of core dump file
RLIMIT_CPU	limit on CPU time per process

RLIMIT_DATA	limit on data segment size
RLIMIT_FSIZE	limit on file size
RLIMIT_NOFILE	limit on number of open files
RLIMIT_STACK	limit on stack size
RLIMIT_AS	limit on address space size

The id_t type is defined through typedef as described in <sys/types.h>. See types.h(3HEAD).

Inclusion of the <sys/resource.h> header can also make visible all symbols from <sys/time.h>. See time.h(3HEAD).

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Standard

SEE ALSO

getrlimit(2), getpriority(3C), time.h(3HEAD), types.h(3HEAD), attributes(5), standards(5)

NAME

sched.h, 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.h`(3HEAD)

NAME

search.h, search – search tables

SYNOPSIS

```
#include <search.h>
```

DESCRIPTION

The `<search.h>` header defines the `ENTRY` type for structure `entry`, which includes the following members:

```
char *key
void *data
```

and defines `ACTION` and `VISIT` as enumeration data types through type definitions as follows:

```
enum { FIND, ENTER } ACTION;
enum { preorder, postorder, endorder, leaf } VISIT;
```

The `size_t` type is defined as described in `<sys/types.h>`. See `types.h`(3HEAD).

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Standard

SEE ALSO

`hsearch`(3C), `insque`(3C), `lsearch`(3C), `tsearch`(3C), `types.h`(3HEAD), `attributes`(5), `standards`(5)

NAME

select.h, select – select types

SYNOPSIS

```
#include <sys/select.h>
```

DESCRIPTION

The <sys/select.h> header defines the timeval structure, which includes the following members:

```
time_t      tv_sec      /* seconds */
suseconds_t tv_usec     /* microseconds */
```

The time_t and suseconds_t types are defined as described in <sys/types.h>. See types.h(3HEAD).

The sigset_t type is defined as described in signal.h(3HEAD).

The timespec structure is defined as described in <time.h>. See time.h(3HEAD).

The <sys/select.h> header defines the fd_set type as a structure.

The following is defined as a macro:

FD_SETSIZE Maximum number of file descriptors in an fd_set structure.

Inclusion of the <sys/select.h> header can make visible all symbols from the headers <signal.h>, <sys/time.h>, and <time.h>.

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Standard

SEE ALSO

select(3C), signal.h(3HEAD), time.h(3HEAD), types.h(3HEAD), attributes(5), standards(5)

NAME

semaphore.h, semaphore – semaphores

SYNOPSIS

```
#include <semaphore.h>
```

DESCRIPTION

The `<semaphore.h>` header defines the `sem_t` type, used in performing semaphore operations. The semaphore can be implemented using a file descriptor, in which case applications are able to open up at least a total of {`OPEN_MAX`} files and semaphores. The symbol `SEM_FAILED` is defined (see `sem_open`(3RT)).

Inclusion of the `<semaphore.h>` header can make visible symbols defined in the headers `<fcntl.h>` and `<sys/types.h>`. See `fcntl.h`(3HEAD) and `types.h`(3HEAD).

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Standard

SEE ALSO

`fcntl.h`(3HEAD), `types.h`(3HEAD), `sem_destroy`(3RT), `sem_getvalue`(3RT), `sem_init`(3RT), `sem_open`(3RT), `sem_post`(3RT), `sem_timedwait`(3RT), `sem_unlink`(3RT), `sem_wait`(3RT), `attributes`(5), `standards`(5)

NAME

sem.h, sem – semaphore facility

SYNOPSIS

```
#include <sys/sem.h>
```

DESCRIPTION

The <sys/sem.h> header defines the following constants and structures.

Semaphore operation flags:

SEM_UNDO Set up adjust on exit entry.

Command definitions for the semctl() function are provided as listed below. See semctl(2).

GETNCNT Get semncnt.

GETPID Get sempid.

GETVAL Get semval.

GETALL Get all cases of semval.

GETZCNT Get semzcnt.

SETVAL Set semval.

SETALL Set all cases of semval.

The semid_ds structure contains the following members:

```
struct ipc_perm sem_perm     /* operation permission structure */
unsigned short  sem_nsems    /* number of semaphores in set */
time_t          sem_otime    /* last semop() time */
time_t          sem_ctime    /* last time changed by semctl() */
```

The pid_t, time_t, key_t, and size_t types are defined as described in <sys/types.h>. See types.h(3HEAD).

A semaphore is represented by an anonymous structure containing the following members:

```
unsigned short semval     /* semaphore value */
pid_t          sempid     /* process ID of last operation */
unsigned short semncnt    /* number of processes waiting for semval
                          /* to become greater than current value */
unsigned short semzcnt    /* number of processes waiting for semval to become 0 */
```

The sembuf structure contains the following members:

```
unsigned short sem_num     /* semaphore number */
short          sem_op      /* semaphore operation */
short          sem_flg     /* operation flags */
```

All of the symbols from <sys/ipc.h> are defined when this header is included. See ipc.h(3HEAD).

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Standard

SEE ALSO

`semctl`(2), `semget`(2), `semop`(2), `ipc.h`(3HEAD), `types.h`(3HEAD), `attributes`(5), `standards`(5)

NAME

setjmp.h, setjmp – stack environment declarations

SYNOPSIS

```
#include <setjmp.h>
```

DESCRIPTION

The `<setjmp.h>` header defines the array types `jmp_buf` and `sigjmp_buf`. Applications must define the appropriate feature test macro to enable the visibility of the symbols in this header.

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Standard

SEE ALSO

`_longjmp`(3C), `setjmp`(3C), `attributes`(5), `standards`(5)

NAME

shm.h, shm – shared memory facility

SYNOPSIS

```
#include <sys/shm.h>
```

DESCRIPTION

The <sys/shm.h> header defines the following symbolic constants:

SHM_RDONLY	attach read-only (else read-write)
SHM_RND	round attach address to SHMLBA

The <sys/shm.h> header defines the following symbolic value:

SHMLBA	segment low boundary address multiple

The following data types are defined through typedef:

shmatt_t	Unsigned integer used for the number of current attaches that must be able to store values at least as large as a type unsigned short.

The shmid_ds structure contains the following members:

```
struct ipc_perm shm_perm       /* operation permission structure */
size_t          shm_segsz      /* size of segment in bytes */
pid_t           shm_lpid       /* process ID of last shared memory operation */
pid_t           shm_cpid       /* process ID of creator */
shmatt_t        shm_nattch     /* number of current attaches */
time_t          shm_atime      /* time of last shmat() */
time_t          shm_dtime      /* time of last shmdt() */
time_t          shm_ctime      /* time of last change by shmctl() */
```

The pid_t, time_t, key_t, and size_t types are defined as described in <sys/types.h>. See types.h(3HEAD).

In addition, all of the symbols from <sys/ipc.h> are defined when this header is included.

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Standard

SEE ALSO

shmctl(2), shmget(2), shmop(2), ipc.h(3HEAD), types.h(3HEAD), attributes(5), standards(5)

NAME

siginfo.h, 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:

```
pid_t        si_pid      /* sending process ID */
uid_t        si_uid      /* sending user ID */
ctid_t       si_ctid     /* sending contract ID */
zoneid_t     si_zoneid   /* sending zone ID */S
```

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.h`(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.h, 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.h`(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. See `thr_create`(3C). Each process can specify a system action to be taken in response to each signal sent to it, called the signal's disposition. All threads 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 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 uses `sigwait`(2) to create a thread to handle the receipt of the signal. The `sigaction()`

function can be used for both synchronously and asynchronously generated signals. The `sigwait()` function will work only for asynchronously generated signals, as synchronously generated signals are sent to the thread that caused the event. The `sigwait()` function is the recommended for use with a multithreaded application.

SIGNAL MASK

Each thread has a signal mask that defines the set of signals currently blocked from delivery to it. The signal mask of the main thread is inherited from the signal mask of the thread that created it in the parent process. The selection of the thread 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 in the receiving process. Signals that are generated by action of a particular thread such as hardware faults are delivered to the thread that caused the signal. See `thr_sigsetmask`(3C) or `sigprocmask`(2). See `alarm`(2) for current semantics of delivery of `SIGALRM`. Signals that are directed to a particular thread are delivered to the targeted thread. See `thr_kill`(3C). If the selected thread has blocked the signal, it remains pending on the thread 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) one of the threads that does not have the signal blocked is selected to process the signal. If all the threads within the process block the signal, it remains pending on the process until a thread 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 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
`SIGKILL`	9	Exit	Killed
`SIGBUS`	10	Core	Bus Error

Name	Value	Default	Event
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	Reserved for threading support
SIGLWP	33	Ignore	Reserved for threading support
SIGFREEZE	34	Ignore	Check point Freeze
SIGTHAW	35	Ignore	Check point Thaw
SIGCANCEL	36	Ignore	Reserved for threading support
SIGLOST	37	Exit	Resource lost (for example, record–lock lost)
SIGXRES	38	Ignore	Resource control exceeded (see setrctl(2))
SIGJVM1	39	Ignore	Reserved for Java Virtual Machine 1

Name	Value	Default	Event
SIGJVM2	40	Ignore	Reserved for Java Virtual Machine 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 can 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 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 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(2) function, however, additional arguments can be requested. 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 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`. If the `SA_RESTART` flag is set, however, certain function calls 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.

SIGEV_PORT — An asynchronous notification is delivered to an event port when the event of interest occurs. The sival_ptr member points to a port_notify_t structure (see port_associate(3C)). The event port identifier as well as an application-defined cookie are part of the port_notify_t structure.

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.

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Standard

SEE ALSO

lockd(1M), intro(2), alarm(2), exit(2), fcntl(2), getrlimit(2), ioctl(2), kill(2), pause(2), setrctl(2), sigaction(2), sigaltstack(2), sigprocmask(2), sigsend(2), sigsuspend(2), sigwait(2), port_associate(3C), setjmp(3C), siginfo.h(3HEAD), signal(3C), sigqueue(3RT), sigsetops(3C), thr_create(3C), thr_kill(3C), thr_sigsetmask(3C), ucontext.h(3HEAD), wait(3C), attributes(5), standards(5)

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.h, 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.

`SCM_UCRED` Indicates that the data array contains a `ucred_t` to be received. The `ucred_t` is the credential of the sending process at the time the message was sent. This is a Sun-specific, Evolving interface. See `ucred_get`(3C).

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.

`CMSG_SPACE(`*len*`)`
Given the length of an ancillary data object, `CMSG_SPACE()` returns the space required by the object and its `cmsghdr` structure, including any padding needed to satisfy alignment requirements. This macro can be used, for example, to allocate space dynamically for the ancillary data. This macro should not be used to initialize the `cmsg_len` member of a `cmsghdr` structure. Use the `CMSG_LEN()` macro instead.

CMSG_LEN(*len*)
Given the length of an ancillary data object, CMSG_LEN() returns the value to store in the cmsg_len member of the cmsghdr structure, taking into account any padding needed to satisfy alignment requirements.

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.
SO_RECVUCRED	Request the reception of user credential ancillary data. This is a Sun-specific, Evolving interface. See ucred_get(3C).

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.

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Standard

SEE ALSO

accept(3SOCKET), accept(3XNET), bind(3SOCKET), bind(3XNET), connect(3SOCKET), connect(3XNET), getpeername(3SOCKET), getpeername(3XNET), getpeerucred(3C), getsockname(3SOCKET), getsockname(3XNET), getsockopt(3SOCKET), getsockopt(3XNET), libsocket(3LIB), 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), ucred_get(3C)attributes(5), standards(5)

NAME

spawn.h, spawn – spawn

SYNOPSIS

```
#include <spawn.h>
```

DESCRIPTION

The `<spawn.h>` header defines the `posix_spawnattr_t` and `posix_spawn_file_actions_t` types used in performing spawn operations.

The `<spawn.h>` header defines the flags that can be set in a `posix_spawnattr_t` object using the `posix_spawnattr_setflags()` function:

```
POSIX_SPAWN_RESETIDS
POSIX_SPAWN_SETPGROUP
POSIX_SPAWN_SETSCHEDPARAM
POSIX_SPAWN_SETSCHEDULER
POSIX_SPAWN_SETSIGDEF
POSIX_SPAWN_SETSIGMASK
```

Inclusion of the `<spawn.h>` header can make visible symbols defined in the `<sched.h>`, `<signal.h>`, and `<sys/types.h>` headers.

