man pages section 5: Standards, Environments, and Macros

Sun Microsystems, Inc.
901 San Antonio Road
Palo Alto, CA 94303-4900
U.S.A.

Part No: 806-0634-10
February 2000

Contents

Preface 7

Intro(5) 13

ascii(5) 14

attributes(5) 16

architecture(5) 16

availability(5) 16

CSI(5) 16

stability(5) 16

MT-Level(5) 16

charmap(5) 24

environ(5) 27

eqnchar(5) 35

extensions(5) 36

filesystem(5) 37

fnmatch(5) 56

fns(5) 60

fns_dns(5) 62

fns_files(5) 65

fns_initial_context(5) 67

fns_nis+(5) 71

fns_nis(5) 73

fns_policies(5) 75

fns_references(5) 79

fns_x500(5) 83

formats(5) 86

iconv_1250(5) 92

iconv_1251(5) 98

iconv(5) 106

iconv_646(5) 112

iconv_852(5) 115

iconv_8859-1(5) 121

iconv_8859-2(5) 127

iconv_8859-5(5) 133

iconv_dhn(5) 141

iconv_koi8-r(5) 145

iconv_mac_cyr(5) 153

iconv_maz(5) 160

iconv_pc_cyr(5) 164

iconv_unicode(5) 169

isalist(5) 174

largefile(5) 176

lf64(5) 179

lfcompile(5) 185

lfcompile64(5) 188

locale(5) 190

man(5) 220

mansun(5) 225

me(5) 229

mm(5) 234

ms(5) 241

nfssec(5) 247

pam_dial_auth(5) 249

pam_krb5(5) 250

pam_ldap(5) 254

pam_rhosts_auth(5) 257

pam_roles(5) 258

pam_sample(5) 260

pam_smartcard(5) 262

pam_unix(5) 264

prof(5) 267

rbac(5) 268

regex(5) 271

regexp(5) 281

compile(5) 281

step(5) 281

advance(5) 281

SEAM(5) 289

sgml(5) 291

solbook(5) 291

smartcard(5) 296

standards(5) 298

ANSI(5) 298

C(5) 298

ISO(5) 298

POSIX(5) 298

POSIX.1(5) 298

POSIX.2(5) 298

SUS(5) 298

SUSv2(5) 298

SVID(5) 298

SVID3(5) 298

XNS(5) 298

XNS4(5) 298

XNS5(5) 298

XPG(5) 298

XPG3(5) 298

XPG4(5) 298

XPG4v2(5) 298

sticky(5) 303

term(5) 304

vgrindefs(5) 308

Index 310

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.

CHAPTER

Headers, Tables, and Macros

12

NAME

Intro – introduction to miscellany

DESCRIPTION

Among the topics presented in this section are:

Standards — The POSIX (IEEE) Standards and the X/Open Specifications are described on the `standards` page.

Environments — The user environment (`environ`), the subset of the user environment that depends on language and cultural conventions (`locale`), the large file compilation environment (`lfcompile`), and the transitional compilation environment (`lfcompile64`) are described.

Macros — The macros to format Reference Manual pages (`man` and `mansun`) as well as other text format macros (`me`, `mm`, and `ms`) are described.

Characters — Tables of character sets (`ascii`, `charmap`, `eqnchar`, and `iconv`), file format notation (`formats`), file name pattern matching (`fnmatch`), and regular expressions (`regex` and `regexp`) are presented.

FNS — Topics concerning the Federated Naming Service (`fns`, `fns_initial_context`, `fns_policies`, and `fns_references`) are discussed.

NAME

ascii – map of ASCII character set

SYNOPSIS

cat `/usr/pub/ascii`

DESCRIPTION

`/usr/pub/ascii` is a map of the ASCII character set, to be printed as needed. It contains octal and hexadecimal values for each character. While not included in that file, a chart of decimal values is also shown here.

```
                         Octal – Character

000 NUL  001 SOH  002 STX  003 ETX  004 EOT  005 ENQ  006 ACK  007 BEL
010 BS   011 HT   012 NL   013 VT   014 NP   015 CR   016 SO   117 SI
020 DLE  021 DC1  022 DC2  023 DC3  024 DC4  025 NAK  026 SYN  027 ETB
030 CAN  031 EM   032 SUB  033 ESC  034 FS   035 GS   036 RS   037 US
040 SP   041 !    042 "    043 #    044 $    045 %    046 &    047 ’
050 (    051 )    052 *    053 +    054 ,    055 –    056 .    057 /
060 0    061 1    062 2    063 3    064 4    065 5    066 6    067 7
070 8    071 9    072 :    073 ;    074 <    075 =    076 >    077 ?
100 @    101 A    102 B    103 C    104 D    105 E    106 F    107 G
110 H    111 I    112 J    113 K    114 L    115 M    116 N    117 O
120 P    121 Q    122 R    123 S    124 T    125 U    126 V    127 W
130 X    131 Y    132 Z    133 [    134 \    135 ]    136 ^    137 _
140 ‘    141 a    142 b    143 c    144 d    145 e    146 f    147 g
150 h    151 i    152 j    153 k    154 l    155 m    156 n    157 o
160 p    161 q    162 r    163 s    164 t    165 u    166 v    167 w
170 x    171 y    172 z    173 {    174 |    175 }    176 ~    177 DEL
```

```
                      Hexadecimal – Character

00 NUL   01 SOH   02 STX   03 ETX   04 EOT   05 ENQ   06 ACK  07 BEL
08 BS    09 HT    0A NL    0B VT    0C NP    0D CR    0E SO   0F SI
10 DLE   11 DC1   12 DC2   13 DC3   14 DC4   15 NAK   16 SYN  17 ETB
18 CAN   19 EM    1A SUB   1B ESC   1C FS    1D GS    1E RS   1F US
20 SP    21 !     22 "     23 #     24 $     25 %     26 &    27 ’
28 (     29 )     2A *     2B +     2C ,     2D –     2E .    2F /
30 0     31 1     32 2     33 3     34 4     35 5     36 6    37 7
38 8     39 9     3A :     3B ;     3C <     3D =     3E >    3F ?
40 @     41 A     42 B     43 C     44 D     45 E     46 F    47 G
48 H     49 I     4A J     4B K     4C L     4D M     4E N    4F O
50 P     51 Q     52 R     53 S     54 T     55 U     56 V    57 W
58 X     59 Y     5A Z     5B [     5C \     5D ]     5E ^    5F _
60 ‘     61 a     62 b     63 c     64 d     65 e     66 f    67 g
68 h     69 i     6A j     6B k     6C l     6D m     6E n    6F o
70 p     71 q     72 r     73 s     74 t     75 u     76 v    77 w
78 x     79 y     7A z     7B {     7C |     7D }     7E ~    7F DEL
```

```
                       Decimal – Character

  0 NUL    1 SOH    2 STX    3 ETX    4 EOT    5 ENQ    6 ACK    7 BEL
  8 BS     9 HT    10 NL    11 VT    12 NP    13 CR    14 SO    15 SI
 16 DLE   17 DC1   18 DC2   19 DC3   20 DC4   21 NAK   22 SYN   23 ETB
 24 CAN   25 EM    26 SUB   27 ESC   28 FS    29 GS    30 RS    31 US
```

32 SP	33 !	34 "	35 #	36 $	37 %	38 &	39 ’
40 (	41)	42 *	43 +	44 ,	45 –	46 .	47 /
48 0	49 1	50 2	51 3	52 4	53 5	54 6	55 7
56 8	57 9	58 :	59 ;	60 <	61 =	62 >	63 ?
64 @	65 A	66 B	67 C	68 D	69 E	70 F	71 G
72 H	73 I	74 J	75 K	76 L	77 M	78 N	79 O
80 P	81 Q	82 R	83 S	84 T	85 U	86 V	87 W
88 X	89 Y	90 Z	91 [	92 \	93]	94 ^	95 _
96 ‘	97 a	98 b	99 c	100 d	101 e	102 f	103 g
104 h	105 i	106 j	107 k	108 l	109 m	110 n	111 o
112 p	113 q	114 r	115 s	116 t	117 u	118 v	119 w
120 x	121 y	122 z	123 {	124 |	125 }	126 ~	127 DEL

FILES

`/usr/pub/ascii` On-line chart of octal and hexadecimal values for the ASCII character set.

NAME

attributes, architecture, availability, CSI, stability, MT-Level – characteristics of commands, utilities, and device drivers

DESCRIPTION

The ATTRIBUTES man page section contains a table (see below) defining attribute types and their corresponding values.

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Architecture	SPARC
Availability	SUNWcsu
CSI	Enabled
Interface Stability	Unstable
MT-Level	Safe

Architecture

Architecture defines processor or specific hardware. (See –p option of uname(1)). In some cases, it may indicate required adapters or peripherals.

Availability

This refers to the software package which contains the command or component being described on the man page. To be able to use the command, the indicated package must have been installed. For information on how to add a package see pkgadd(1M) .

Code Set Independence (CSI)

OS utilities and libraries which are free of dependencies on the properties of any code sets are said to have Code Set Independence (CSI). They have the attribute of being CSI enabled. This is in contrast to many commands and utilities in Solaris, for example, that work only with Extended Unix Codesets (EUC), an encoding method that allows concurrent support for up to four code sets and is commonly used to represent Asian character sets.