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Standard

SEE ALSO

`sched.h`(3HEAD), `semaphore.h`(3HEAD), `signal.h`(3HEAD), `types.h`(3HEAD), `attributes`(5), `standards`(5)

NAME

stat.h, 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
```

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Standard

SEE ALSO

`stat`(2), `types.h`(3HEAD), `attributes`(5), `standards`(5)

NAME

statvfs.h, statvfs – VFS File System information structure

SYNOPSIS

```
#include <sys/statvfs.h>
```

DESCRIPTION

The <sys/statvfs.h> header defines the statvfs structure, which includes the following members:

```
unsigned long f_bsize     /* file system block size */
unsigned long f_frsize    /* fundamental file system block size */
fsblkcnt_t    f_blocks    /* total number of blocks on file system */
                          /* in units of f_frsize */
fsblkcnt_t    f_bfree     /* total number of free blocks */
fsblkcnt_t    f_bavail    /* number of free blocks available to */
                          /* non-privileged process */
fsfilcnt_t    f_files     /* total number of file serial numbers */
fsfilcnt_t    f_ffree     /* total number of free file serial numbers */
fsfilcnt_t    f_favail    /* number of file serial numbers available */
                          /* to non-privileged process */
unsigned long f_fsid      /* file system ID */
unsigned long f_flag      /* bit mask of f_flag values */
unsigned long f_namemax   /* maximum filename length */
```

The fsblkcnt_t and fsfilcnt_t types are defined as described in <sys/types.h>. See types.h(3HEAD).

The following flags for the f_flag member are defined:

ST_RDONLY — read-only file system

ST_NOSUID — does not support setuid()/setgid() semantics

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Standard

SEE ALSO

statvfs(2), types.h(3HEAD), attributes(5), standards(5)

NAME

stdbool.h, stdbool – boolean type and values

SYNOPSIS

```
#include <stdbool.h>
```

DESCRIPTION

The `<stdbool.h>` header defines the following macros:

`bool`	expands to `_Bool`
`true`	expands to the integer constant 1
`false`	expands to the integer constant 0
`__bool_true_false_are_defined`	expands to the integer constant 1

An application can undefine and then possibly redefine the macros `bool`, `true`, and `false`.

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Standard

SEE ALSO

`attributes`(5), `standards`(5)

NAME

stddef.h, stddef – standard type definitions

SYNOPSIS

```
#include <stddef.h>
```

DESCRIPTION

The `<stddef.h>` header defines the following macros:

`NULL`
Null pointer constant.

`offsetof(type, member-designator)`
Integer constant expression of type `size_t`, the value of which is the offset in bytes to the structure member (`member-designator`), from the beginning of its structure (`type`).

The `<stddef.h>` header defines the following types:

`ptrdiff_t`	Signed integer type of the result of subtracting two pointers.
`wchar_t`	Integer type whose range of values can represent distinct wide-character codes for all members of the largest character set specified among the locales supported by the compilation environment: the null character has the code value 0 and each member of the portable character set has a code value equal to its value when used as the lone character in an integer character constant.
`size_t`	Unsigned integer type of the result of the `sizeof` operator.

The implementation supports one or more programming environments in which the widths of `ptrdiff_t`, `size_t`, and `wchar_t` are no greater than the width of type `long`. The names of these programming environments can be obtained using the `confstr`(3C) function or the `getconf`(1) utility.

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Standard

SEE ALSO

`getconf`(1), `confstr`(3C), `types.h`(3HEAD), `wchar.h`(3HEAD), `attributes`(5), `standards`(5)

NAME

stdint.h, stdint – integer types

SYNOPSIS

```
#include <stdint.h>
```

DESCRIPTION

The `<stdint.h>` header declares sets of integer types having specified widths, and defines corresponding sets of macros. It also defines macros that specify limits of integer types corresponding to types defined in other standard headers.

The ''width'' of an integer type is the number of bits used to store its value in a pure binary system; the actual type can use more bits than that (for example, a 28-bit type could be stored in 32 bits of actual storage). An N-bit signed type has values in the range -2^{N-1} or $1-2^{N-1}$ to $2^{N-1}-1$, while an N-bit unsigned type has values in the range 0 to $2^{N}-1$.

Types are defined in the following categories:

- integer types having certain exact widths
- integer types having at least certain specified widths
- fastest integer types having at least certain specified widths
- integer types wide enough to hold pointers to objects
- integer types having greatest width

Some of these types may denote the same type.

Corresponding macros specify limits of the declared types and construct suitable constants.

For each type described herein that the implementation provides, the `<stdint.h>` header declares that `typedef` name and defines the associated macros. Conversely, for each type described herein that the implementation does not provide, the `<stdint.h>` header does not declare that `typedef` name, nor does it define the associated macros. An implementation provides those types described as required, but need not provide any of the others (described as optional).

Integer Types

When `typedef` names differing only in the absence or presence of the initial `u` are defined, they denote corresponding signed and unsigned types as described in the ISO/IEC 9899: 1999 standard, Section 6.2.5; an implementation providing one of these corresponding types must also provide the other.

In the following descriptions, the symbol *N* represents an unsigned decimal integer with no leading zeros (for example, 8 or 24, but not 04 or 048).

Exact-width integer types

The `typedef` name `int`*N*`_t` designates a signed integer type with width *N*, no padding bits, and a two's-complement representation. Thus, `int8_t` denotes a signed integer type with a width of exactly 8 bits.

The typedef name `uint`*N*`_t` designates an unsigned integer type with width *N*. Thus, `uint24_t` denotes an unsigned integer type with a width of exactly 24 bits.

The following types are required:

```
int8_t
int16_t
int32_t
uint8_t
uint16_t
uint32_t
```

If an implementation provides integer types with width 64 that meet these requirements, then the following types are required:

```
int64_t
uint64_t
```

In particular, this is the case if any of the following are true:

- The implementation supports the `_POSIX_V6_ILP32_OFFBIG` programming environment and the application is being built in the `_POSIX_V6_ILP32_OFFBIG` programming environment (see the Shell and Utilities volume of IEEE Std 1003.1-200x, c99, Programming Environments).
- The implementation supports the `_POSIX_V6_LP64_OFF64` programming environment and the application is being built in the `_POSIX_V6_LP64_OFF64` programming environment.
- The implementation supports the `_POSIX_V6_LPBIG_OFFBIG` programming environment and the application is being built in the `_POSIX_V6_LPBIG_OFFBIG` programming environment.

All other types of this form are optional.

Minimum-width integer types

The `typedef` name `int_least`*N*`_t` designates a signed integer type with a width of at least *N*, such that no signed integer type with lesser size has at least the specified width. Thus, `int_least32 _t` denotes a signed integer type with a width of at least 32 bits.

The `typedef` name `uint_least`*N*`_t` designates an unsigned integer type with a width of at least *N*, such that no unsigned integer type with lesser size has at least the specified width. Thus, `uint_ least16_t` denotes an unsigned integer type with a width of at least 16 bits.

The following types are required:

```
int_least8_t
int_least16_t
int_least32_t
int_least64_t
uint_least8_t
uint_least16_t
uint_least32_t
uint_least64_t
```

All other types of this form are optional.

Fastest minimum-width integer types

Each of the following types designates an integer type that is usually fastest to operate with among all integer types that have at least the specified width.

The designated type is not guaranteed to be fastest for all purposes; if the implementation has no clear grounds for choosing one type over another, it will simply pick some integer type satisfying the signedness and width requirements.

The `typedef` name `int_fast`*N*`_t` designates the fastest signed integer type with a width of at least *N*. The `typedef` name `uint_fast`*N*`_t` designates the fastest unsigned integer type with a width of at least *N*.

The following types are required:

```
int_fast8_t
int_fast16_t
int_fast32_t
int_fast64_t
uint_fast8_t
uint_fast16_t
uint_fast32_t
uint_fast64_t
```

All other types of this form are optional.

Integer types capable of holding object pointers

`intptr_t` Designates a signed integer type with the property that any valid pointer to void can be converted to this type, then converted back to a pointer to void, and the result will compare equal to the original pointer.

`uintptr_t` Designates an unsigned integer type with the property that any valid pointer to void can be converted to this type, then converted back to a pointer to void, and the result will compare equal to the original pointer.

On standard-conforming systems, the `intptr_t` and `uintptr_t` types are required; otherwise, they are optional.

Greatest-width integer types

`intmax_t` Designates a signed integer type capable of representing any value of any signed integer type.

`uintmax_t` Designates an unsigned integer type capable of representing any value of any unsigned integer type.

These types are required.

Applications can test for optional types by using the corresponding limit macro from `Limits of Specified-Width Integer Types`.

Limits of Specified-Width Integer Types

The following macros specify the minimum and maximum limits of the types declared in the `<stdint.h>` header. Each macro name corresponds to a similar type name in `Integer Types`.

Each instance of any defined macro is replaced by a constant expression suitable for use in `#if` preprocessing directives. This expression has the same type as would an expression that is an object of the corresponding type converted according to the integer promotions. Its implementation-defined value is equal to or greater in magnitude (absolute value) than the corresponding value given below, with the same sign, except where stated to be exactly the given value.

Limits of exact-width integer types

- Minimum values of exact-width signed integer types:

  {`INTN_MIN`} Exactly $-(2^{N-1})$
- Maximum values of exact-width signed integer types:

  {`INTN_MAX`} Exactly $2^{N-1} -1$
- Maximum values of exact-width unsigned integer types:

  {`UINTN_MAX`} Exactly $2^{N} -1$

Limits of minimum-width integer types

- Minimum values of minimum-width signed integer types:

  {`INT_LEAST`*N*`_MIN`} $-(2^{N-1} -1)$
- Maximum values of minimum-width signed integer types:

  {`INT_LEAST`*N*`_MAX`} $2^{N-1} -1$
- Maximum values of minimum-width unsigned integer types:

  {`UINT_LEAST`*N*`_MAX`} $2^{N} -1$

Limits of fastest minimum-width integer types

- Minimum values of fastest minimum-width signed integer types:

  {`INT_FAST`*N*`_MIN`} $-(2^{N-1} -1)$
- Maximum values of fastest minimum-width signed integer types:

  {`INT_FAST`*N*`_MAX`} $2^{N-1} -1$
- Maximum values of fastest minimum-width unsigned integer types:

  {`UINT_FAST`*N*`_MAX`} $2^{N-1} -1$

Limits of integer types capable of holding object pointers

- Minimum value of pointer-holding signed integer type:

  {`INTPTR_MIN`} $-(2^{15} -1)$
- Maximum value of pointer-holding signed integer type:

  {`INTPTR_MAX`} $2^{15} -1$
- Minimum value of pointer-holding signed integer type:

  {`UINTPTR_MAX`} $2^{16} -1$

Limits of greatest-width integer types

- Minimum value of greatest-width signed integer type:

  {INTMAX_MIN} $-(2^{63} -1)$

- Maximum value of greatest-width signed integer type:

  {INTMAX_MIN} $2^{63} -1$

- Maximum value of greatest-width unsigned integer type:

  {UINTMAX_MIN} $2^{64} -1$

Limits of Other Integer Types

The following macros specify the minimum and maximum limits of integer types corresponding to types defined in other standard headers.

Each instance of these macros is replaced by a constant expression suitable for use in `#if` preprocessing directives. This expression has the same type as would an expression that is an object of the corresponding type converted according to the integer promotions. Its implementation-defined value is equal to or greater in magnitude (absolute value) than the corresponding value given below, with the same sign.

Limits of `ptrdiff_t`:

{PTRDIFF_MIN}	-65535
{PTRDIFF_MAX}	+65535

Limits of `sig_atomic_t`:

{SIG_ATOMIC_MIN}	See below.
{SIG_ATOMIC_MAX}	See below.

Limits of `size_t`:

{SIZE_MAX}	65535

Limits of `wchar_t`:

{WCHAR_MIN}	See below.
{WCHAR_MAX}	See below.

Limits of `wint_t`:

{WINT_MIN}	See below.
{WINT_MAX}	See below.

If sig_atomic_t (see the <signal.h> header) is defined as a signed integer type, the value of {SIG_ATOMIC_MIN} is no greater than -127 and the value of {SIG_ATOMIC_MAX} is no less than 127. Otherwise, sig_atomic_t is defined as an unsigned integer type, the value of {SIG_ATOMIC_MIN} is 0, and the value of {SIG_ATOMIC_MAX} is no less than 255.

If wchar_t (see the <stddef.h> header) is defined as a signed integer type, the value of {WCHAR_MIN} is no greater than -127 and the value of {WCHAR_MAX} is no less than 127. Otherwise, wchar_t is defined as an unsigned integer type, and the value of {WCHAR_MIN} is 0 and the value of {WCHAR_MAX} is no less than 255.

If wint_t (see the <wchar.h> header) is defined as a signed integer type, the value of {WINT_MIN} is no greater than -32767 and the value of {WINT_MAX} is no less than 32767. Otherwise, wint_t is defined as an unsigned integer type, and the value of {WINT_MIN} is 0 and the value of {WINT_MAX} is no less than 65535.

Macros for Integer Constant Expressions

The following macros expand to integer constant expressions suitable for initializing objects that have integer types corresponding to types defined in the <stdint.h> header. Each macro name corresponds to a similar type name listed under minimum-width integer types and greatest-width integer types.

Each invocation of one of these macros expands to an integer constant expression suitable for use in #if preprocessing directives. The type of the expression has the same type as would an expression that is an object of the corresponding type converted according to the integer promotions. The value of the expression is that of the argument. The argument in any instance of these macros is a decimal, octal, or hexadecimal constant with a value that does not exceed the limits for the corresponding type.

Macros for minimum-width integer constant expressions

The macro INTN_C(*value*) expands to an integer constant expression corresponding to the type int_least*N*_t. The macro UINTN_C(*value*) expands to an integer constant expression corresponding to the type uint_least*N*_t. For example, if uint_least64_t is a name for the type unsigned long long, then UINT64_C(0x123) might expand to the integer constant 0x123ULL.

Macros for greatest-width integer constant expressions

The following macro expands to an integer constant expression having the value specified by its argument and the type intmax_t:

INTMAX_C(*value*)

The following macro expands to an integer constant expression having the value specified by its argument and the type uintmax_t:

UINTMAX_C(*value*)

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Standard

SEE ALSO

`inttypes.h`(3HEAD), `signal.h`(3HEAD), `stddef.h`(3HEAD), `wchar.h`(3HEAD), `attributes`(5), `standards`(5)

NAME

stdio.h, stdio – standard buffered input/output

SYNOPSIS