However, for practical reasons, this independence is not absolute. Certain assumptions are still applied to the current CSI implementation:

- File code is a superset of ASCII.
- In order to support multi-byte characters and NULL -terminated UNIX file names, the NULL and / (slash) characters cannot be part of any multi-byte characters.
- Only "stateless" file code encodings are supported. Stateless encoding avoids shift, locking shift, designation, invocation, and so forth, although single shift is not excluded.
- Process code (wchar_t values) is implementation dependent and can change over time or between implementations or between locales.
- Not every object in Solaris 2 and Solaris 7can have names composed of arbitrary characters. The names of the following objects must be composed of ASCII characters:

  - User names, group name, and passwords
  - System name
  - Names of printers and special devices
  - Names of terminals (`/dev/tty*`)
  - Process ID numbers
  - Message queues, semaphores, and shared memory labels.
  - The following may be composed of ISO Latin-1 or EUC characters:
    - File names
    - Directory names
    - Command names
    - Shell variables and environmental variable names
    - Mount points for file systems
    - NIS key names and domain names
- The names of NFS shared files should be composed of ASCII characters. Although files and directories may have names and contents composed of characters from non-ASCII code sets, using only the ASCII codeset allows NFS mounting across any machine, regardless of localization. For the commands and utilities that are CSI enabled, all can handle single-byte and multi-byte locales released in 2.6. For applications to get full support of internationalization services, dynamic binding has to be applied. Statically bound programs will only get support for C and POSIX locales.

Interface Stability

Sun often provides developers with early access to new technologies, which allows developers to evaluate with them as soon as possible. Unfortunately, new technologies are prone to changes and standardization often results in interface incompatibility from previous versions.

To make reasonable risk assessments, developers need to know how likely an interface is to change in future releases. To aid developers in making these assessments, interface stability information is included on some manual pages for commands, entry-points, and file formats.

The more stable interfaces can safely be used by nearly all applications, because Sun will endeavor to ensure that these continue to work in future minor releases. Applications that depend only on Standard and Stable interfaces should reliably continue to function correctly on future minor releases (but not necessarily on earlier major releases).

The less stable interfaces allow experimentation and prototyping, but should be used only with the understanding that they might change incompatibly or even be dropped or replaced with alternatives in future minor releases.

"Interfaces" that Sun does not document (for example, most kernel data structures and some symbols in system header files) may be implementation artifacts. Such internal interfaces are not only subject to incompatible change or removal, but we are unlikely to mention such a change in release notes.

Release Levels

Products are given release levels, as well as names, to aid compatibility discussions. Each release level may also include changes suitable for lower levels.

Release	**Version**	**Significance**
Major	x.0	Likely to contain major feature additions; adhere to different, possibly incompatible Standard revisions; and though unlikely, could change, drop, or replace Standard or Stable interfaces. Initial product releases are usually 1.0.
Minor	x.y	Compared to an x.0 or earlier release (y!=0), it's likely to contain: minor feature additions, compatible Standard and Stable interfaces, possibly incompatible Evolving interfaces, or likely incompatible Unstable interfaces.
Micro	x.y.z	Intended to be interface compatible with the previous release (z!=0), but likely to add bug fixes, performance enhancements, and support for additional hardware.

Classifications

The following table summarizes how stability level classifications relate to release level. The first column lists the Stability Level. The second column lists the Release Level for Incompatable Changes, and the third column lists other comments. For a complete discussion of individual classifications, see the appropriate subsection below.

Stability	**Release**	**Comments**
Standard	Major (x.0)	Actual or de facto.
Stable	Major (x.0)	Incompatibilities are exceptional.
Evolving	Minor (x.y)	Migration advice might accompany an incompatibility.
Unstable	Minor (x.y)	Experimental or transitional: incompatibilities are common.
Obsolete	Minor (x.y)	Deprecated interface: likely to be removed in a future minor release.

The interface stability levels described in this manual page apply to both source and binary interfaces unless otherwise stated. The stability level of each interface is unknown unless explicitly stated.

`Standard:` *organization_name, standard_name, version*

The documented command or function complies with the standard listed. Most of these interfaces are defined by a formal standard, and controlled by a standards organization. Changes will usually be made in accordance with approved changes to that standard. his stability level can also apply to interfaces that have been adopted (without a formal standard) by an "industry convention."

Support is provided for only the specified version(s) of a standard; support of later versions is not guaranteed. If the standards organization approves a non-upwards-compatible change to a Standard interface that Sun decides to support, we will announce a compatibility and migration strategy.

`Stable`

A Stable interface is a mature interface under Sun's control. Sun will try to avoid non-upwards-compatible changes to these interfaces, especially in minor or micro releases.

If support of a Stable interface must be discontinued, Sun will attempt to provide notification and the stability level changes to Obsolete.

`Evolving`

An Evolving interface may eventually become Standard or Stable but is still in transition.

Sun will make reasonable efforts to ensure compatibility with previous releases as it evolves. When non-upwards compatible changes become necessary, they will occur in minor and major releases; such changes will be avoided in micro releases whenever possible. If such a change is necessary, it will be documented in the release notes for the effected release, and when feasible, Sun will provide migration aids for binary compatibility and continued source development.

`Unstable`

An Unstable interface is provided to give developers early access to new or rapidly changing technology or as an interim solution to a problem for which a more stable solution is anticipated in the future.

For Unstable interfaces, Sun no claims about either source or binary compatibility from one minor release to another. Applications developed based on these interfaces may not work in future minor releases.

`Obsolete: Scheduled for removal after` *event*

An Obsolete interface is supported in the current release, but is scheduled to be removed in a future (minor) release. When support of an interface is to be discontinued, Sun will attempt to provide notification before discontinuing support. Use of an Obsolete interface may produce warning messages.

MT-Level

Libraries are classified into four categories which define their ability to support multiple threads. Manual pages containing routines that are of multiple or differing levels show this within their `NOTES` or `USAGE` section.

`Safe`

Safe is an attribute of code that can be called from a multithreaded application. The effect of calling into a Safe interface or a safe code segment is that the results are valid even when called by multiple threads. Often overlooked is the fact that the result of this Safe interface or safe code segment can have global consequences that affect all threads. For example, the action of opening or closing a file from one thread is visible by all the threads within a process. A multi-threaded application has the responsibility for using these interfaces in a safe manner, which is different from whether or not the interface is Safe. For example, a multi-threaded application that closes a file that is still in use by other threads within the application is not using the `close`(2) interface safely.

`Unsafe`

An Unsafe library contains global and static data that is not protected. It is not safe to use unless the application arranges for only one thread at time to execute within the library. Unsafe libraries may contain routines that are Safe; however, most of the library's routines are unsafe to call.

The following table contains reentrant counterparts for Unsafe functions. This table is subject to change by Sun.

Reentrant functions for libc:

Unsafe Function	Reentrant counterpart
ctime	ctime_r
localtime	localtime_r
asctime	asctime_r
gmtime	gmtime_r
ctermid	ctermid_r
getlogin	getlogin_r
rand	rand_r
readdir	readdir_r

Unsafe Function	Reentrant counterpart
strtok	strtok_r
tmpnam	tmpnam_r

MT-Safe

An MT-Safe library is fully prepared for multithreaded access. It protects its global and static data with locks, and can provide a reasonable amount of concurrency. Note that a library can be safe to use, but not MT-Safe. For example, surrounding an entire library with a monitor makes the library Safe, but it supports no concurrency so it is not considered MT-Safe. An MT-Safe library must permit a reasonable amount of concurrency. (This definition's purpose is to give precision to what is meant when a library is described as Safe. The definition of a Safe library does not specify if the library supports concurrency. The MT-Safe definition makes it clear that the library is Safe, and supports some concurrency. This clarifies the Safe definition, which can mean anything from being single threaded to being any degree of multithreaded.)

Async-Signal-Safe

Async-Signal-Safe refers to particular library routines that can be safely called from a signal handler. A thread that is executing an Async-Signal-Safe routine will not deadlock with itself if interrupted by a signal. Signals are only a problem for MT-Safe routines that acquire locks.

Signals are disabled when locks are acquired in Async-Signal-Safe routines. This prevents a signal handler that might acquire the same lock from being called. The list of Async-Signal-Safe functions includes:

_exit	access	aio_error
aio_return	aio_suspend	alarm
cfgetispeed	cfgetospeed	cfsetispeed
cfsetospeed	chdir	chmod
chown	clock_gettime	close
creat	dup	dup2
execle	execve	fcntl
fdatasync	fork	fstat
fsync	getegid	geteuid
getgid	getgroups	getpgrp
getpid	getppid	getuid

kill	link	lseek
mkdir	mkfifo	open
pathconf	pause	pipe
read	rename	rmdir
sem_post	sema_post	setgid
setpgid	setsid	setuid
sigaction	sigaddset	sigdelset
sigemptyset	sigfillset	sigismember
sigpending	sigprocmask	sigqueue
sigsuspend	sleep	stat
sysconf	tcdrain	tcflow
tcflush	tcgetattr	tcgetpgrp
tcsendbreak	tcsetattr	tcsetpgrp
thr_kill	thr_sigsetmask	time
timer_getoverrun	timer_gettime	timer_settime
times	umask	uname
unlink	utime	wait
waitpid	write	

MT-Safe with Exceptions
: See the NOTES or USAGE sections of these pages for a description of the exceptions.

Safe with Exceptions
: See the NOTES or USAGE sections of these pages for a description of the exceptions.

Fork1-Safe
: A Fork1-Safe library releases the locks it had held whenever fork1(2) is called in a Solaris thread program, or fork(2) in a POSIX (see standards(5)) thread program. Calling fork(2) in a POSIX thread program has the same semantic as calling fork1(2) in a Solaris thread program. All system calls, libpthread , and libthread are Fork1-Safe . Otherwise, you should handle the locking clean-up yourself (see pthread_atfork(3THR)).

Cancel-Safety

If a multi-threaded application uses pthread_cancel(3THR) to cancel (that is, kill) a thread, it is possible that the target thread is killed while holding a resource, such as a lock or allocated memory. If the thread has not installed the appropriate cancellation cleanup handlers to release the resources appropriately (see pthread_cancel(3THR)), the application is "cancel-unsafe", that is, it is not safe with respect to cancellation. This unsafety could result in deadlocks due to locks not released by a thread that gets cancelled, or resource leaks; for example, memory not being freed on thread cancellation. All applications that use pthread_cancel(3THR) should ensure that they operate in a Cancel-Safe environment. Libraries that have cancellation points and which acquire resources such as locks or allocate memory dynamically, also contribute to the cancel-unsafety of applications that are linked with these libraries. This introduces another level of safety for libraries in a multi-threaded program: Cancel-Safety. There are two sub-categories of Cancel-Safety: Deferred-Cancel-Safety, and Asynchronous-Cancel-Safety. An application is considered to be Deferred-Cancel-Safe when it is Cancel-Safe for threads whose cancellation type is PTHREAD_CANCEL_DEFERRED . An application is considered to be Asynchronous-Cancel-Safe when it is Cancel-Safe for threads whose cancellation type is PTHREAD_CANCEL_ASYNCHRONOUS . Deferred-Cancel-Safety is easier to achieve than Asynchronous-Cancel-Safety, since a thread with the deferred cancellation type can be cancelled only at well-defined cancellation points, whereas a thread with the asynchronous cancellation type can be cancelled anywhere. Since all threads are created by default to have the deferred cancellation type, it may never be necessary to worry about asynchronous cancel safety. Indeed, most applications and libraries are expected to always be Asynchronous-Cancel-Unsafe. An application which is Asynchronous-Cancel-Safe is also, by definition, Deferred-Cancel-Safe.

SEE ALSO

uname(1) , pkgadd(1M) , Intro(3) , standards(5)

NAME

charmap – character set description file

DESCRIPTION

A character set description file or *charmap* defines characteristics for a coded character set. Other information about the coded character set may also be in the file. Coded character set character values are defined using symbolic character names followed by character encoding values.

The character set description file provides:

- The capability to describe character set attributes (such as collation order or character classes) independent of character set encoding, and using only the characters in the portable character set. This makes it possible to create generic `localedef`(1) source files for all codesets that share the portable character set.
- Standardized symbolic names for all characters in the portable character set, making it possible to refer to any such character regardless of encoding.

Symbolic Names

Each symbolic name is included in the file and is mapped to a unique encoding value (except for those symbolic names that are shown with identical glyphs). If the control characters commonly associated with the symbolic names in the following table are supported by the implementation, the symbolic names and their corresponding encoding values are included in the file. Some of the encodings associated with the symbolic names in this table may be the same as characters in the portable character set table.

<ACK>	<DC2>	<ENQ>	<FS>	<IS4>	<SOH>
<BEL>	<DC3>	<EOT>	<GS>	<LF>	<STX>
<BS>	<DC4>	<ESC>	<HT>	<NAK>	<SUB>
<CAN>	<DEL>	<ETB>	<IS1>	<RS>	<SYN>
<CR>	<DLE>	<ETX>	<IS2>	<SI>	<US>
<DC1>	<EM>	<FF>	<IS3>	<SO>	<VT>

Declarations

The following declarations can precede the character definitions. Each must consist of the symbol shown in the following list, starting in column 1, including the surrounding brackets, followed by one or more blank characters, followed by the value to be assigned to the symbol.

<code_set_name> The name of the coded character set for which the character set description file is defined.

<mb_cur_max> The maximum number of bytes in a multi-byte character. This defaults to 1.

<*mb_cur_min*> — An unsigned positive integer value that defines the minimum number of bytes in a character for the encoded character set.

<*escape_char*> — The escape character used to indicate that the characters following will be interpreted in a special way, as defined later in this section. This defaults to backslash (`\thinsp;`), which is the character glyph used in all the following text and examples, unless otherwise noted.

<*comment_char*> — The character that when placed in column 1 of a charmap line, is used to indicate that the line is to be ignored. The default character is the number sign (#).

Format

The character set mapping definitions will be all the lines immediately following an identifier line containing the string `CHARMAP` starting in column 1, and preceding a trailer line containing the string `END CHARMAP` starting in column 1. Empty lines and lines containing a <*comment_char*> in the first column will be ignored. Each non-comment line of the character set mapping definition (that is, between the `CHARMAP` and `END CHARMAP` lines of the file) must be in either of two forms:

`"%s %s %s\n",`<*symbolic-name*>,<*encoding*>,<*comments*>

or

`"%s. . .%s %s %s\n",`<*symbolic-name*>,<*symbolic-name*>, <*encoding*>,<*comments*>

In the first format, the line in the character set mapping definition defines a single symbolic name and a corresponding encoding. A character following an escape character is interpreted as itself; for example, the sequence <`\i\`> represents the symbolic name `\` enclosed between angle brackets.

In the second format, the line in the character set mapping definition defines a range of one or more symbolic names. In this form, the symbolic names must consist of zero or more non-numeric characters, followed by an integer formed by one or more decimal digits. The characters preceding the integer must be identical in the two symbolic names, and the integer formed by the digits in the second symbolic name must be equal to or greater than the integer formed by the digits in the first name. This is interpreted as a series of symbolic names formed from the common part and each of the integers between the first and the second integer, inclusive. As an example, `<j0101>. . .<j0104>` is interpreted as the symbolic names `<j0101>`, `<j0102>`, `<j0103>`, and `<j0104>`, in that order.

A character set mapping definition line must exist for all symbolic names and must define the coded character value that corresponds to the character glyph indicated in the table, or the coded character value that corresponds with the control character symbolic name. If the control characters commonly associated with the symbolic names are supported by the implementation, the symbolic name and the corresponding encoding value must be included in the file. Additional unique symbolic names may be included. A coded character value can be represented by more than one symbolic name.

The encoding part is expressed as one (for single-byte character values) or more concatenated decimal, octal or hexadecimal constants in the following formats:

`"%cd%d",` <*escape_char*>,<*decimal byte value*>
`"%cx%x",` <*escape_char*>,<*hexadecimal byte value*>
`"%c%o",` <*escape_char*>,<*octal byte value*>

Decimal Constants

Decimal constants must be represented by two or three decimal digits, preceded by the escape character and the lower-case letter `d`; for example, `\d05`, `\d97`, or `\d143`. Hexadecimal constants must be represented by two hexadecimal digits, preceded by the escape character and the lower-case letter `x`; for example, `\x05`, `\x61`, or `\x8f`. Octal constants must be represented by two or three octal digits, preceded by the escape character; for example, `\05`, `\141`, or `\217`. In a portable charmap file, each constant must represent an 8-bit byte. Implementations supporting other byte sizes may allow constants to represent values larger than those that can be represented in 8-bit bytes, and to allow additional digits in constants. When constants are concatenated for multi-byte character values, they must be of the same type, and interpreted in byte order from first to last with the least significant byte of the multi-byte character specified by the last constant.

Ranges of Symbolic Names

In lines defining ranges of symbolic names, the encoded value is the value for the first symbolic name in the range (the symbolic name preceding the ellipsis). Subsequent symbolic names defined by the range will have encoding values in increasing order. For example, the line

```
<j0101>. . .<j0104>     \d129\d254
```

will be interpreted as:

```
<j0101>                \d129\d254
<j0102>                \d129\d255
<j0103>                \d130\d0
<j0104>                \d130\d1
```

Note that this line will be interpreted as the example even on systems with bytes larger than 8 bits. The comment is optional.

SEE ALSO

`locale`(1) `localedef`(1) `nl_langinfo`(3C) `extensions`(5), `locale`(5)

NAME

environ – user environment

DESCRIPTION

When a process begins execution, one of the `exec` family of functions makes available an array of strings called the environment; see `exec`(2). By convention, these strings have the form *variable=value*, for example, `PATH=/sbin:/usr/sbin`. These environmental variables provide a way to make information about a program's environment available to programs.

A name may be placed in the environment by the `export` command and *name=value* arguments in `sh`(1), or by one of the `exec` functions. It is unwise to conflict with certain shell variables such as `MAIL`, `PS1`, `PS2`, and `IFS` that are frequently exported by `.profile` files; see `profile`(4).

The following environmental variables can be used by applications and are expected to be set in the target run-time environment.

`HOME` The name of the user's login directory, set by `login`(1) from the password file; see `passwd`(4).