```
#include <stdio.h>
```

DESCRIPTION

The <stdio.h> header defines the following macros as positive integer constant expressions:

BUFSIZ	size of <stdio.h> buffers
_IOFBF	input/output fully buffered
_IOLBF	input/output line buffered
_IONBF	input/output unbuffered
L_ctermid	maximum size of character array to hold ctermid() output
L_tmpnam	maximum size of character array to hold tmpnam() output
SEEK_CUR	seek relative to current position
SEEK_END	seek relative to end-of-file
SEEK_SET	seek relative to start-of-file

The following macros are defined as positive integer constant expressions that denote implementation limits:

{FILENAME_MAX}	Maximum size in bytes of the longest filename string that the implementation guarantees can be opened.
{FOPEN_MAX}	Number of streams that the implementation guarantees can be open simultaneously. The value is at least eight.
{TMP_MAX}	Minimum number of unique filenames generated by tmpnam(). Maximum number of times an application can call tmpnam() reliably. The value of {TMP_MAX} is at least 25. On XSI-conformant systems, the value of {TMP_MAX} is at least 10000.

The following macro name is defined as a negative integer constant expression:

EOF	end-of-file return value

The following macro name is defined as a null pointer constant:

NULL	null pointer

The following macro name is defined as a string constant:

P_tmpdir	default directory prefix for tempnam()

The following is defined as expressions of type "pointer to FILE" point to the FILE objects associated, respectively, with the standard error, input, and output streams:

stderr	standard error output stream
stdin	standard input stream

`stdout`	standard output stream

The following data types are defined through `typedef`:

`FILE`	structure containing information about a file
`fpos_t`	non-array type containing all information needed to specify uniquely every position within a file
`va_list`	as described in `<stdarg.h>`
`size_t`	as described in `<stddef.h>`

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Standard

SEE ALSO

`rename`(2), `ctermid`(3C), `fclose`(3C), `fdopen`(3C), `fflush`(3C), `fgetc`(3C), `fgetpos`(3C), `fgets`(3C), `flockfile`(3C), `fopen`(3C), `fputc`(3C), `fputs`(3C), `fputwc`(3C), `fread`(3C), `freopen`(3C), `fseek`(3C), `fsetpos`(3C), `ftell`(3C), `fwrite`(3C), `getwchar`(3C), `getopt`(3C), `perror`(3C), `popen`(3C), `printf`(3C), `remove`(3C), `rewind`(3C), `scanf`(3C), `setbuf`(3C), `stdio`(3C), `system`(3C), `tmpfile`(3C), `tmpnam`(3C), `ungetc`(3C), `vprintf`(3C), `attributes`(5), `standards`(5)

NAME

stdlib.h, stdlib – standard library definitions

SYNOPSIS

```
#include <stdlib.h>
```

DESCRIPTION

The `<stdlib.h>` header defines the following macros:

`EXIT_FAILURE`	Unsuccessful termination for `exit()`; evaluates to a non-zero value. See `exit`(3C).
`EXIT_SUCCESS`	Successful termination for `exit()`; evaluates to 0.
`NULL`	Null pointer.
{`RAND_MAX`}	Maximum value returned by `rand()`; at least 32767. See `rand`(3C).
{`MB_CUR_MAX`}	Integer expression whose value is the maximum number of bytes in a character specified by the current locale.

The following data types are defined through `typedef`:

`div_t`	structure type returned by the `div()` function
`ldiv_t`	structure type returned by the `ldiv()` function
`lldiv_t`	structure type returned by the `lldiv()` function
`size_t`	as described in `<stddef.h>`
`wchar_t`	as described in `<stddef.h>`

See `div`(3C), which covers `div()`, `ldiv()`, and `lldiv()`, and `stddef.h`(3HEAD).

In addition, the symbolic names and macros listed below are defined as in `<sys/wait.h>`, for use in decoding the return value from `system()`. See `wait.h`(3HEAD) and `system`(3C).

```
WNOHANG
WUNTRACED
WEXITSTATUS
WIFEXITED
WIFSIGNALED
WIFSTOPPED
WSTOPSIG
WTERMSIG
```

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Standard

SEE ALSO | a64l(3C), abort(3C), abs(3C), atexit(3C), bsearch(3C), div(3C), drand48(3C), exit(3C), getenv(3C), getsubopt(3C), grantpt(3C), malloc(3C), mblen(3C), mbstowcs(3C), mbtowc(3C), mkstemp(3C), ptsname(3C), putenv(3C), qsort(3C), random(3C), realpath(3C), strtod(3C), strtol(3C), strtoul(3C), unlockpt(3C), wcstombs(3C), wctomb(3C), limits.h(3HEAD), math.h(3HEAD), stddef.h(3HEAD), types.h(3HEAD), wait.h(3HEAD), attributes(5), standards(5)

NAME

string.h, string – string operations

SYNOPSIS

```
#include <string.h>
```

DESCRIPTION

The `<string.h>` header defines the following:

`NULL`	null pointer constant
`size_t`	as described in `<stddef.h>`

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Standard

SEE ALSO

`memory`(3C), `strcoll`(3C), `string`(3C), `strxfrm`(3C), `stddef.h`(3HEAD), `types.h`(3HEAD), `attributes`(5), `standards`(5)

NAME

strings.h, strings – string operations

SYNOPSIS

```
#include <strings.h>
```

DESCRIPTION

The `size_t` type specified in `<strings.h>` is defined through `typedef` as described in `<stddef.h>`.

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Standard

SEE ALSO

`ffs`(3C), `string`(3C), `stddef.h`(3HEAD), `attributes`(5), `standards`(5)

NAME

stropts.h, stropts – STREAMS interface (STREAMS)

SYNOPSIS

```
#include <stropts.h>
```

DESCRIPTION

The <stropts.h> header defines the bandinfo structure, which includes the following members:

```
unsigned char bi_pri     /* priority band */
int           bi_flag    /* flushing type */
```

The <stropts.h> header defines the strpeek structure that includes the following members:

```
struct strbuf ctlbuf     /* control portion of the message */
struct strbuf databuf    /* data portion of the message */
t_uscalar_t   flags      /* RS_HIPRI or 0 */
```

The <stropts.h> header defines the strbuf structure that includes the following members:

```
int  maxlen     /* maximum buffer length */
int  len        /* length of data */
char *buf       /* pointer to buffer */
```

The <stropts.h> header defines the strfdinsert structure that includes the following members:

```
struct strbuf ctlbuf     /* control portion of the message */
struct strbuf databuf    /* data portion of the message */
t_uscalar_t   flags      /* RS_HIPRI or 0 */
int           fildes     /* file descriptor of the other STREAM */
int           offset     /* relative location of the stored value */
```

The <stropts.h> header defines the strioctl structure that includes the following members:

```
int  ic_cmd        /* ioctl() command */
int  ic_timout     /* timeout for response */
int  ic_len        /* length of data */
char *ic_dp        /* pointer to buffer */
```

The <stropts.h> header defines the strrecvfd structure that includes the following members:

```
int   fda     /* received file descriptor */
uid_t uid     /* UID of sender */
gid_t gid     /* GID of sender */
```

The uid_t and gid_t types are defined through typedef as described in <sys/types.h>. See types.h(3HEAD).

The <stropts.h> header defines the t_scalar_t and t_uscalar_t types, respectively, as signed and unsigned opaque types of equal length of at least 32 bits.

The <stropts.h> header defines the str_list structure that includes the following members:

```
int             sl_nmods        /* number of STREAMS module names */
struct str_mlist *sl_modlist    /* STREAMS module names */
```

The <stropts.h> header defines the str_mlist structure that includes the following member:

char l_name[FMNAMESZ+1]	a STREAMS module name

The following macros are defined for use as the request argument to ioctl():

I_PUSH	Push a STREAMS module.
I_POP	Pop a STREAMS module.
I_LOOK	Get the top module name.
I_FLUSH	Flush a STREAM.
I_FLUSHBAND	Flush one band of a STREAM.
I_SETSIG	Ask for notification signals.
I_GETSIG	Retrieve current notification signals.
I_FIND	Look for a STREAMS module.
I_PEEK	Peek at the top message on a STREAM.
I_SRDOPT	Set the read mode.
I_GRDOPT	Get the read mode.
I_NREAD	Size the top message.
I_FDINSERT	Send implementation-defined information about another STREAM.
I_STR	Send a STREAMS ioctl().
I_SWROPT	Set the write mode.
I_GWROPT	Get the write mode.
I_SENDFD	Pass a file descriptor through a STREAMS pipe.
I_RECVFD	Get a file descriptor sent via I_SENDFD.
I_LIST	Get all the module names on a STREAM.
I_ATMARK	Is the top message ‘‘marked’’?
I_CKBAND	See if any messages exist in a band.
I_GETBAND	Get the band of the top message on a STREAM.
I_CANPUT	Is a band writable?
I_SETCLTIME	Set close time delay.
I_GETCLTIME	Get close time delay.

I_LINK	Connect two STREAMs.
I_UNLINK	Disconnect two STREAMs.
I_PLINK	Persistently connect two STREAMs.
I_PUNLINK	Dismantle a persistent STREAMS link.

The following macro is defined for use with I_LOOK:

FMNAMESZ	minimum size in bytes of the buffer referred to by the arg argument

The following macros are defined for use with I_FLUSH:

FLUSHR	flush read queues
FLUSHW	flush write queues
FLUSHRW	flush read and write queues

The following macros are defined for use with I_SETSIG:

S_RDNORM	A normal (priority band set to 0) message has arrived at the head of a STREAM head read queue.
S_RDBAND	A message with a non-zero priority band has arrived at the head of a STREAM head read queue.
S_INPUT	A message, other than a high-priority message, has arrived at the head of a STREAM head read queue.
S_HIPRI	A high-priority message is present on a STREAM head read queue.
S_OUTPUT	The write queue for normal data (priority band 0) just below the STREAM head is no longer full. This notifies the process that there is room on the queue for sending (or writing) normal data downstream.
S_WRNORM	Equivalent to S_OUTPUT.
S_WRBAND	The write queue for a non-zero priority band just below the STREAM head is no longer full.
S_MSG	A STREAMS signal message that contains the SIGPOLL signal reaches the front of the STREAM head read queue.
S_ERROR	Notification of an error condition reaches the STREAM head.
S_HANGUP	Notification of a hangup reaches the STREAM head.
S_BANDURG	When used in conjunction with S_RDBAND, SIGURG is generated instead of SIGPOLL when a priority message reaches the front of the STREAM head read queue.

The following macro is defined for use with I_PEEK:

RS_HIPRI	Only look for high-priority messages.

The following macros are defined for use with I_SRDOPT:

RNORM	Byte-STREAM mode, the default.
RMSGD	Message-discard mode.
RMSGN	Message-non-discard mode.
RPROTNORM	Fail read() with [EBADMSG] if a message containing a control part is at the front of the STREAM head read queue.
RPROTDAT	Deliver the control part of a message as data when a process issues a read()
RPROTDIS	Discard the control part of a message, delivering any data part, when a process issues a read()

The following macro is defined for use with I_SWOPT:

SNDZERO	Send a zero-length message downstream when a write() of 0 bytes occurs.

The following macros are defined for use with I_ATMARK:

ANYMARK	Check if the message is marked.
LASTMARK	Check if the message is the last one marked on the queue.

The following macro is defined for use with I_UNLINK:

MUXID_ALL	Unlink all STREAMs linked to the STREAM associated with fildes.

The following macros are defined for getmsg(), getpmsg(), putmsg(), and putpmsg():

MSG_ANY	Receive any message.
MSG_BAND	Receive message from specified band.
MSG_HIPRI	Send/receive high-priority message.
MORECTL	More control information is left in message.
MOREDATA	More data is left in message.

The <stropts.h> header can make visible all of the symbols from <unistd.h>.

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Standard

SEE ALSO | close(2), fcntl(2), getmsg(2), ioctl(2), open(2), pipe(2), poll(2), putmsg(2), read(2), write(2), signal(3C), types.h(3HEAD), unistd.h(3HEAD), attributes(5), standards(5)

NAME

syslog.h, syslog – definitions for system error logging

SYNOPSIS