`LANG` The string used to specify internationalization information that allows users to work with different national conventions. The `setlocale`(3C) function checks the `LANG` environment variable when it is called with `""` as the `locale` argument. `LANG` is used as the default locale if the corresponding environment variable for a particular category is unset or null. If, however, `LC_ALL` is set to a valid, non-empty value, its contents are used to override both the `LANG` and the other `LC_*` variables. For example, when invoked as

```
setlocale(LC_CTYPE, ""),
```

`setlocale( )` will query the `LC_CTYPE` environment variable first to see if it is set and non-null. If `LC_CTYPE` is not set or null, then `setlocale( )` will check the `LANG` environment variable to see if it is set and non-null. If both `LANG` and `LC_CTYPE` are unset or `NULL`, the default "C" locale will be used to set the `LC_CTYPE` category.

Most commands will invoke

setlocale(LC_ALL, "")

prior to any other processing. This allows the command to be used with different national conventions by setting the appropriate environment variables.

The following environment variables correspond to each category of `setlocale`(3C):

`LC_ALL`
: If set to a valid, non-empty string value, override the values of `LANG` and all the other `LC_*`variables.

`LC_COLLATE`
: This category specifies the character collation sequence being used. The information corresponding to this category is stored in a database created by the `localedef`(1) command. This environment variable affects `strcoll`(3C) and `strxfrm`(3C).

`LC_CTYPE`
: This category specifies character classification, character conversion, and widths of multibyte characters. When `LC_CTYPE` is set to a valid value, the calling utility can display and handle text and file names containing valid characters for that locale; Extended Unix Code (EUC) characters where any individual character can be 1, 2, or 3 bytes wide; and EUC characters of 1, 2, or 3 column widths. The default "C" locale corresponds to the 7-bit ASCII character set; only characters from ISO 8859-1 are valid. The information corresponding to this category is stored in a database created by the `localedef( )` command. This environment variable is used by `ctype`(3C), `mblen`(3C), and many commands, such as `cat`(1), `ed`(1), `ls`(1), and `vi`(1).

`LC_MESSAGES`	This category specifies the language of the message database being used. For example, an application may have one message database with French messages, and another database with German messages. Message databases are created by the `mkmsgs`(1) command. This environment variable is used by `exstr`(1), `gettxt`(1), `srchtxt`(1), `gettxt`(3C), and `gettext`(3C).
`LC_MONETARY`	This category specifies the monetary symbols and delimiters used for a particular locale. The information corresponding to this category is stored in a database created by the `localedef`(1) command. This environment variable is used by `localeconv`(3C).
`LC_NUMERIC`	This category specifies the decimal and thousands delimiters. The information corresponding to this category is stored in a database created by the `localedef()` command. The default `C` locale corresponds to `"."` as the decimal delimiter and no thousands delimiter. This environment variable is used by `localeconv`(3C), `printf`(3C), and `strtod`(3C).
`LC_TIME`	This category specifies date and time formats. The information corresponding to this category is stored in a database specified in `localedef()`. The

default `C` locale corresponds to U.S. date and time formats. This environment variable is used by many commands and functions; for example: `at`(1), `calendar`(1), `date`(1), `strftime`(3C), and `getdate`(3C).

`MSGVERB` Controls which standard format message components `fmtmsg` selects when messages are displayed to `stderr`; see `fmtmsg`(1) and `fmtmsg`(3C).

`NETPATH` A colon-separated list of network identifiers. A network identifier is a character string used by the Network Selection component of the system to provide application-specific default network search paths. A network identifier must consist of non-null characters and must have a length of at least 1. No maximum length is specified. Network identifiers are normally chosen by the system administrator. A network identifier is also the first field in any `/etc/netconfig` file entry. `NETPATH` thus provides a link into the `/etc/netconfig` file and the information about a network contained in that network's entry. `/etc/netconfig` is maintained by the system administrator. The library routines described in `getnetpath`(3NSL) access the `NETPATH` environment variable.

`NLSPATH` Contains a sequence of templates which `catopen`(3C) and `gettext`(3C) use when attempting to locate message catalogs. Each template consists of an optional prefix, one or more substitution fields, a filename and an optional suffix. For example:

```
NLSPATH="/system/nlslib/%N.cat"
```

defines that `catopen( )` should look for all message catalogs in the directory `/system/nlslib`, where the catalog name should be constructed from the *name* parameter passed to `catopen`( ), `%N`, with the suffix `.cat`.

Substitution fields consist of a `%` symbol, followed by a single-letter keyword. The following keywords are currently defined:

%N The value of the *name* parameter passed to `catopen( )`.

%L The value of `LANG` or `LC_MESSAGES`.

%l The language element from `LANG` or `LC_MESSAGES`.

%t The territory element from `LANG` or `LC_MESSAGES`.

%c The codeset element from `LANG` or `LC_MESSAGES`.

%% A single `%` character.

An empty string is substituted if the specified value is not currently defined. The separators “`_`” and “`.`” are not included in `%t` and `%c` substitutions.

Templates defined in `NLSPATH` are separated by colons (`:`). A leading colon or two adjacent colons (`::`) is equivalent to specifying `%N`. For example:

NLSPATH=":%N.cat:/nlslib/%L/%N.cat"

indicates to `catopen()` that it should look for the requested message catalog in *name*, *name*`.cat` and `/nlslib/$LANG/`*name*.cat. For `gettext( )`, `%N` automatically maps to "messages".

If `NLSPATH` is unset or `NULL`, `catopen( )` and `gettext( )` call `setlocale`(3C), which checks `LANG` and the `LC_*` variables to locate the message catalogs.

`NLSPATH` will normally be set up on a system wide basis (in `/etc/profile`) and thus makes the location and naming conventions associated with message catalogs transparent to both programs and users.

`PATH` The sequence of directory prefixes that `sh`(1), `time`(1), `nice`(1), `nohup`(1), and other utilities apply in searching for a file known by an incomplete path name. The prefixes are separated by colons (`:`). `login`(1) sets `PATH=/usr/bin`. For more detail, see `sh`(1).

`SEV_LEVEL` Define severity levels and associate and print strings with them in standard format error messages; see `addseverity`(3C), `fmtmsg`(1), and `fmtmsg`(3C).

`TERM`
: The kind of terminal for which output is to be prepared. This information is used by commands, such as `vi`(1), which may exploit special capabilities of that terminal.

`TZ`
: Timezone information. The contents of this environment variable are used by the functions `ctime`(3C), `localtime`(3C), `strftime`(3C), and `mktime`(3C) to override the default timezone. If `TZ` is not in the following form, it designates a path to a timezone database file relative to `/usr/share/lib/zoneinfo/`, ignoring the first character if it is a colon (`:`); otherwise, `TZ` has the form:

std offset [dst [offset], [start [/ time], end [/ time]]]

std and *dst*
: Three or more bytes that are the designation for the standard (*std*) and daylight savings time (*dst*) timezones. Only *std* is required. If *dst* is missing, then daylight savings time does not apply in this locale. Upper- and lower-case letters are allowed. Any characters except a leading colon (`:`), digits, a comma (`,`), a minus (–) or a plus (+) are allowed.

offset
: Indicates the value one must add to the local time to arrive at Coordinated Universal Time. The offset has the form:

hh [: mm [: ss]]

The minutes (*mm*) and seconds (*ss*) are optional. The hour (*hh*) is required and may be a single digit. The *offset* following *std* is required. If no *offset* follows *dst* , daylight savings time is assumed to be one hour ahead of standard time. One or more digits may be used; the value is always interpreted as a decimal number. The hour must be between 0 and 24, and the minutes (and seconds) if present between 0 and 59. Out of range values may cause unpredictable behavior. If preceded by a “–” the timezone is east of the Prime Meridian; otherwise it is west (which may be indicated by an optional preceding “+” sign).

start / time, end / time

Indicate when to change to and back from daylight savings time, where *start/time* describes when the change from standard time to daylight savings time occurs, and *end/time* describes when the change back happens. Each `time` field describes when, in current local time, the change is made.

The formats of *start* and *end* are one of the following:

`J`*n*

The Julian day *n* ($1 \leq n \leq 365$). Leap days are not counted. That is, in all years, February 28 is day 59 and March 1 is day 60. It is impossible to refer to the occasional February 29.

n

The zero-based Julian day ($0 \leq n \leq 365$). Leap days are counted, and it is possible to refer to February 29.

M*m.n.d* — The d^{th} day, ($0 \le d \le 6$) of week *n* of month *m* of the year ($1 \le n \le 5$, $1 \le m \le 12$), where week 5 means “the last *d*-day in month *m*” which may occur in either the fourth or the fifth week). Week 1 is the first week in which the d^{th} day occurs. Day zero is Sunday.

Implementation specific defaults are used for *start* and *end* if these optional fields are not given.

The `time` has the same format as *offset* except that no leading sign (“–” or “+” is allowed. The default, if `time` is not given is 02:00:00.

SEE ALSO

`cat`(1), `date`(1), `ed`(1), `fmtmsg`(1), `localedef`(1), `login`(1), `ls`(1), `mkmsgs`(1), `nice`(1), `nohup`(1), `sh`(1), `sort`(1), `time`(1), `vi`(1), `exec`(2), `addseverity`(3C), `catopen`(3C), `ctime`(3C), `ctype`(3C), `fmtmsg`(3C), `getdate`(3C), `getnetpath`(3NSL), `gettext`(3C), `gettxt`(3C), `localeconv`(3C), `mblen`(3C), `mktime`(3C), `printf`(3C), `setlocale`(3C), `strcoll`(3C), `strftime`(3C), `strtod`(3C), `strxfrm`(3C), `TIMEZONE`(4), `netconfig`(4), `passwd`(4), `profile`(4)

NAME

eqnchar – special character definitions for eqn

SYNOPSIS

eqn /usr/share/lib/pub/eqnchar *filename* | troff *options*

neqn /usr/share/lib/pub/eqnchar *filename* | troff *options*

DESCRIPTION

The `eqnchar` command contains `nroff`(1) and `troff`(1) character definitions for constructing characters that are not available on the Graphic Systems typesetter. These definitions are primarily intended for use with `eqn`(1) and `neqn`(1). It contains definitions for the following characters:

ciplus	⊕	*||*	‖	*square*	□
citimes	⊗	*langle*	⟨	*circle*	○
wig	~	*rangle*	⟩	*blot*	■
-wig	≃	*hbar*	ℏ	*bullet*	•
>wig	≳	*ppd*	⊥	*prop*	∝
<wig	≲	*<->*	↔	*empty*	∅
=wig	≅	*<=>*	⇔	*member*	∈
star	∗	*|<*	≮	*nomem*	∉
bigstar	✱	*|>*	≯	*cup*	∪
=dot	≐	*ang*	∠	*cap*	∩
orsign	∨	*rang*	∟	*incl*	⊑
andsign	∧	*3dot*	⋮	*subset*	⊂
=del	≜	*thf*	∴	*supset*	⊃
oppA	∀	*quarter*	¼	*!subset*	⊆
oppE	∃	*3quarter*	¾	*!supset*	⊇
angstrom	Å	*degree*	°		

FILES

`/usr/share/lib/pub/eqnchar`

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWdoc

SEE ALSO

`eqn`(1), `nroff`(1), `troff`(1), `attributes`(5)

NAME

extensions – localedef extensions description file

DESCRIPTION

A localedef extensions description file or *extensions* file defines various extensions for the `localedef`(1) command.

The localedef extensions description file provides:

- EUC code set width information via the `cswidth` keyword: `cswidth bc1 : sw1,bc2 : sw2,bc3 : sw3`where `bc1`, `bc2`, and `bc3` indicate the number of bytes (byte count) per character for EUC codesets `1`, `2`, and `3`, respectively. `sw1`, `sw2`, and `sw3` indicate screen width for EUC codesets `1`, `2`, and `3`, respectively.
- Other extensions which will be documented in a future release.

SEE ALSO

`locale`(1), `localedef`(1), `environ`(5), `locale`(5)

NAME

filesystem – file system organization

SYNOPSIS

/

/usr

DESCRIPTION

The file system tree is organized for administrative convenience. Distinct areas within the file system tree are provided for files that are private to one machine, files that can be shared by multiple machines of a common platform, files that can be shared by all machines, and home directories. This organization allows sharable files to be stored on one machine but accessed by many machines using a remote file access mechanism such as NFS. Grouping together similar files makes the file system tree easier to upgrade and manage.

The file system tree consists of a root file system and a collection of mountable file systems. The `mount`(2) program attaches mountable file systems to the file system tree at mount points (directory entries) in the root file system or other previously mounted file systems. Two file systems, `/` (the root) and `/usr`, must be mounted in order to have a completely functional system. The root file system is mounted automatically by the kernel at boot time; the `/usr` file system is mounted by the system start-up script, which is run as part of the booting process.

Certain locations, noted below, are approved installation locations for bundled Foundation Solaris software. In some cases, the approved locations for bundled software are also approved locations for add-on system software or for applications. The following descriptions make clear where the two locations differ. For example, `/etc` is the installation location for platform-dependent configuration files that are bundled with Solaris software. The analogous location for applications is `/etc/opt/`*packagename*.

In the following descriptions, *subsystem* is a category of application or system software, such as a window system (`dt`) or a language (`java1.2`)

The following descriptions make use of the terms *platform*, *platform-dependent*, *platform-independent*, and *platform-specific*. Platform refers to a machines Instruction Set Architecture or processor type, such as is returned by `uname –i`. *Platform-dependent* refers to a file that is installed on all platforms and whose contents vary depending on the platform. Like a platform-dependent file, a *platform-independent* file is installed on all platforms. However, the contents of the latter type remains the same on all platforms. An example of a platform-dependent file is compiled, executable program. An example of a platform-independent file is a standard configuration file, such as `/etc/hosts`. Unlike a platform-dependent or a platform-independent file, the *platform-specific* file is installed only on a subset of supported platforms. Most platform-specific files are gathered under `/platform` and `/usr/platform`.

Root File System

The root file system contains files that are unique to each machine. It contains the following directories:

`/dev`
Primary location for special files. Typically, device files are built to match the kernel and hardware configuration of the machine.

`/dev/cfg`
Symbolic links to physical `ap_ids`.

`/dev/cua`
Device files for `uucp`.

`/dev/dsk`
Block disk devices.

`/dev/fbs`
Frame buffer device files.

`/dev/fd`
File descriptors.

`/dev/md`
Logical volume management meta-disk devices.

`/dev/pts`
Pseudo-terminal devices.

`/dev/rdsk`
Raw disk devices.

`/dev/rmt`
Raw tape devices.

`/dev/sad`
Entry points for the STREAMS Administrative driver.

`/dev/sound`
Audio device and audio device control files.

`/dev/swap`
Default swap device.

`/dev/term`
Terminal devices.

`/etc`
Platform-dependent administrative and configuration files and databases that are not shared among systems. `/etc` may be viewed as the directory that defines the machine's identity. An approved installation location

for bundled Solaris software. The analogous location for add-on system software or for applications is `/etc/opt/`*packagename*.

`/etc/acct`
Accounting system configuration information.

`/etc/cron.d`
Configuration information for `cron`(1M).

`/etc/default`
Defaults information for various programs.

`/etc/dfs`
Configuration information for shared file systems.

`/etc/dhcp`
Dynamic Host Configuration Protocol (DHCP) configuration files.

`/etc/dmi`
Solstice Enterprise Agents configuration files.

`/etc/fn`
Federated Naming Service and X.500 support files.

`/etc/fs`
Binaries organized by file system types for operations required before `/usr` is mounted.

`/etc/gss`
Generic Security Service (GSS) Application Program Interface configuration files.

`/etc/inet`
Configuration files for Internet services.

`/etc/init.d`
Shell scripts for transitioning between run levels.

`/etc/lib`
Shared libraries needed during booting.

`/etc/lp`
Configuration information for the printer subsystem.

`/etc/llc2`
Logical link control (`llc2`) driver configuration files.

`/etc/lp`
Configuration information for the printer subsystem.

`/etc/mail`

Mail subsystem configuration.

`/etc/net`
Configuration information for transport independent network services.

`/etc/nfs`
NFS server logging configuration file.

`/etc/opt`
Configuration information for optional packages.

`/etc/openwin`
OpenWindows configuration files.

`/etc/rc0.d`
Scripts for entering or leaving run level 0. See `init`(1M).

`/etc/rc1.d`
Scripts for entering or leaving run level 1. See `init`(1M).

`/etc/rc2.d`
Scripts for entering or leaving run level 2. See `init`(1M).

`/etc/rc3.d`
Scripts for entering or leaving run level 3. See `init`(1M).

`/etc/rcS.d`
Scripts for bringing the system up in single user mode.

`/etc/rpcsec`
This directory might contain an NIS+ authentication configuration file.

`/etc/saf`
Service Access Facility files.

`/etc/security`
Basic Security Module (BSM) configuration files.

`/etc/skel`
Default profile scripts for new user accounts. See `useradd`(1M).

`/etc/`*subsystem*
Platform-dependent *subsystem* configuration files that are not shared among systems. An approved installation location for bundled Solaris software. The analogous location for add-on system software or for applications is `/etc/opt/`*packagename*.

`/etc/tm`
Trademark files; contents displayed at boot time.

`/etc/uucp`

UUCP configuration information. See uucp(1C).

/etc/volatile
Non-permanent, temporary files created by the system. An approved installation location for bundled Solaris software and for add-on system software.

/export
Default root of the shared file system tree.

/home
Default root of a subtree for user directories.

/kernel
Subtree of platform-dependent loadable kernel modules required as part of the boot process. It includes the generic part of the core kernel that is platform-independent, /kernel/genunix. See kernel(1M) An approved installation location for bundled Solaris software and for add-on system software.

/kernel/drv
32-bit device drivers.

/kernel/drv/sparcv9
64-bit SPARC device drivers.

/kernel/genunix
Platform-independent kernel.

/kernel/*subsystem*/ia64
64-bit Intel IA64 platform-dependent modules required for boot. An approved installation location for bundled Solaris software and for add-on system software. Note that ia64 is an example name; the actual name might be different.

/kernel/*subsystem*/sparcv9
64-bit SPARC platform-dependent modules required for boot. An approved installation location for bundled Solaris software and for add-on system software.

/mnt
Default temporary mount point for file systems. This is an empty directory on which file systems can be temporarily mounted.