```
#include <syslog.h>
```

DESCRIPTION

The `<syslog.h>` header defines the following symbolic constants, zero or more of which can be OR'ed together to form the `logopt` option of `openlog()`:

`LOG_PID`	Log the process ID with each message.
`LOG_CONS`	Log to the system console on error.
`LOG_NDELAY`	Connect to syslog daemon immediately.
`LOG_ODELAY`	Delay open until `syslog()` is called.
`LOG_NOWAIT`	Do not wait for child processes.

The following symbolic constants are defined as possible values of the *facility* argument to `openlog()`:

`LOG_KERN`	reserved for message generated by the system
`LOG_USER`	message generated by a process
`LOG_MAIL`	reserved for message generated by mail system
`LOG_NEWS`	reserved for message generated by news system
`LOG_UUCP`	reserved for message generated by UUCP system
`LOG_DAEMON`	reserved for message generated by system daemon
`LOG_AUTH`	reserved for message generated by authorization daemon
`LOG_CRON`	reserved for message generated by clock daemon
`LOG_LPR`	reserved for message generated by printer system
`LOG_LOCAL0`	reserved for local use
`LOG_LOCAL1`	reserved for local use
`LOG_LOCAL2`	reserved for local use
`LOG_LOCAL3`	reserved for local use
`LOG_LOCAL4`	reserved for local use
`LOG_LOCAL5`	reserved for local use
`LOG_LOCAL6`	reserved for local use
`LOG_LOCAL7`	reserved for local use

The following is declared as a macro for constructing the *maskpri* argument to `setlogmask()`. The following macro expands to an expression of type `int` when the argument *pri* is an expre ssion of type `int`:

`LOG_MASK(`*pri*`)`	a mask for priority *pri*

The following constants are defined as possible values for the *priority* argument of `syslog()`:

`LOG_EMERG`	A panic condition was reported to all processes.
`LOG_ALERT`	A condition that should be corrected immediately.
`LOG_CRIT`	A critical condition.
`LOG_ERR`	An error message.
`LOG_WARNING`	A warning message.
`LOG_NOTICE`	A condition requiring special handling.
`LOG_INFO`	A general information message.
`LOG_DEBUG`	A message useful for debugging programs.

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Standard

SEE ALSO

`syslog`(3C), `attributes`(5), `standards`(5)

NAME

tar.h, tar – extended tar definitions

SYNOPSIS

```
#include <tar.h>
```

DESCRIPTION

The `<tar.h>` header defines header block definitions as follows.

Name	Description	Value
TMAGIC	"ustar"	ustar plus null byte
TMAGLEN	6	length of the above
TVERSION	"00"	00 without a null byte
TVERSLEN	2	length of the above

`Typeflag` field definitions:

Name	Description	Value
REGTYPE	'0'	regular file
AREGTYPE	'\0'	regular file
LNKTYPE	'1'	link
SYMTYPE	'2'	symbolic link
CHRTYPE	'3'	character special
BLKTYPE	'4'	block special
DIRTYPE	'5'	directory
FIFOTYPE	'6'	FIFO special
CONTTYPE	'7'	reserved

`Mode` field bit definitions (octal):

Name	Description	Value
TSUID	04000	set UID on execution
TSGID	02000	set GID on execution
TSVTX	01000	on directories, restricted deletion flag
TUREAD	00400	read by owner
TUWRITE	00200	write by owner special

Name	Description	Value
TUEXEC	00100	execute/search by owner
TGREAD	00040	read by group
TGWRITE	00020	write by group
TGEXEC	00010	execute/search by group
TOREAD	00004	read by other
TOWRITE	00002	write by other
TOEXEC	00001	execute/search by other

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Standard

SEE ALSO

pax(1), attributes(5), standards(5)

NAME

tcp.h, tcp – definitions for the Internet Transmission Control Protocol (TCP)

SYNOPSIS

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

DESCRIPTION

The `<netinet/tcp.h>` header defines the following macro for use as a socket option at the `IPPROTO_TCP` level:

`TCP_NODELAY` Avoid coalescing of small segments.

The macro is defined in the header. The implementation need not allow the value of the option to be set with `setsockopt()` or retrieved with `getsockopt()`.

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Standard

SEE ALSO

`getsockopt`(3XNET), `socket.h`(3HEAD), `attributes`(5), `standards`(5)

NAME

termios.h, termios – define values for termios

SYNOPSIS

```
#include <termios.h>
```

DESCRIPTION

The `<termios.h>` header contains the definitions used by the terminal I/O interfaces. See `termios`(3C) and `termio`(7I) for an overview of the terminal interface.

The termios Structure

The following data types are defined through `typedef`:

`cc_t` used for terminal special characters

`speed_t` used for terminal baud rates

`tcflag_t` used for terminal modes

The above types are all unsigned integer types.

The implementation supports one or more programming environments in which the widths of `cc_t`, `speed_t`, and `tcflag_t` are no greater than the width of type `long`. The names of these programming environments can be obtained using the `confstr`(3C) function or the `getconf`(1) utility.

The `termios` structure is defined and includes the following members:

```
tcflag_t c_iflag        /* input modes */
tcflag_t c_oflag        /* output modes */
tcflag_t c_cflag        /* control modes */
tcflag_t c_lflag        /* local modes */
cc_t     c_cc[NCCS]     /* control characters */
```

A definition is provided for:

`NCCS` size of the array `c_cc` for control characters

The following subscript names for the array `c_cc` are defined:

Subscript Usage Canonical Mode	Subscript Usage Non-Canonical Mode	Description
`VEOF`		EOF character
`VEOL`		EOL character
`VERASE`		ERASE character
`VINTR`	`VINTR`	INTR character
`VKILL`		KILL character
	`VMIN`	MIN value
`VQUIT`	`VQUIT`	QUIT character
`VSTART`	`VSTART`	START character

Subscript Usage Canonical Mode	Subscript Usage Non-Canonical Mode	Description
VSTOP	VSTOP	STOP character
VSUSP	VSUSP	SUSP character
	VTIME	TIME value

The subscript values are unique, except that the VMIN and VTIME subscripts can have the same values as the VEOF and VEOL subscripts, respectively.

The header file provides the flags described below.

Input Modes

The c_iflag field describes the basic terminal input control:

BRKINT	Signal interrupt on break.
ICRNL	Map CR to NL on input.
IGNBRK	Ignore break condition.
IGNCR	Ignore CR.
IGNPAR	Ignore characters with parity errors.
INLCR	Map NL to CR on input.
INPCK	Enable input parity check.
ISTRIP	Strip character.
IXANY	Enable any character to restart output.
IXOFF	Enable start/stop input control.
IXON	Enable start/stop output control.
PARMRK	Mark parity errors.

Output Modes

The c_oflag field specifies the system treatment of output:

OPOST	Post-process output.
ONLCR	Map NL to CR-NL on output.
OCRNL	Map CR to NL on output.
ONOCR	No CR output at column 0.
ONLRET	NL performs CR function.
OFILL	Use fill characters for delay.
NLDLY	Select newline delays:
	NL0 newline type 0

	NL1	newline type 1
CRDLY	Select carriage-return delays:	
	CR0	carriage-return delay type 0
	CR1	carriage-return delay type 1
	CR2	carriage-return delay type 2
	CR3	carriage-return delay type 3
TABDLY	Select horizontal-tab delays:	
	TAB0	horizontal-tab delay type 0
	TAB1	horizontal-tab delay type 1
	TAB2	horizontal-tab delay type 2
	TAB3	expand tabs to spaces
BSDLY	Select backspace delays:	
	BS0	backspace-delay type 0
	BS1	backspace-delay type 1
VTDLY	Select vertical-tab delays:	
	VT0	vertical-tab delay type 0
	VT1	vertical-tab delay type 1
FFDLY	Select form-feed delays:	
	FF0	form-feed delay type 0
	FF1	form-feed delay type 1

Baud Rate Selection

The input and output baud rates are stored in the `termios` structure. These are the valid values for objects of type `speed_ t`. The following values are defined, but not all baud rates need be supported by the underlying hardware.

B0	Hang up
B50	50 baud
B75	75 baud
B110	110 baud
B134	134.5 baud
B150	150 baud
B200	200 baud
B300	300 baud
B600	600 baud

B1200	1 200 baud
B1800	1 800 baud
B2400	2 400 baud
B4800	4 800 baud
B9600	9 600 baud
B19200	19 200 baud
B38400	38 400 baud

Control Modes

The c_cflag field describes the hardware control of the terminal; not all values specified are required to be supported by the underlying hardware:

CSIZE	Character size:
CS5	5 bits
CS6	6 bits
CS7	7 bits
CS8	8 bits
CSTOPB	Send two stop bits, else one.
CREAD	Enable receiver.
PARENB	Parity enable.
PARODD	Odd parity, else even.
HUPCL	Hang up on last close.
CLOCAL	Ignore modem status lines.

The implementation supports the functionality associated with the symbols CS7, CS8, CSTOPB, PARODD, and PARENB.

Local Modes

The c_lflag field of the argument structure is used to control various terminal functions:

ECHO	Enable echo.
ECHOE	Echo erase character as error-correcting backspace.
ECHOK	Echo KILL.
ECHONL	Echo NL.
ICANON	Canonical input (erase and kill processing).
IEXTEN	Enable extended input character processing.
ISIG	Enable signals.
NOFLSH	Disable flush after interrupt or quit.

TOSTOP Send SIGTTOU for background output.

Attribute Selection

The following symbolic constants for use with tcsetattr() are defined:

TCSANOW Change attributes immediately.

TCSADRAIN Change attributes when output has drained.

TCSAFLUSH Change attributes when output has drained; also flush pending input.

Line Control

The following symbolic constants for use with tcflush() are defined:

TCIFLUSH Flush pending input.

TCIOFLUSH Flush both pending input and untransmitted output.

TCOFLUSH Flush untransmitted output.

The following symbolic constants for use with tcflow() are defined:

TCIOFF Transmit a STOP character, intended to suspend input data.

TCION Transmit a START character, intended to restart input data.

TCOOFF Suspend output.

TCOON Restart output.

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Standard

SEE ALSO

getconf(1), cfgetispeed(3C), cfsetispeed(3C), confstr(3C), tcdrain(3C), tcflow(3C), tcflush(3C), tcgetattr(3C), tcgetsid(3C), tcsendbreak(3C), tcsetattr(3C), attributes(5), standards(5)

NAME

tgmath.h, tgmath – type-generic macros

SYNOPSIS

```
#include <tgmath.h>
```

DESCRIPTION

The <tgmath.h> header includes the headers <math.h> and <complex.h> and defines several type-generic macros.

Of the functions contained within the <math.h> and <complex.h> headers without an f (float) or l (long double) suffix, several have one or more parameters whose corresponding real type is double. For each such function except modf(3M), there is a corresponding type-generic macro. The parameters whose corresponding real type is double in the function synopsis are generic parameters. Use of the macro invokes a function whose corresponding real type and type domain are determined by the arguments for the generic parameters.

Use of the macro invokes a function whose generic parameters have the corresponding real type determined as follows:

- First, if any argument for generic parameters has type long double, the type determined is long double.
- Otherwise, if any argument for generic parameters has type double or is of integer type, the type determined is double.
- Otherwise, the type determined is float.

For each unsuffixed function in the <math.h> header for which there is a function in the <complex.h> header with the same name except for a c prefix, the corresponding type-generic macro (for both functions) has the same name as the function in the <math.h> header. The corresponding type-generic macro for fabs() and cabs() is fabs().

<math.h> Function	<complex.h> Function	Type-Generic Macro
acos()	cacos()	acos()
asin()	casin()	asin()
atan()	catan()	atan()
acosh()	cacosh()	acosh()
asinh()	casinh()	asinh()
atanh()	catanh()	atanh()
cos()	ccos()	cos()
sin()	csin()	sin()
tan()	ctan()	tan()
cosh()	ccosh()	cosh()

<math.h> Function	<complex.h> Function	Type-Generic Macro
sinh()	csinh()	sinh()
tanh()	ctanh()	tanh()
exp()	cexp()	exp()
log()	clog()	log()
pow()	cpow()	pow()
sqrt()	csqrt()	sqrt()
fabs()	cfabs()	fabs()

If at least one argument for a generic parameter is complex, then use of the macro invokes a complex function; otherwise, use of the macro invokes a real function.

For each unsuffixed function in the <math.h> header without a c-prefixed counterpart in the <complex.h> header, the corresponding type-generic macro has the same name as the function. These type-generic macros are:

```
atan2()       fma()     llround()       remainder()
cbrt()        fmax()    log10()         remquo()
ceil()        fmin()    log1p()         rint()
copysign()    fmod()    log2()          round()
erf()         frexp()   logb()          scalbn()
erfc()        hypot()   lrint()         scalbln()
exp2()        ilogb()   lround()        tgamma()
expm1()       ldexp()   nearbyint()     trunc()
fdim()        lgamma()  nextafter()
floor()       llrint()  nexttoward()
```

If all arguments for generic parameters are real, then use of the macro invokes a real function; otherwise, use of the macro results in undefined behavior.

For each unsuffixed function in the <complex.h> header that is not a c-prefixed counterpart to a function in the <math.h> header, the corresponding type-generic macro has the same name as the function. These type-generic macros are:

```
carg()
cimag()
conj()
cproj()
creal()
```

Use of the macro with any real or complex argument invokes a complex function.

USAGE

Functions invoked by use of type-generic macros are invoked with the declarations listed below.

```
#include <tgmath.h>
int n;
float f;
```

```
double d;
long double ld;
float complex fc;
double complex dc;
long double complex ldc;
```

The following are the type-generic macros that invoke the functions that are invoked with the preceding declarations.

Macro	Use Invokes
exp(n)	exp(n), the function
acosh(f)	acoshf(f)
sin(d)	sin(d), the function
atan(ld)	atanl(ld)
log(fc)	clogf(fc)
sqrt(dc)	csqrt(dc)
pow(ldc,f)	cpowl(ldc, f)
remainder(n,n)	remainder(n, n), the function
nextafter(d,f)	nextafter(d, f), the function
nexttoward(f,ld)	nexttowardf(f, ld)
copysign(n,ld)	copysignl(n, ld)
ceil(fc)	undefined behavior
rint(dc)	undefined behavior
fmax(ldc,ld)	undefined behavior
carg(n)	carg(n), the function
cproj(f)	cprojf(f)
creal(d)	creal(d), the function
cimag(ld)	cimagl(ld)
cabs(fc)	cabsf(fc)
carg(dc)	carg(dc), the function
cproj(ldc)	cprojl(ldc)

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Standard

SEE ALSO

`modf`(3M), `complex.h`(3HEAD), `math.h`(3HEAD), `cabs`(3M), `fabs`(3M), `attributes`(5), `standards`(5)

NAME

timeb.h, timeb – additional definitions for date and time

SYNOPSIS

```
#include <sys/timeb.h>
```

DESCRIPTION

The `<sys/timeb.h>` header defines the `timeb` structure, which includes the following members:

```
time_t         time        /* the seconds portion of the current time */
unsigned short millitm     /* the milliseconds portion of the current time */
short          timezone    /* the local timezone in minutes west  of Greenwich */
short          dstflag     /* TRUE if Daylight Savings Time is in effect */
```

The `time_t` type is defined as described in `<sys/types.h>`.

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Standard

SEE ALSO

`time.h`(3HEAD), `types.h`(3HEAD), `attributes`(5), `standards`(5)

NAME

time.h, 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 [0,60] */
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.

The <time.h> header defines the following symbolic names:

NULL
Null pointer constant.

CLOCKS_PER_SEC
A number used to convert the value returned by the clock() function into seconds. See clock(3C).

CLOCK_PROCESS_CPUTIME_ID
The identifier of the CPU-time clock associated with the process making a clock() or timer*() function call.

CLOCK_THREAD_CPUTIME_ID
The identifier of the CPU-time clock associated with the thread making a clock() or timer*() function call.

The <time.h> header declares the timespec structure, which has the following members:

```
time_t tv_sec     /* seconds */
long   tv_nsec     /* nanoseconds */
```

The <time.h> header declares the itimerspec structure, which has 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 system-wide realtime clock.
TIMER_ABSTIME	Flag indicating time is absolute. For functions taking timer objects, this refers to the clock associated with the timer.

CLOCK_MONOTONIC	The identifier for the system-wide monotonic clock, which is defined as a clock whose value cannot be set with clock_settime() and that cannot have backward clock jumps. The maximum possible clock jump is implementation-defined. See clock_settime(3RT).

The clock_t, size_t, time_t, clockid_t, and timer_t types are defined as described in <sys/types.h>. See types.h(3HEAD).

Although the value of CLOCKS_PER_SEC is required to be 1 million on all standard-conforming systems, it can be variable on other systems, and it should not be assumed that CLOCKS_PER_SEC is a compile-time constant.

The <time.h> header provides a declaration for getdate_err.

The following are declared as variables:

```
extern int daylight;
extern long timezone;
extern char *tzname[];
```

Inclusion of the <time.h> header can make visible all symbols from the <signal.h> header.

USAGE

The range [0,60] for tm_sec allows for the occasional leap second.

tm_year is a signed value; therefore, years before 1900 can be represented.

To obtain the number of clock ticks per second returned by the times() function, applications should call sysconf(_SC_CLK_TCK). See times(2) and sysconf(3C).

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Standard

SEE ALSO

time(2), utime(2), clock(3C), ctime(3C), difftime(3C), getdate(3C), mktime(3C), strftime(3C), strptime(3C), types.h(3HEAD), clock_settime(3RT), nanosleep(3RT), timer_create(3RT), timer_delete(3RT), timer_settime(3RT), attributes(5), standards(5)

NAME

times.h, times – file access and modification times structure

SYNOPSIS

```
#include <sys/times.h>
```

DESCRIPTION

The <sys/times.h> header defines the structure tms, which is returned by times() and includes the following members:

```
clock_t tms_utime      /* user CPU time */
clock_t tms_stime      /* system CPU time */
clock_t tms_cutime     /* user CPU time of terminated child processes */
clock_t tms_cstime     /* system CPU time of terminated child processes */
```

The clock_t type is defined as described in <sys/types.h>.

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Standard

SEE ALSO

times(2), types.h(3HEAD), attributes(5), standards(5)

NAME

types32.h, 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.h, types – primitive system data types

SYNOPSIS

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

DESCRIPTION

The data types defined in <sys/types.h> are as follows:

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
```

Preprocessor Symbols

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 be compiled only 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 be compiled only as part of a 64-bit version of a given C program. See EXAMPLES.

This header incorporates definitions of other preprocessor symbols that can be useful when keeping code portable between different instruction set architectures.

`_LITTLE_ENDIAN`
`_BIG_ENDIAN`
: The natural byte order of the processor. A pointer to an `int` points to the least/most significant byte of that `int`.

`_STACK_GROWS_UPWARD`
`_STACK_GROWS_DOWNWARD`
: The processor specific direction of stack growth. A push onto the stack increases/decreases the stack pointer, so it stores data at successively higher/lower addresses.

`_CHAR_IS_UNSIGNED`
`_CHAR_IS_SIGNED`
: The C Compiler implements objects of type `char` as `unsigned` or `signed` respectively. This is really an implementation choice of the compiler, but it is specified in the ABI and tends to be uniform across compilers for an instruction set architecture.

`_CHAR_ALIGNMENT`
`_SHORT_ALIGNMENT`
`_INT_ALIGNMENT`
`_LONG_ALIGNMENT`
`_LONG_LONG_ALIGNMENT`
`_DOUBLE_ALIGNMENT`
`_LONG_DOUBLE_ALIGNMENT`
`_POINTER_ALIGNMENT`
`_FLOAT_ALIGNMENT`
: The ABI defines alignment requirements of each of the primitive object types. Some, if not all, might be hardware requirements as well. The values are expressed in bytes.

_MAX_ALIGNMENT
The most stringent alignment requirement as specified by the ABI. Equal to the maximum of all the above _*XXX*_ALIGNMENT values.

_LONG_LONG_ALIGNMENT_32
The 32-bit ABI supported by a 64-bit kernel may have different alignment requirements for primitive object types. The value of this identifier is expressed in bytes.

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.

EXAMPLES

EXAMPLE 1 Use of preprocessor symbol _LP64.

In the following example, the preprocessor symbol _LP64 defines sections of code that will be compiled only as part of a 64-bit version of the given C program.

```
#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
```

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Stable

SEE ALSO

types32.h(3HEAD), attributes(5), standards(5)

NAME

ucontext.h, ucontext – user context

SYNOPSIS

```
#include <ucontext.h>
```

DESCRIPTION

The `<ucontext.h>` header defines the `ucontext_t` type as a structure that includes at least the following members:

```
ucontext_t  uc_link
sigset_t    uc_sigmask
stack_t     uc_stack
mcontext_t  uc_mcontext
```

The `uc_link` member is a pointer to the context that to be resumed when this context returns. If `uc_link` is equal to 0, this context is the main context and the process exits when this context returns.

The `uc_sigmask` member defines the set of signals that are blocked when this context is active. See `sigprocmask`(2).

The `uc_stack` member defines the stack used by this context. See `sigaltstack`(2).

The `uc_mcontext` member contains the saved set of machine registers and any implementation-specific context data. Portable applications should not modify or access `uc_mcontext`.

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Standard

SEE ALSO

`getcontext`(2), `sigaction`(2), `sigaltstack`(2), `sigprocmask`(2), `makecontext`(3C), `attributes`(5), `standards`(5)

NAME

uio.h, uio – definitions for vector I/O operations

SYNOPSIS

```
#include <sys/uio.h>
```

DESCRIPTION

The <sys/uio.h> header defines the iovec structure, which includes the following members:

```
void   *iov_base     /* base address of a memory region for input or output */
size_t iov_len       /* size of the memory pointed to by iov_base */
```

The <sys/uio.h> header uses the iovec structure for scatter/gather I/O.

The ssize_t and size_t types are defined as described in <sys/types.h>.

USAGE

The symbol {IOV_MAX} defined in <limits.h> should always be used to learn about the limits on the number of scatter/gather elements that can be processed in one call, instead of assuming a fixed value.

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Standard

SEE ALSO

read(2), write(2), limits.h(3HEAD), types.h(3HEAD), attributes(5), standards(5)

NAME

ulimit.h, ulimit – ulimit commands

SYNOPSIS

```
#include <ulimit.h>
```

DESCRIPTION

The `<ulimit.h>` header defines the following symbolic constants used by the `ulimit()` function.

`UL_GETFSIZE` Get maximum file size.

`UL_SETFSIZE` Set maximum file size.

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Standard

SEE ALSO

`ulimit`(2), `attributes`(5), `standards`(5)

NAME

un.h, 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.h(3HEAD).

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Standard

SEE ALSO

bind(3SOCKET), bind(3XNET), socket.h(3HEAD), socket(3SOCKET), socket(3XNET), socketpair(3SOCKET), socketpair(3XNET), attributes(5), standards(5)

NAME

unistd.h, 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. This constant is not defined for the SUSv3 environment.

Mandatory Symbolic Constants

The following symbolic constants, if defined in `<unistd.h>`, have a value of -1, 0, or greater, unless otherwise specified below. If these are undefined, the `fpathconf`(2), `pathconf`(2), or `sysconf`(3C) functions can be used to determine whether the option is provided for a particular invocation of the application.

If a symbolic constant is defined with the value -1, the option is not supported. Headers, data types, and function interfaces required only for the option need not be supplied. An application that attempts to use anything associated only with the option is considered to be requiring an extension.

If a symbolic constant is defined with a value greater than zero, the option is always supported when the application is executed. All headers, data types, and functions are present and operate as specified.

If a symbolic constant is defined with the value zero, all headers, data types, and functions are present. The application can check at runtime to see whether the option is supported by calling `fpathconf()`, `pathconf()`, or `sysconf()` with the indicated *name* parameter.

Unless explicitly specified otherwise, the behavior of functions associated with an unsupported option is unspecified, and an application that uses such functions without first checking `fpathconf()`, `pathconf()`, or `sysconf()` is considered to be requiring an extension.

_POSIX_ADVISORY_INFO
Implementation supports the Advisory Information option.

_POSIX_ASYNCHRONOUS_IO
Implementation supports the Asynchronous Input and Output option.

_POSIX_BARRIERS
Implementation supports the Barriers option.