/opt
Root of a subtree for add-on application packages.

/platform

Subtree of platform-specific objects which need to reside on the root filesystem. It contains a series of directories, one per supported platform. The semantics of the series of directories is equivalent to `/` (root).

`/platform/‘uname –i‘/kernel`
Platform-specific modules required for boot. These modules have semantics equivalent to `/kernel`. It includes the file `unix`, the core kernel. See `kernel`(1M). An approved installation location for bundled Solaris software and for add-on system software.

`/platform/‘uname –m‘/kernel`
Hardware class-specific modules required for boot. An approved installation location for bundled Solaris software and for add-on system software.

`/platform/‘uname –i‘/kernel/`*subsystem*`/ia64`
Intel 64-bit, platform-dependent modules required for boot. Note that `ia64` is an example name; the actual name might be different. An approved installation location for bundled Solaris software.

`/platform/‘uname –i‘/kernel/`*subsystem*`/sparcv9`
SPARC 64-bit platform-specific modules required for boot. An approved installation location for bundled Solaris software.

`/platform/‘uname –i‘/kernel/sparcv9/unix`
64-bit platform-dependent kernel.

`/platform/‘uname –i‘/kernel/unix`
32-bit platform-dependent kernel.

`/platform/‘uname –i‘/lib`
Platform-specific shared objects required for boot. Semantics are equivalent to `/lib`. An approved installation location for bundled Solaris software and for add-on system software.

`/platform/‘uname –i‘/sbin`
Platform-specific administrative utilities required for boot. Semantics are equivalent to `/sbin`. An approved installation location for bundled Solaris software and for add-on system software.

`/proc`
Root of a subtree for the process file system.

`/sbin`
Essential executables used in the booting process and in manual system recovery. The full complement of utilities is available only after `/usr` is mounted. `/sbin` is an approved installation location for bundled Solaris software.

`/tmp`

Temporary files; cleared during the boot operation.

`/usr`
Mount point for the `/usr` file system. See description of `/usr` file system, below.

`/var`
Root of a subtree for varying files. Varying files are files that are unique to a machine but that can grow to an arbitrary (that is, variable) size. An example is a log file. An approved installation location for bundled Solaris software. The analogous location for add-on system software or for applications is `/var/opt/`*packagename*.

`/var/adm`
System logging and accounting files.

`/var/audit`
Basic Security Module (BSM) audit files.

`/var/crash`
Default depository for kernel crash dumps.

`/var/cron`
Log files for `cron`(1M).

`/var/dmi`
Solstice Enterprise Agents (SEA) Desktop Management Interface (DMI) run-time components.

`/var/dt`
`dtlogin` configuration files.

`/var/ftp`
FTP server directory.

`/var/inet`
IPv6 router state files.

`/var/log`
System log files.

`/var/lp`
Line printer subsystem logging information.

`/var/mail`
Directory where users' mail is kept.

`/var/news`
Community service messages. This is not the same as USENET-style news.

`/var/nfs`

NFS server log files.

`/var/nis`
NIS+ databases.

`/var/ntp`
Network Time Protocol (NTP) server state directory.

`/var/opt`
Root of a subtree for varying files associated with optional software packages. An approved installation location for add-on system software and applications.

`/var/preserve`
Backup files for `vi`(1) and `ex`(1).

`/var/run`
Temporary files which are not needed across reboots. Only root may modify the contents of this directory.

`/var/sadm`
Databases maintained by the software package management utilities.

`/var/sadm/system/logs`
Status log files produced by software management functions and/or applications. For example, log files produced for product installation. An approved installation location for bundled Solaris software and for add-on system software and applications.

`/var/saf`
Service access facility logging and accounting files.

`/var/spool`
Contains directories for files used in printer spooling, mail delivery, `cron`(1M), `at`(1), and so forth.

`/var/spool/cron`
`cron`(1M) and `at`(1) spooling files.

`/var/spool/locks`
Spooling lock files.

`/var/spool/lp`
Line printer spool files. See `lp`(1).

`/var/spool/mqueue`
Mail queued for delivery.

`/var/spool/pkg`
Spooled packages.

`/var/spool/uucp`
Queued `uucp`(1C) jobs.

`/var/spool/uucppublic`
Files deposited by `uucp`(1C).

`/var/statmon`
Network status monitor files.

`/var/tmp`
Files that vary in size or presence during normal system operations. This directory is *not* cleared during the boot operation. An approved installation location for bundled Solaris software and for add-on system software and applications.

`/var/uucp`
`uucp`(1C) log and status files.

`/var/yp`
Databases needed for backwards compatibility with NIS and `ypbind`(1M); unnecessary after full transition to NIS+.

/usr File System

Because it is desirable to keep the root file system small and not volatile, on disk-based systems larger file systems are often mounted on `/home`, `/opt`, `/usr`, and `/var`.

The file system mounted on `/usr` contains platform-dependent and platform-independent sharable files. The subtree rooted at `/usr/share` contains platform-independent sharable files; the rest of the `/usr` tree contains platform-dependent files. By mounting a common remote file system, a group of machines with a common platform may share a single `/usr` file system. A single `/usr/share` file system can be shared by machines of any platform. A machine acting as a file server can share many different `/usr` file systems to support several different architectures and operating system releases. Clients usually mount `/usr` read-only so that they do not accidentally change any shared files.

The `/usr` file system contains the following subdirectories:

`/usr/4lib`
`a.out` libraries for the Binary Compatibility Package. See the *Binary Compatibility Guide*.

`/usr/5bin`
Symbolic link to the `/usr/bin` directory.

`/usr/X`
Symbolic link to the `/usr/openwin` directory.

`/usr/adm`
Symbolic link to the `/var/adm` directory.

`/usr/aset`
Directory for Automated Security Enhancement Tools (ASET) programs and files.

`/usr/bin`
Platform-dependent, user-invoked executables. These are commands users expect to be run as part of their normal `$PATH`. For executables that are different on a 64–bit system than on a 32–bit system, a wrapper that selects the appropriate executable is placed here. See `isaexec`(3C). An approved installation location for bundled Solaris software. The analogous location for add-on system software or for applications is `/opt/`*packagename*`/bin`.

`/usr/bin/ia64`
Intel 64–bit, platform-dependent, user-invoked executables. Note that `ia64` is an example name; the actual name might be different. This directory should not be part of a user's `$PATH`. A wrapper in `/usr/bin` should invoke the executable in this directory. See `isaexec`(3C). An approved installation location for bundled Solaris software. The analogous location for add-on system software or for applications is `/opt/`*packagename*`/bin/ia64`.

`/usr/bin/sparcv9`
SPARC 64–bit, platform-dependent, user-invoked executables. This directory should not be part of a user's `$PATH`. A wrapper in `/usr/bin` should invoke the executable in this directory. See `isaexec`(3C). An approved installation location for bundled Solaris software. The analogous location for add-on system software or for applications is `/opt/`*packagename*`/bin/sparcv9`.

`/usr/bin/`*subsystem*
Platform-dependent user-invoked executables that are associated with *subsystem*. These are commands users expect to be run as part of their normal `$PATH`. An approved installation location for bundled Solaris software. The analogous location for add-on system software or for applications is `/opt/`*packagename*`/bin`.

`/usr/bin/`*subsystem*`/ia64`
Intel 64–bit, platform-dependent, user-invoked executables. Note that `ia64` is an example name; the actual name might be different. This directory should not be part of a user's `$PATH`. A wrapper in `/usr/bin` should invoke the executable in this directory. See `isaexec`(3C). An approved installation location for bundled Solaris software. The analogous location for add-on system software or for applications is `/opt/`*packagename*`/bin/ia64`.

`/usr/bin/`*subsystem*`/sparcv9`
SPARC 64–bit, platform-dependent, user-invoked executables. This directory should not be part of a user's `$PATH`. A wrapper in `/usr/bin`

should invoke the executable in this directory. See `isaexec`(3C). An approved installation location for bundled Solaris software. The analogous location for add-on system software or for applications is `/opt/`*packagename*`/bin/sparcv9`.

`/usr/`*subsystem*`/bin`
Platform-dependent user-invoked executables that are associated with *subsystem*. These are commands users expect to be run as part of their normal `$PATH`. An approved installation location for bundled Solaris software. The analogous location for add-on system software or for applications is `/opt/`*packagename*`/bin`.

`/usr/`*subsystem*`/bin/ia64`
Intel 64–bit, platform-dependent, user-invoked executables. Note that `ia64` is an example name; the actual name might be different. This directory should not be part of a user's `$PATH`. A wrapper in `/usr/bin` should invoke the executable in this directory. See `isaexec`(3C). An approved installation location for bundled Solaris software. The analogous location for add-on system software or for applications is `/opt/`*packagename*`/bin/ia64`.

`/usr/`*subsystem*`/bin/sparcv9`
SPARC 64–bit, platform-dependent, user-invoked executables. This directory should not be part of a user's `$PATH`. A wrapper in `/usr/bin` should invoke the executable in this directory. See `isaexec`(3C). An approved installation location for bundled Solaris software. The analogous location for add-on system software or for applications is `/opt/`*packagename*`/bin/sparcv9`.

`/usr/ccs`
C compilation system.

`/usr/ccs/bin`
C compilation commands and system utilities.