_POSIX_CLOCK_SELECTION
Implementation supports the Clock Selection option.

_POSIX_CPUTIME
Implementation supports the Process CPU-Time Clocks option.

_POSIX_FSYNC
Implementation supports the File Synchronisation option.

_POSIX_IPV6
Implementation supports the IPv6 option.

_POSIX_JOB_CONTROL
Implementation supports job control.

_POSIX_MAPPED_FILES
Implementation supports the Memory Mapped Files option.

_POSIX_MEMLOCK
Implementation supports the Process Memory Locking option.

_POSIX_MEMLOCK_RANGE
Implementation supports the Range Memory Locking option.

_POSIX_MEMORY_PROTECTION
Implementation supports the Memory Protection option.

_POSIX_MESSAGE_PASSING
Implementation supports the Message Passing option.

_POSIX_MONOTONIC_CLOCK
Implementation supports the Monotonic Clock option.

_POSIX_PRIORITY_SCHEDULING
Implementation supports the Process Scheduling option.

_POSIX_RAW_SOCKETS
Implementation supports the Raw Sockets option.

_POSIX_READER_WRITER_LOCKS
Implementation supports the Read-Write Locks option.

_POSIX_REALTIME_SIGNALS
Implementation supports the Realtime Signals Extension option.

_POSIX_REGEXP
Implementation supports the Regular Expression Handling option.

_POSIX_SAVED_IDS
The exec functions (see exec(2)) save the effective user and group.

_POSIX_SEMAPHORES
Implementation supports the Semaphores option.

_POSIX_SHARED_MEMORY_OBJECTS
Implementation supports the Shared Memory Objects option.

_POSIX_SHELL
Implementation supports the POSIX shell.

_POSIX_SPAWN
Implementation supports the Spawn option.

_POSIX_SPIN_LOCKS
Implementation supports the Spin Locks option.

_POSIX_SPORADIC_SERVER
Implementation supports the Process Sporadic Server option.

_POSIX_SYNCHRONIZED_IO
Implementation supports the Synchronized Input and Output option.

_POSIX_THREAD_ATTR_STACKADDR
Implementation supports the thread stack address attribute option.

_POSIX_THREAD_ATTR_STACKSIZE
Implementation supports the thread stack size attribute option.

_POSIX_THREAD_CPUTIME
Implementation supports the Thread CPU-Time Clocks option.

_POSIX_THREAD_PROCESS_SHARED
Implementation supports the process-shared synchronization option.

_POSIX_THREAD_SAFE_FUNCTIONS
Implementation supports the thread-safe functions option.

_POSIX_THREAD_SPORADIC_SERVER
Implementation supports the Thread Sporadic Server option.

_POSIX_THREADS
Implementation supports the threads option.

_POSIX_TIMERS
Implementation supports the Timers option.

_POSIX_TIMEOUTS
Implementation supports the Timeouts option.

_POSIX_TRACE
Implementation supports the Trace option.

_POSIX_TRACE_EVENT_FILTER
Implementation supports the Trace Event Filter option.

`_POSIX_TRACE_INHERIT`
Implementation supports the Trace Inherit option.

`_POSIX_TRACE_LOG`
Implementation supports the Trace Log option.

`_POSIX_TYPED_MEMORY_OBJECTS`
Implementation supports the Typed Memory Objects option.

`_POSIX_V6_ILP32_OFF32`
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.

`_POSIX_V6_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.

`_POSIX_V6_LP64_OFF64`
Implementation provides a C-language compilation environment with 32-bit `int` and 64-bit `long`, `pointer`, and `off_t` types.

`_POSIX_V6_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.

`_POSIX_XOPEN_STREAMS`
Implementation supports the XSI STREAMS Option Group.

`_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_PBS`
Implementation supports the Batch Environment Services and Utilities option.

`_POSIX2_PBS_ACCOUNTING`
Implementation supports the Batch Accounting option.

`_POSIX2_PBS_CHECKPOINT`
Implementation supports the Batch Checkpoint/Restart option.

`_POSIX2_PBS_LOCATE`
Implementation supports the Locate Batch Job Request option.

`_POSIX2_PBS_MESSAGE`
Implementation supports the Batch Job Message Request option.

`_POSIX2_PBS_TRACK`
Implementation supports the Track Batch Job Request option.

`_POSIX2_SW_DEV`
Implementation supports the Software Development Utilities option.

`_POSIX2_UPE`
Implementation supports the User Portability Utilities option.

`_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.

`_XOPEN_ENH_I18N`
Implementation supports the Issue 4, Version 2 Enhanced Internationalization Feature Group.

`_XOPEN_LEGACY`
Implementation supports the Legacy Feature Group.

`_XOPEN_REALTIME`
Implementation supports the X/Open Realtime Feature Group.

`_XOPEN_SHM`
Implementation supports the Issue 4, Version 2 Shared Memory Feature Group.

`_XOPEN_UNIX`
X/Open CAE Specification, January 1997, System Interfaces and Headers, Issue 5 (ISBN: 1-85912-181-0, C606).

`_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).

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, can be queried with respect to a specific file using the pathconf() or fpathconf() functions.

_POSIX_ASYNC_IO	Asynchronous input or output operations can be performed for the associated file.
_POSIX_PRIO_IO	Prioritized input or output operations can be performed for the associated file.
_POSIX_SYNC_IO	Synchronized input or output operations can 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 x86:

_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_POSIX_V6_ILP32_OFF32_CFLAGS
_CS_POSIX_V6_ILP32_OFF32_LDFLAGS	_CS_POSIX_V6_ILP32_OFF32_LIBS
_CS_POSIX_V6_ILP32_OFF32_LINTFLAGS	_CS_POSIX_V6_ILP32_OFFBIG_CFLAGS
_CS_POSIX_V6_ILP32_OFFBIG_LDFLAGS	_CS_POSIX_V6_ILP32_OFFBIG_LIBS
_CS_POSIX_V6_ILP32_OFFBIG_LINTFLAGS	_CS_POSIX_V6_WIDTH_RESTRICTED_ENV
_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_POSIX_V6_LP64_OFF64_CFLAGS	_CS_POSIX_V6_LP64_OFF64_LDFLAGS
_CS_POSIX_V6_LP64_OFF64_LIBS	_CS_POSIX_V6_LP64_OFF64_LINTFLAGS
_CS_POSIX_V6_LPBIG_OFFBIG_CFLAGS	_CS_POSIX_V6_LPBIG_OFFBIG_LDFLAGS
_CS_POSIX_V6_LPBIG_OFFBIG_LIBS	_CS_POSIX_V6_LPBIG_OFFBIG_LINTFLAGS
_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_PBS	_SC_2_PBS_ACCOUNTING
_SC_2_PBS_CHECKPOINT	_SC_2_PBS_LOCATE
_SC_2_PBS_MESSAGE	_SC_2_PBS_TRACK
_SC_2_SW_DEV	_SC_2_UPE
_SC_2_VERSION	_SC_ADVISORY_INFO
_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_BARRIERS
_SC_BC_BASE_MAX	_SC_BC_DIM_MAX
_SC_BC_SCALE_MAX	_SC_BC_STRING_MAX
_SC_CHILD_MAX	_SC_CLK_TCK
_SC_CLOCK_SELECTION	_SC_COLL_WEIGHTS_MAX
_SC_CPUTIME	_SC_DELAYTIMER_MAX
_SC_EXPR_NEST_MAX	_SC_FSYNC
_SC_GETGR_R_SIZE_MAX	_SC_GETPW_R_SIZE_MAX
_SC_HOST_NAME_MAX	_SC_IOV_MAX
_SC_IPV6	_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_MONOTONIC_CLOCK	_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_RAW_SOCKETS
_SC_READER_WRITER_LOCKS	_SC_REALTIME_SIGNALS

_SC_REGEXP	_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_SHELL	_SC_SIGQUEUE_MAX
_SC_SPAWN	_SC_SPIN_LOCKS
_SC_SPORADIC_SERVER	_SC_SS_REPL_MAX
_SC_STREAM_MAX	_SC_SYMLOOP_MAX
_SC_SYNCHRONIZED_IO	_SC_THREAD_ATTR_STACKADDR
_SC_THREAD_ATTR_STACKSIZE	_SC_THREAD_CPUTIME
_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_THREAD_SPORADIC_SERVER	_SC_THREADS
_SC_THREAD_SAFE_FUNCTIONS	_SC_THREAD_STACK_MIN
_SC_THREAD_THREADS_MAX	_SC_TIMEOUTS
_SC_TIMER_MAX	_SC_TIMERS
_SC_TRACE	_SC_TRACE_EVENT_FILTER
_SC_TRACE_EVENT_NAME_MAX	_SC_TRACE_INHERIT
_SC_TRACE_LOG	_SC_TRACE_NAME_MAX
_SC_TRACE_SYS_MAX	_SC_TRACE_USER_EVENT_MAX
_SC_TTY_NAME_MAX	_SC_TYPED_MEMORY_OBJECTS
_SC_TZNAME_MAX	_SC_V6_ILP32_OFF32
_SC_V6_ILP32_OFFBIG	_SC_V6_LP64_OFF64
_SC_V6_LPBIG_OFFBIG	_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_STREAMS
_SC_XOPEN_UNIX	_SC_XOPEN_VERSION

_SC_XOPEN_XCU_VERSION

The constants _SC_PAGESIZE and _SC_PAGE_SIZE can be defined to have the same value.

The following symbolic constants are defined for the fpathconf(2) function:

_PC_2_SYMLINKS	_PC_ALLOC_SIZE_MIN
_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_REC_INCR_XFER_SIZE
_PC_REC_MAX_XFER_SIZE	_PC_REC_MIN_XFER_SIZE
_PC_REC_XFER_ALIGN	_PC_SYMLINK_MAX
_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.

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Standard

SEE ALSO

access(2), exec(2), fcntl(2), fpathconf(2), lseek(2), confstr(3C), lockf(3C), sysconf(3C), termios(3C), group(4), passwd(4), attributes(5), standards(5), termio(7I)

NAME

utime.h, utime – access and modification times structure

SYNOPSIS

```
#include <utime.h>
```

DESCRIPTION

The `<utime.h>` header declares the structure `utimbuf`, which includes the following members:

```
time_t actime     /* access time */
time_t modtime    /* modification time */
```

The times are measured in seconds since the Epoch.

The type `time_t` is defined as described in `<sys/types.h>`.

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Standard

SEE ALSO

`utime`(2), `types.h`(3HEAD), `attributes`(5), `standards`(5)

NAME

utmpx.h, utmpx – user accounting database definitions

SYNOPSIS

```
#include <utmpx.h>
```

DESCRIPTION

The <utmpx.h> header defines the utmpx structure, which includes the following members:

```
char           ut_user[]     /* user login name */
char           ut_id[]       /* unspecified initialization process identifier */
char           ut_line[]     /* device name */
pid_t          ut_pid        /* process ID */
short          ut_type       /* type of entry */
struct timeval ut_tv         /* time entry was made */
```

The pid_t type is defined through typedef as described in <sys/types.h>.

The timeval structure is defined as described in <sys/time.h>.

Inclusion of the <utmpx.h> header can also make visible all symbols from <sys/time.h>.

The following symbolic constants are defined as possible values for the ut_type member of the utmpx structure:

EMPTY	No valid user accounting information.
BOOT_TIME	Identifies time of system boot.
OLD_TIME	Identifies time when system clock changed.
NEW_TIME	Identifies time after system clock changed.
USER_PROCESS	Identifies a process.
INIT_PROCESS	Identifies a process spawned by the init process.
LOGIN_PROCESS	Identifies the session leader of a logged-in user.
DEAD_PROCESS	Identifies a session leader who has exited.

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Standard

SEE ALSO

endutxent(3C), time.h(3HEAD), types.h(3HEAD), attributes(5), standards(5)

NAME

utsname.h, utsname – system name structure

SYNOPSIS

```
#include <sys/utsname.h>
```

DESCRIPTION

The `<sys/utsname.h>` header defines the structure `utsname`, which includes the following members:

```
char sysname[]      /* name of this implementation of the operating */
                    /* system */
char nodename[]     /* name of this node within an implementation- */
                    /* defined communications network */
char release[]      /* current release level of this implementation */
char version[]      /* current version level of this release */
char machine[]      /* name of the hardware type on which the system is */
                    /* running */
```

The character arrays are of unspecified size, but the data stored in them is terminated by a null byte.

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Standard

SEE ALSO

`uname`(2), `attributes`(5), `standards`(5)

NAME

values.h, 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.h`(3HEAD)

NAME

wait.h, wait – wait status

SYNOPSIS

```
#include <sys/wait.h>
```

DESCRIPTION

When a process waits for status from its children using either the `wait`(3C) or `waitpid`(3C) function, the status returned can 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()`.

`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.
`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`.
`WIFCONTINUED`(*stat*)	Evaluates to a non-zero value if status was returned for a child process that has continued.
`WIFEXITED`(*stat*)	Evaluates to a non-zero value if status was returned for a child process that terminated normally.
`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.
`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.
`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.

The `<sys/wait.h>` header defines the symbolic constants listed below for use with `waitpid`(3C).

`WNOHANG`	Do not hang if no status is available; return immediately.

WUNTRACED	Report status of stopped child process.

The symbolic constants listed below are defined as possible values for the *options* argument to waitid(2).

WEXITED	Wait for processes that have exited.
WSTOPPED	Status is returned for any child that has stopped upon receipt of a signal.
WCONTINUED	Status is returned for any child that was stopped and has been continued.
WNOHANG	Return immediately if there are no children to wait for.
WNOWAIT	Keep the process whose status is returned in infop in a waitable state.

The type idtype_t is defined as an enumeration type whose possible values include the following:

```
P_ALL
P_PID
P_PGID
```

The id_t and pid_t types are defined as described in <sys/types.h>.

The siginfo_t type is defined as described in <signal.h>.

The rusage structure is defined as described in <sys/resource.h>.

Inclusion of the <sys/wait.h> header can also make visible all symbols from <signal.h> and <sys/resource.h>.

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Standard

SEE ALSO

exit(2), waitid(2), exit(3C), wait(3C), waitpid(3C),attributes(5), standards(5)

NAME

wchar.h, wchar – wide-character handling

SYNOPSIS

```
#include <wchar.h>
```

DESCRIPTION

The <wchar.h> header defines the following types:

wchar_t	As described in <stddef.h>.
wint_t	An integer type capable of storing any valid value of wchar_t or WEOF.
wctype_t	A scalar type of a data object that can hold values which represent locale-specific character classification.
mbstate_t	An object type other than an array type that can hold the conversion state information necessary to convert between sequences of (possibly multi-byte) characters and wide characters. If a codeset is being used such that an mbstate_t needs to preserve more than two levels of reserved state, the results are unspecified.
FILE	As described in <stdio.h>.
size_t	As described in <stddef.h>.
va_list	As described in <stdarg.h>.

The implementation supports one or more programming environments in which the width of wint_t is no greater than the width of type long. The names of these programming environments can be obtained using the confstr(3C) function or the getconf(1) utility.

The <wchar.h> header defines the following macros:

WCHAR_MAX	The maximum value representable by an object of type wchar_t.
WCHAR_MIN	The minimum value representable by an object of type wchar_t.
WEOF	Constant expression of type wint_t that is returned by several WP functions to indicate end-of-file.
NULL	As described in <stddef.h>.

The tag tm is declared as naming an incomplete structure type, the contents of which are described in the header <time.h>.

Inclusion of the <wchar.h> header can make visible all symbols from the headers <ctype.h>, <string.h>, <stdarg.h>, <stddef.h>, <stdio.h>, <stdlib.h>, and <time.h>.

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Standard

SEE ALSO

`getconf`(1), `btowc`(3C), `confstr`(3C), `fgetwc`(3C), `getws`(3C), `fputwc`(3C), `fputws`(3C), `fwide`(3C), `fwprintf`(3C), `fwscanf`(3C), `getwc`(3C), `getwchar`(3C), `iswalpha`(3C), `iswctype`(3C), `mbsinit`(3C), `mbrlen`(3C), `mbrtowc`(3C), `mbsrtowcs`(3C), `towlower`(3C), `towupper`(3C), `ungetwc`(3C), `vfwprintf`(3C), `wcrtomb`(3C), `wcsrtombs`(3C), `wcstring`(3C), `wcsstr`(3C), `wcstod`(3C), `wcscoll`(3C), `wcsftime`(3C), `wcstol`(3C), `wcstoul`(3C), `wcswidth`(3C), `wcsxfrm`(3C), `wctob`(3C), `wctype`(3C), `wcwidth`(3C), `wmemchr`(3C), `wmemcmp`(3C), `wmemcpy`(3C), `wmemmove`(3C), `wmemset`(3C), `stdarg`(3EXT), `stddef.h`(3HEAD), `stdio.h`(3HEAD), `stdlib.h`(3HEAD), `string.h`(3HEAD), `time.h`(3HEAD), `wctype.h`(3HEAD), `attributes`(5), `standards`(5)

NAME

wctype.h, wctype – wide-character classification and mapping utilities

SYNOPSIS

```
#include <wctype.h>
```

DESCRIPTION

The <wctype.h> header defines the following types:

wint_t	As described in <wchar.h>.
wctrans_t	A scalar type that can hold values th at represent locale-specific character mappings.
wctype_t	As described in <wchar.h>.

The <wctype.h> header defines the following macro name:

WEOF	Constant expression of type wint_t that is returned by several MSE functions to indicate end-of-file.

For all functions described in this header that accept an argument of type wint_t, the value is representable as a wchar _t or equals the value of WEOF. If this argument has any other value, the behavior is undefined.

The behavior of these functions is affected by the LC_CTYPE category of the current locale.

Inclusion of the <wctype.h> header can make visible all symbols from the headers <ctype.h>, <stdarg.h>, <stddef.h>, <stdio.h>, <stdlib.h>, <string.h>, <time.h>, and <wchar.h>.

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Standard

SEE ALSO

iswalpha(3C), iswctype(3C), locale.h(3HEAD), setlocale(3C), stdarg(3EXT), stddef.h(3HEAD), stdio.h(3HEAD), stdlib.h(3HEAD), string.h(3HEAD), time.h(3HEAD), towctrans(3C), towlower(3C), towupper(3C), wctrans(3C), wctype(3C), attributes(5), standards(5)

NAME

wordexp.h, wordexp – word-expansion types

SYNOPSIS

```
#include <wordexp.h>
```

DESCRIPTION

The <wordexp.h> header defines the structures and symbolic constants used by the wordexp() and wordfree() functions. See wordexp(3C).

The structure type wordexp_t contains the following members:

```
size_t we_wordc        /* count of words matched by words */
char   **we_wordv      /* pointer to list of expanded words */
size_t we_offs         /* slots to reserve at the beginning of we_wordv
```

The *flags* argument to the wordexp() function is the bitwise-inclusive OR of the following flags:

WRDE_APPEND	Append words to those previously generated.
WRDE_DOOFFS	Number of null pointers to prepend to we_wordv.
WRDE_NOCMD	Fail if command substitution is requested.
WRDE_REUSE	The pwordexp argument was passed to a previous successful call to wordexp(), and has not been passed to wordfree(). The result is the same as if the application had called wordfree() and then called wordexp() without WRDE_REUSE.
WRDE_SHOWERR	Do not redirect stderr to /dev/null.
WRDE_UNDEF	Report error on an attempt to expand an undefined shell variable.

The following constants are defined as error return values:

WRDE_BADCHAR	One of the unquoted characters—<newline>, '|', '&', ';', '<', '>', '(', ')', '{', '}'—appears in words in an inappropriate context.
WRDE_BADVAL	Reference to undefined shell variable when WRDE_UNDEF is set in *flags*.
WRDE_CMDSUB	Command substitution requested when WRDE_NOCMD was set in *flags*.
WRDE_NOSPACE	Attempt to allocate memory failed.
WRDE_NOSYS	Reserved.
WRDE_SYNTAX	Shell syntax error, such as unbalanced parentheses or unterminated string.

The <wordexp.h> header defines the following type:

size_t	As described in <stddef.h>.

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Standard

SEE ALSO

`wordexp`(3C), `attributes`(5), `standards`(5)

Index

A

access and modification times structure — utime.h, 454
access and modification times structure — utime, 454
accounting files, — acct, 30
acct — process accounting file format, 30
additional definitions for date and time — timeb.h, 432
additional definitions for date and time — timeb, 432
alternative spellings — iso646.h, 69
alternative spellings — iso646, 69
alternative memory allocator library — libmapmalloc, 184
ar — archive file format, 33
archive file format — ar, 33
assert — verify program assertion, 36
assert.h — verify program assertion, 36
asynchronous I/O library — libaio, 75

B

basic security library — libbsm, 78
boolean type and values — stdbool.h, 398
boolean type and values — stdbool, 398

C

C library — libc, 80
C math library — libm, 166
category macros — locale.h, 349
category macros — locale, 349
codeset conversion facility — iconv.h, 61
codeset conversion facility — iconv, 61
Common Fibre Channel HBA information library — libhbaapi, 154
complex — complex arithmetic, 37
complex arithmetic — complex.h, 37
complex arithmetic — complex, 37
complex.h — complex arithmetic, 37
configuration adminstartion library — libcfgadm, 112
contract management library — libcontract, 113
cpio — cpio archive values, 39
cpio archive values — cpio.h, 39
cpio archive values — cpio, 39
cpio.h — cpio archive values, 39

D

define values for termios — termios.h, 423
define values for termios — termios, 423
definitions for internet operations — inet, 63
definitions for ndbm database operations — ndbm, 359
definitions for network database operations — netdb, 360
definitions for pattern matching functions — libgen.h, 151
definitions for pattern matching functions — libgen, 151

definitions for resource operations — resource.h, 369
definitions for resource operations — resource, 369
definitions for system error logging — syslog.h, 418
definitions for system error logging — syslog, 418
definitions for the `poll()` function — poll.h, 363
definitions for the `poll()` function — poll, 363
definitions for the Internet Transmission Co ntrol Protocol (TCP) — tcp.h, 422
definitions for the Internet Transmission Co ntrol Protocol (TCP) — tcp, 422
definitions for UNIX-domain sockets — un, 443
definitions for vector I/O operations — uio.h, 441
definitions for vector I/O operations — uio, 441
device ID library — libdevid, 131
layout service library — liblayout, 164
name—value pair library — libnvpair, 257
device information library — libdevinfo, 132
dirent — format of directory entries, 41
dirent.h — format of directory entries, 41
doors library — libdoor, 140
DTrace dynamic tracing software library — libdtrace, 141
dynamic linking library — libdl, 136

E

ELF access library — libelf, 143
encryption/decryption library — libcrypt, 117
errno — system error numbers, 42
errno.h — system error numbers, 42
extended tar definitions — tar.h, 420
extended tar definitions — tar, 420

F

fenv — floating-point environment, 47
fenv.h — floating-point environment, 47
file access and modification times structure — times.h, 435
file access and modification times structure — times, 435
file tree traversal — ftw.h, 58
file tree traversal — ftw, 58
File Access Control List library — libsec, 300
file control options, — fcntl, 43
filename-matching types — fnmatch.h, 57
filename-matching types — fnmatch, 57
fixed size integer types — inttypes.h, 66
fixed size integer types — inttypes, 66
fixed-width data types — types32, 436
float — floating types, 50
float.h — floating types, 50
floating types — float.h, 50
floating types — float, 50
floating-point environment — fenv.h, 47
floating-point environment — fenv, 47
floatingpoint — IEEE floating point definitions, 53
fmtmsg — message display structures, 55
fmtmsg.h — message display structures, 55
fnmatch — filename-matching types, 57
fnmatch.h — filename-matching types, 57
format of directory entries — dirent.h, 41
format of directory entries — dirent, 41
forms library — libform, 147
ftw — file tree traversal, 58
ftw.h — file tree traversal, 58

G

general administrative library — libadm, 74
Generic Security Services library — libgss, 152
glob — pathname pattern-matching types, 59
glob.h — pathname pattern-matching types, 59
graphics interface libraries
— lib300, 269
— lib300s, 269
— lib4014, 269
— lib450, 269
— libplot, 269
— libvt0, 269
group structure — grp.h, 60
group structure — grp, 60
grp — group structure, 60
grp.h — group structure, 60

I

iconv — codeset conversion facility, 61
iconv.h — codeset conversion facility, 61
IEEE arithmetic, floating point definitions — floatingpoint, 53
if — sockets local interfaces, 62
if.h — sockets local interfaces, 62
implementation-defined constants — limits.h, 339
implementation-defined constants — limits, 339
in — Internet Protocol family, 64
inet — definitions for internet operations, 63
  Default, 63
  Standard-conforming, 63
integer types — stdint.h, 400
integer types — stdint, 400
interactive command line input library — libtecla, 316
internationalization library — libintl, 160
Internet Protocol family — in, 64
Internet Protocol family — socket, 390
inttypes — fixed size integer types, 66
inttypes.h — fixed size integer types, 66
ipc — XSI interprocess communication access structure, 68
ipc.h — XSI interprocess communication access structure, 68
iso646 — alternative spellings, 69
iso646.h — alternative spellings, 69

K

kernel statistics library — libkstat, 161
Kernel Virtual Memory access library — libkvm, 162