`/usr/ccs/lib`
Symbolic link to `/usr/lib`.

`/usr/demo`
Demo programs and data.

`/usr/dict`
Symbolic link to the `/usr/share/lib/dict` directory, which contains the dictionary file used by the UNIX spell program.

`/usr/dt`
root of a subtree for CDE software.

`/usr/dt/bin`

Primary location for CDE system utilities.

`/usr/dt/include`
Header files for CDE software.

`/usr/dt/lib`
Libraries for CDE software.

`/usr/dt/man`
On-line reference manual pages for CDE software.

`/usr/games`
An empty directory, a remnant of the SunOS 4.0/4.1 software.

`/usr/include`
Include headers (for C programs).

`/usr/java*`
Directories containing Java programs and libraries.

`/usr/kernel`
Subtree of platform-dependent loadable kernel modules, not needed in the root filesystem. An approved installation location for bundled Solaris software.

`/usr/kvm`
A mount point, retained for backward compatibility, that formerly contained platform-specific binaries and libraries.

`/usr/lib`
Platform-dependent libraries, various databases, commands and daemons not invoked directly by a human user. An approved installation location for bundled Solaris software. The analogous location for add-on system software or for applications is `/opt/`*packagename*`/lib`.

`/usr/lib/64`
Symbolic link to the most portable 64-bit Solaris interfaces.

`/usr/lib/acct`
Accounting scripts and binaries. See `acct`(1M).

`/usr/lib/class`
Scheduling-class-specific directories containing executables for `priocntl`(1) and `dispadmin`(1M).

`/usr/lib/dict`
Database files for `spell`(1).

`/usr/lib/font`
`troff`(1) font description files.

`/usr/lib/fs`
File system type dependent modules; generally not intended to be invoked directly by the user.

`/usr/lib/ia64`
Intel 64–bit, platform-dependent libraries, various databases, commands and daemons not invoked directly by a human user. Note that `ia64` is an example name; the actual name might be different. An approved installation location for bundled Solaris software. The analogous location for add-on system software or for applications is `/opt/`*packagename*`/lib/ia64`.

`/usr/lib/iconv`
Conversion tables for `iconv`(1).

`/usr/lib/libp`
Profiled libraries.

`/usr/lib/locale`
Localization databases.

`/usr/lib/lp`
Line printer subsystem databases and back-end executables.

`/usr/lib/mail`
Auxiliary programs for the `mail`(1) subsystem.

`/usr/lib/netsvc`
Internet network services.

`/usr/lib/nfs`
Auxiliary NFS-related programs and daemons.

`/usr/lib/pics`
Position Independent Code (PIC) archives needed to rebuild the run-time linker.

`/usr/lib/refer`
Auxiliary programs for `refer`(1).

`/usr/lib/sa`
Scripts and commands for the system activity report package. See `sar`(1).

`/usr/lib/saf`
Auxiliary programs and daemons related to the service access facility.

`/usr/lib/sparcv9`
SPARC 64-bit, platform-dependent libraries, various databases, commands and daemons not invoked directly by a human user. An approved installation location for bundled Solaris software. The

analogous location for add-on system software or for applications is `/opt/`*packagename*`/lib/sparcv9`.

`/usr/lib/spell`
Auxiliary programs and databases for `spell`(1). This directory is only present when the Binary Compatibility Package is installed.

`/usr/lib/uucp`
Auxiliary programs and daemons for `uucp`(1C).

`/usr/lib/`*subsystem*
Platform-dependent libraries, various databases, commands and daemons that are associated with *subsystem* and that are not invoked directly by a human user. An approved installation location for bundled Solaris software. The analogous location for add-on system software or for applications is `/opt/`*packagename*`/lib`.

`/usr/lib/`*subsystem*`/ia64`
Intel 64–bit, platform-dependent libraries, various databases, commands and daemons that are associated with *subsystem* and that are not invoked directly by a human user. Note that `ia64` is an example name; the actual name might be different. An approved installation location for bundled Solaris software. The analogous location for add-on system software or for applications is `/opt/`*packagename*`/lib/ia64`.

`/usr/lib/`*subsystem*`/sparcv9`
SPARC 64-bit, platform-dependent libraries, various databases, commands and daemons that are associated with *subsystem* and that are not invoked directly by a human user. An approved installation location for bundled Solaris software. The analogous location for add-on system software or for applications is `/opt/`*packagename*`/lib/sparcv9`.

`/usr/`*subsystem*`/lib`
Platform-dependent libraries, various databases, commands and daemons not invoked directly by a human user. An approved installation location for bundled Solaris software. The analogous location for add-on system software or for applications is `/opt/`*packagename*`/lib`.

`/usr/`*subsystem*`/lib/ia64`
Intel 64–bit, platform-dependent libraries, various databases, commands and daemons that are associated with *subsystem* and that are not invoked directly by a human user. Note that `ia64` is an example name; the actual name might be different. An approved installation location for bundled Solaris software. The analogous location for add-on system software or for applications is `/opt/`*packagename*`/lib/ia64`.

`/usr/`*subsystem*`/lib/sparcv9`

SPARC 64-bit, platform-dependent libraries, various databases, commands and daemons that are associated with *subsystem* and that are not invoked directly by a human user. An approved installation location for bundled Solaris software. The analogous location for add-on system software or for applications is `/opt/`*packagename*`/lib/sparcv9`.

`/usr/local`
Commands local to a site.

`/usr/mail`
Symbolic link to the `/var/mail` directory.

`/usr/man`
Symbolic link to the `/usr/share/man` directory.

`/usr/net/servers`
Entry points for foreign name service requests relayed using the network listener. See `listen`(1M).

`/usr/news`
Symbolic link to the `/var/news` directory.

`/usr/oasys`
Commands and files related to the Form and Menu Language Interpreter (FMLI) execution environment. See `face`(1).

`/usr/old`
Programs that are being phased out.

`/usr/openwin`
Installation or mount point for the OpenWindows software.

`/usr/perl5`
Perl 5 programs and documentation

`/usr/platform`
Subtree of platform-specific objects which does not need to reside on the root filesystem. It contains a series of directories, one per supported platform. The semantics of the series of directories is equivalent to `/platform`, except for subdirectories which do not provide utility under one or the other (for example, `/platform/include` is not needed).

`/usr/platform/‘uname –i‘/include`
Platform-specific system (`sys`, `vm`) header files with semantics equivalent to `/usr/include`. An approved installation location for bundled Solaris software and for add-on system software.

`/usr/platform/‘uname –i‘/kernel`

Platform-specific modules with semantics equivalent to `/usr/kernel`. An approved installation location for bundled Solaris software and for add-on system software.

`/usr/platform/‘uname –i‘/lib`
Platform-specific daemon and shared objects with semantics equivalent to `/usr/lib`. An approved installation location for bundled Solaris software and for add-on system software.

`/usr/platform/‘uname –i‘/lib/ia64`
Intel IA64 64–bit, platform-specific daemon and shared objects. Note that `ia64` is an example name; the actual name might be different. An approved installation location for bundled Solaris software and for add-on system software.

`/usr/platform/‘uname –i‘/lib/sparcv9`
SPARC 64–bit, platform-specific daemon and shared objects. An approved installation location for bundled Solaris software and for add-on system software.

`/usr/platform/‘uname –i‘/[s]man`*num*
Where *num* can be one of `3x`, `1m`, `4`, `7d`, or `9e`. Platform-specific system manual pages for documenting platform-specific, shared objects, administration utilities, configuration files, special files/modules, and header files. An approved installation location for bundled Solaris software and for add-on system software.

`/usr/platform/‘uname –i‘/sbin`
Platform-specific system administration utilities with semantics equivalent to `/usr/sbin`. An approved installation location for bundled Solaris software and for add-on system software.

`/usr/preserve`
Symbolic link to the `/var/preserve` directory.

`/usr/proc`
Directory for the `proc` tools.

`/usr/proc/bin`
Contains links to SPARC Version 8 binaries in `/usr/bin`.

`/usr/pub`
Files for online man page and character processing.

`/usr/sadm`
System administration files and directories.

`/usr/sadm/bin`

Binaries for the Form and Menu Language Interpreter (FMLI) scripts. See `fmli`(1).

`/usr/sadm/install`
Executables and scripts for package management.

`/usr/sbin`
Platform-dependent executables for system administration, expected to be run only by system administrators. An approved installation location for bundled Solaris software. The analogous location for add-on system software or for applications is `/opt/`*packagename*`/sbin`.

`/usr/sbin/`*subsystem*
Platform-dependent executables for system administration, expected to be run only by system administrators, and associated with *subsystem*. An approved installation location for bundled Solaris software. The analogous location for add-on system software or for applications is `/opt/`*packagename*`/sbin`.

`/usr/`*subsystem*`/sbin`
Platform-dependent executables for system administration, expected to be run only by system administrators, and associated with *subsystem*. An approved installation location for bundled Solaris software. The analogous location for add-on system software or for applications is `/opt/`*packagename*`/sbin`.

`/usr/sbin/static`
Statically linked version of selected programs from `/usr/bin` and `/usr/sbin`. These are used to recover from broken dynamic linking and before all pieces necessary for dynamic linking are present.

`/usr/share`
Platform-independent sharable files. An approved installation location for bundled Solaris software.