L

langinfo — language information constants, 70
language data types, native — nl_types, 362
language information constants — langinfo, 70
lex library — libl, 163
lib300 — graphics interface libraries, 269
lib300s — graphics interface libraries, 269
lib4014 — graphics interface libraries, 269
lib450 — graphics interface libraries, 269
libadm — general administrative library, 74
libaio — asynchronous I/O library, 75
libbsdmalloc — memory allocator interface library, 77
libbsm — basic security library, 78
libc — C library, 80
libc — the C library, 80
libcfgadm — configuration adminstartion library, 112
libcontract — contract management library, 113
libcrypt — encryption/decryption library, 117
libcurses — screen handling and optimization library, 118
libdevid — device ID library, 131
liblayout — layout service library, 164
libnvpair — name—value pair library, 257
libdevinfo — device information library, 132
libdl — dynamic linking library, 136
libdmi — Sun Solstice Enterprise Agent DMI library, 137
libdmici — Sun Solstice Enterprise Agent Component library, 138
libdmimi — Sun Solstice Enterprise Agent Management library, 139
libdoor — doors library, 140
libdtrace — DTrace dynamic tracing software library, 141
libelf — ELF access library, 143
libform — forms library, 147
libgen — definitions for pattern matching functions, 151
libgen — string pattern-matching library, 149
libgen.h — definitions for pattern matching functions, 151
libgss — Generic Security Services library, 152
libhbaapi — Common Fibre Channel HBA information library, 154
libintl — internationalization library, 160
libkstat — kernel statistics library, 161
libkvm — Kernel Virtual Memory access library, 162
  /usr/lib/libkvm.so.1, 162
libl — lex library, 163
libm — C math library, 166
libmail — user mailbox lockfile management library, 182
libmalloc — memory allocation library, 183

libmapmalloc — alternative memory allocator library, 184
libmenu — menus library, 186
libmlib — mediaLib library, 188
libmp — multiple precision library, 245
libmtmalloc — multi-threaded memory allocator library, 246
libmvec — vector math library, 247
libnls — network listener service library, 248
libnsl — network services library, 249
libpam — PAM (Pluggable Authentication Module) library, 260
libpanel — panels library, 262
libpicl — PICL library, 264
libpicltree — PICL plug-in library, 265
libpkcs11 — PKCS#11 Cryptographic Framework library, 266
libplot — graphics interface libraries, 269
libpool — pool configuration manipulation library, 271
libposix4 — POSIX.1b Realtime Extensions library, 290
libpthread — POSIX threads library, 280
libpthread — posix threads library, /usr/lib/libpthread.so.1, 280
librac — remote asynchronous calls library, 283
library file format — ar, 33
libresolv — resolver library, 284
librpcsoc — obsolete RPC library, 286
librpcsvc — RPC services library, 287
librsm — remote shared memory interface library, 288
librt — POSIX.1b Realtime Extensions library, 290
librtld_db — runtime linker debugging library, 292
libscf — service configuration facility library, 295
libsec — File Access Control List library, 300
/usr/lib/libsec.so.1, 300
libsecdb — security attributes database library, 301
libsendfile — sendfile library, 303
libslp — service location protocol library, 304
libsmartcard — smartcard library, 305
libsocket — sockets library, 306
libssagent — Sun Solstice Enterprise Agent library, 308
libssasnmp — Sun Solstice Enterprise SNMP library, 309
libsys — system library, 310
libtecla — interactive command line input library, 316
libtermcap — screen handling and optimization library, 118
libtermlib — screen handling and optimization library, 118
libthread — threads library, 320
libtnfctl — TNF probe control library, 322
libucb — UCB source compatibility library, 324
libvolmgt — volume management library, 330
libvt0 — graphics interface libraries, 269
libw — wide character library, 331
libwsreg — product install registry library, 333
libxnet — X/Open Networking library, 335
liby — yacc library, 338
limits — implementation-defined constants, 339
limits.h — implementation-defined constants, 339
locale — category macros, 349
locale.h — category macros, 349

M

machine-dependent values, — values, 457
math — mathematical declarations, 351
math.h — mathematical declarations, 351
mathematical declarations — math.h, 351
mathematical declarations — math, 351
mediaLib library — libmlib, 188
memory management declarations — mman.h, 354
memory management declarations — mman, 354
memory allocation library — libmalloc, 183
memory allocator interface library — libbsdmalloc, 77
menus library — libmenu, 186
message display structures — fmtmsg.h, 55
message display structures — fmtmsg, 55
message queue structures — msg.h, 358
message queue structures — msg, 358
mman — memory management declarations, 354

mman.h — memory management declarations, 354
monetary — monetary types, 356
monetary types — monetary.h, 356
monetary types — monetary, 356
monetary.h — monetary types, 356
msg — message queue structures, 358
msg.h — message queue structures, 358
multi-threaded memory allocator library — libmtmalloc, 246
multiple precision library — libmp, 245

N

ndbm — definitions for ndbm database operations, 359
netdb — definitions for network database operations, 360
network listener service library — libnls, 248
network services library — libnsl, 249
nl_types — native language data types, 362

O

obsolete RPC library — librpcsoc, 286

P

PAM (Pluggable Authentication Module) library — libpam, 260
panels library — libpanel, 262
password structure — pwd.h, 366
password structure — pwd, 366
pathname pattern-matching types — glob.h, 59
pathname pattern-matching types — glob, 59
PICL library — libpicl, 264
PICL plug-in library — libpicltree, 265
PKCS#11 Cryptographic Framework library — libpkcs11, 266
poll — definitions for the `poll()` function, 363
poll.h — definitions for the `poll()` function, 363
pool configuration manipulation library — libpool, 271
POSIX.1b Realtime Extensions library — libposix4, 290
POSIX.1b Realtime Extensions library — librt, 290
POSIX threads library — libpthread, 280
primitive system data types — types.h, 437
primitive system data types — types, 437
process accounting, — acct, 30
processes
  base signals — signal, 383
  signal generation information — siginfo, 379
product install registry library — libwsreg, 333
pthread — threads, 364
pthread.h — threads, 364
pwd — password structure, 366
pwd.h — password structure, 366

R

regex — regular expression matching types, 367
regex.h — regular expression matching types, 367
regular expression matching types — regex.h, 367
regular expression matching types — regex, 367
remote asynchronous calls library — librac, 283
remote shared memory interface library — librsm, 288
resolver library — libresolv, 284
resource — definitions for resource operations, 369
resource.h — definitions for resource operations, 369
RPC services library — librpcsvc, 287
runtime linker debugging library — librtld_db, 292

S

screen handling and optimization library
  — libcurses, 118
  — libtermcap, 118
  — libtermlib, 118
search — search tables, 372

search tables — search.h, 372
search tables — search, 372
search.h — search tables, 372
security attributes database library — libsecdb, 301
select — select types, 373
select types — select.h, 373
select types — select, 373
select.h — select types, 373
sem — semaphore facility, 375
sem.h — semaphore facility, 375
semaphore — semaphores, 374
semaphore.h — semaphores, 374
semaphores — semaphore.h, 374
semaphores — semaphore, 374
sendfile library — libsendfile, 303
service configuration facility library — libscf, 295
service location protocol library — libslp, 304
setjmp — stack environment declarations, 377
setjmp.h — stack environment declarations, 377
shared memory facility — shm.h, 378
shared memory facility — shm, 378
shm — shared memory facility, 378
shm.h — shared memory facility, 378
signal — base signals, 383
signal generation information, — siginfo, 379
smartcard library — libsmartcard, 305
socket — Internet Protocol family, 390
sockets local interfaces — if.h, 62
sockets local interfaces — if, 62
sockets library — libsocket, 306
spawn — spawn.h, 394
spawn — spawn, 394
spawn.h — spawn, 394
stack environment declarations — setjmp.h, 377
stack environment declarations — setjmp, 377
standard buffered input/output — stdio.h, 407
standard buffered input/output — stdio, 407
standard library definitions — stdlib.h, 409
standard library definitions — stdlib, 409
standard symbolic constants and types — unistd, 444
standard type definitions — stddef.h, 399
standard type definitions — stddef, 399
stat — data returned by stat system call, 395
statvfs — VFS File System information structure, 397
statvfs.h — VFS File System information structure, 397
stdbool — boolean type and values, 398
stdbool.h — boolean type and values, 398
stddef — standard type definitions, 399
stddef.h — standard type definitions, 399
stdint — integer types, 400
stdint.h — integer types, 400
stdio — standard buffered input/output, 407
stdio.h — standard buffered input/output, 407
stdlib — standard library definitions, 409
stdlib.h — standard library definitions, 409
STREAMS interface (STREAMS) — stropts.h, 413
STREAMS interface (STREAMS) — stropts, 413
string — string operations, 411
string operations — string.h, 411
string operations — string, 411
string operations — strings.h, 412
string operations — strings, 412
string.h — string operations, 411
string pattern-matching library — libgen, 149
strings — string operations, 412
strings.h — string operations, 412
stropts — STREAMS interface (STREAMS), 413
stropts.h — STREAMS interface (STREAMS), 413
Sun Solstice Enterprise Agent Component library — libdmici, 138
Sun Solstice Enterprise Agent DMI library — libdmi, 137
Sun Solstice Enterprise Agent library — libssagent, 308
Sun Solstice Enterprise Agent Management library — libdmimi, 139
Sun Solstice Enterprise SNMP library — libssasnmp, 309
syslog — definitions for system error logging, 418
syslog.h — definitions for system error logging, 418
system error numbers — errno.h, 42
system error numbers — errno, 42
system name structure — utsname.h, 456
system name structure — utsname, 456
system calls, — stat, 395

system library — libsys, 310

T

tar — extended tar definitions, 420
tar.h — extended tar definitions, 420
tcp — definitions for the Internet Transmission Co ntrol Protocol (TCP), 422
tcp.h — definitions for the Internet Transmission Co ntrol Protocol (TCP), 422
termios — define values for termios, 423
termios.h — define values for termios, 423
tgmath — type-generic macros, 428
tgmath.h — type-generic macros, 428
threads — pthread.h, 364
threads — pthread, 364
threads library — libthread, 320
time — time types, 433
time types — time.h, 433
time types — time, 433
time.h — time types, 433
timeb — additional definitions for date and time, 432
timeb.h — additional definitions for date and time, 432
times — file access and modification times structure, 435
times.h — file access and modification times structure, 435
TNF probe control library — libtnfctl, 322
type-generic macros — tgmath.h, 428
type-generic macros — tgmath, 428
types — primitive system data types, 437
types.h — primitive system data types, 437
types32 — fixed-width data types, 436

U

the UCB compatibility library — libucb, 324
ucontext — user context, 440
ucontext.h — user context, 440
uio — definitions for vector I/O operations, 441
uio.h — definitions for vector I/O operations, 441
ulimit — ulimit commands, 442
ulimit commands — ulimit.h, 442
ulimit commands — ulimit, 442
ulimit.h — ulimit commands, 442
un — definitions for UNIX-domain sockets, 443
unistd — standard symbolic constants and types, 444
user accounting database definitions — utmpx.h, 455
user accounting database definitions — utmpx, 455
user context — ucontext.h, 440
user context — ucontext, 440
user mailbox lockfile management library — libmail, 182
utime — access and modification times structure, 454
utime.h — access and modification times structure, 454
utmpx — user accounting database definitions, 455
utmpx.h — user accounting database definitions, 455
utsname — system name structure, 456
utsname.h — system name structure, 456

V

values — machine-dependent values, 457
vector math library — libmvec, 247
verify program assertion — assert.h, 36
verify program assertion — assert, 36
VFS File System information structure — statvfs.h, 397
VFS File System information structure — statvfs, 397
volume management library — libvolmgt, 330

W

wait — wait status, 459
wait status — wait.h, 459
wait status — wait, 459
wait.h — wait status, 459
wchar — wide-character handling, 461
wchar.h — wide-character handling, 461

wctype — wide-character classification and
mapping utilities, 463
wctype.h — wide-character classification and
mapping utilities, 463
wide-character classification and mapping
utilities — wctype.h, 463
wide-character classification and mapping
utilities — wctype, 463
wide-character handling — wchar.h, 461
wide-character handling — wchar, 461
wide character library — libw, 331
word-expansion types — wordexp.h, 464
word-expansion types — wordexp, 464
wordexp — word-expansion types, 464
wordexp.h — word-expansion types, 464

X

X/Open Networking library — libxnet, 335
semaphore facility — sem, 375
XSI interprocess communication access
structure — ipc.h, 68
XSI interprocess communication access
structure — ipc, 68
XSI message queue structures — msg, 358
XSI semaphore facility — sem.h, 375

Y

yacc library — liby, 338