`/usr/share/lib`
Platform-independent sharable databases. An approved installation location for bundled Solaris software.

`/usr/share/lib/dict`
Contains word list for `spell`(1).

`/usr/share/lib/keytables`
Keyboard layout description tables.

`/usr/share/lib/mailx`
Help files for `mailx`(1).

`/usr/share/lib/nterm`

`nroff`(1) terminal tables.

`/usr/share/lib/pub`
Character set data files.

`/usr/share/lib/tabset`
Tab setting escape sequences.

`/usr/share/lib/terminfo`
Terminal description files for `terminfo`(4).

`/usr/share/lib/tmac`
Macro packages and related files for text processing tools, for example, `nroff`(1) and `troff`(1).

`/usr/share/lib/zoneinfo`
Time zone information.

`/usr/share/[s]man`
Platform-independent sharable manual pages. An approved installation location for bundled Solaris software. The analogous location for add-on system software or for applications is `/opt/`*packagename*`/[s]man`.

`/usr/share/src`
Source code for kernel, utilities, and libraries.

`/usr/snadm`
Files related to system and network administration..

`/usr/spool`
Symbolic link to the `/var/spool` directory.

`/usr/src`
Symbolic link to the `/usr/share/src` directory.

`/usr/tmp`
Symbolic link to the `var/tmp` directory.

`/usr/ucb`
Berkeley compatibility package binaries. See *Source Compatibility Guide*

`/usr/ucbinclude`
Berkeley compatibility package headers.

`/usr/ucblib`
Berkeley compatibility package libraries.

`/usr/vmsys`
Commands and files related to the Framed Access Command Environment (FACE) programs. See `face`(1).

`/usr/xpg4`
Directory for POSIX-compliant utilities.

SEE ALSO

`at`(1), `ex`(1), `face`(1), `fmli`(1), `iconv`(1), `lp`(1), `isainfo`(1), `mail`(1), `mailx`(1), `nroff`(1), `priocntl`(1), `refer`(1), `sar`(1), `sh`(1), `spell`(1), `troff`(1), `uname`(1), `uucp`(1C), `vi`(1), `acct`(1M), `cron`(1M), `dispadmin`(1M), `fsck`(1M), `init`(1M), `kernel`(1M), `mknod`(1M), `mount`(1M), `useradd`(1M), `ypbind`(1M), `mount`(2), `intro`(4), `terminfo`(4)

Binary Compatibility Guide Source Compatibility Guide

NAME

fnmatch – file name pattern matching

DESCRIPTION

The pattern matching notation described below is used to specify patterns for matching strings in the shell. Historically, pattern matching notation is related to, but slightly different from, the regular expression notation. For this reason, the description of the rules for this pattern matching notation is based on the description of regular expression notation described on the `regex`(5) manual page.

Patterns Matching a Single Character

The following *patterns matching a single character* match a single character: *ordinary characters*, *special pattern characters* and *pattern bracket expressions*. The pattern bracket expression will also match a single collating element.

An ordinary character is a pattern that matches itself. It can be any character in the supported character set except for `NUL`, those special shell characters that require quoting, and the following three special pattern characters. Matching is based on the bit pattern used for encoding the character, not on the graphic representation of the character. If any character (ordinary, shell special, or pattern special) is quoted, that pattern will match the character itself. The shell special characters always require quoting.

When unquoted and outside a bracket expression, the following three characters will have special meaning in the specification of patterns:

`?` A question-mark is a pattern that will match any character.

`*` An asterisk is a pattern that will match multiple characters, as described in `Patterns Matching Multiple Characters`, below.

`[` The open bracket will introduce a pattern bracket expression.

The description of basic regular expression bracket expressions on the `regex`(5) manual page also applies to the pattern bracket expression, except that the exclamation-mark character `( ! )` replaces the circumflex character (`^`) in its role in a *non-matching list* in the regular expression notation. A bracket expression starting with an unquoted circumflex character produces unspecified results.

The restriction on a circumflex in a bracket expression is to allow implementations that support pattern matching using the circumflex as the negation character in addition to the exclamation-mark. A portable application must use something like `[\^!]` to match either character.

When pattern matching is used where shell quote removal is not performed (such as in the argument to the `find –name` primary when `find` is being called using one of the `exec` functions, or in the *pattern* argument to the `fnmatch`(3C) function, special characters can be escaped to remove their special meaning by preceding them with a backslash character. This escaping backslash will be discarded. The sequence `\\` represents one literal backslash. All of the requirements and effects of quoting on ordinary, shell special and special pattern characters will apply to escaping in this context.

Both quoting and escaping are described here because pattern matching must work in three separate circumstances:

- Calling directly upon the shell, such as in pathname expansion or in a `case` statement. All of the following will match the string or file `abc`:

`abc`	`"abc"`	`a"b"c`	`a\bc`	`a[b]c`
`a["b"]c`	`a[\b]c`	`a["\b"]c`	`a?c`	`a*c`

  The following will not:

`"a?c"`	`a\*c`	`a\[b]c`

- Calling a utility or function without going through a shell, as described for `find`(1) and the function `fnmatch`(3C)
- Calling utilities such as `find`, `cpio`, `tar` or `pax` through the shell command line. In this case, shell quote removal is performed before the utility sees the argument. For example, in:

  find /bin -name e\c[\h]o -print

  after quote removal, the backslashes are presented to `find` and it treats them as escape characters. Both precede ordinary characters, so the `c` and `h` represent themselves and `echo` would be found on many historical systems (that have it in `/bin`). To find a file name that contained shell special characters or pattern characters, both quoting and escaping are required, such as:

  ```
  pax -r . . .  "*a\(\?"
  ```

  to extract a filename ending with `a(?`.

Conforming applications are required to quote or escape the shell special characters (sometimes called metacharacters). If used without this protection, syntax errors can result or implementation extensions can be triggered. For example, the KornShell supports a series of extensions based on parentheses in patterns; see `ksh`(1)

Patterns Matching Multiple Characters

The following rules are used to construct *patterns matching multiple characters* from *patterns matching a single character*:

- The asterisk (*) is a pattern that will match any string, including the null string.

- The concatenation of *patterns matching a single character* is a valid pattern that will match the concatenation of the single characters or collating elements matched by each of the concatenated patterns.
- The concatenation of one or more *patterns matching a single character* with one or more asterisks is a valid pattern. In such patterns, each asterisk will match a string of zero or more characters, matching the greatest possible number of characters that still allows the remainder of the pattern to match the string.

Since each asterisk matches zero or more occurrences, the patterns `a*b` and `a**b` have identical functionality.

Examples:

`a[bc]` matches the strings `ab` and `ac`.

`a*d` matches the strings `ad`, `abd` and `abcd`, but not the string `abc`.

`a*d*` matches the strings `ad`, `abcd`, `abcdef`, `aaaad` and `adddd`.

`*a*d` matches the strings `ad`, `abcd`, `efabcd`, `aaaad` and `adddd`.

Patterns Used for Filename Expansion

The rules described so far in `Patterns Matching Multiple Characters` and `Patterns Matching a Single Character` are qualified by the following rules that apply when pattern matching notation is used for filename expansion.

1. The slash character in a pathname must be explicitly matched by using one or more slashes in the pattern; it cannot be matched by the asterisk or question-mark special characters or by a bracket expression. Slashes in the pattern are identified before bracket expressions; thus, a slash cannot be included in a pattern bracket expression used for filename expansion. For example, the pattern `a[b/c]d` will not match such pathnames as `abd` or `a/d`. It will only match a pathname of literally `a[b/c]d`.
2. If a filename begins with a period (.), the period must be explicitly matched by using a period as the first character of the pattern or immediately following a slash character. The leading period will not be matched by:
   - the asterisk or question-mark special characters
   - a bracket expression containing a non-matching list, such as:

   `[!a]`

   a range expression, such as:

   `[%–0]`

or a character class expression, such as:

```
[[:punct:]]
```

It is unspecified whether an explicit period in a bracket expression matching list, such as:

```
[.abc]
```

can match a leading period in a filename.

3. Specified patterns are matched against existing filenames and pathnames, as appropriate. Each component that contains a pattern character requires read permission in the directory containing that component. Any component, except the last, that does not contain a pattern character requires search permission. For example, given the pattern:

   ```
   /foo/bar/x*/bam
   ```

   search permission is needed for directories `/` and `foo`, search and read permissions are needed for directory `bar`, and search permission is needed for each `x*` directory.

   If the pattern matches any existing filenames or pathnames, the pattern will be replaced with those filenames and pathnames, sorted according to the collating sequence in effect in the current locale. If the pattern contains an invalid bracket expression or does not match any existing filenames or pathnames, the pattern string is left unchanged.

SEE ALSO

`find`(1), `ksh`(1), `fnmatch`(3C), `regex`(5)

NAME

fns – overview of FNS

DESCRIPTION

Federated Naming Service (FNS) provides a method for federating multiple naming services under a single, simple interface for the basic naming operations. The service supports resolution of *composite* names, names that span multiple naming systems, through the naming interface. In addition to the naming interface, FNS also specifies *policies* for composing names in the enterprise namespace. See `fns_policies`(5) and `fns_initial_context`(5).

Fundamental to the FNS model are the notions of composite names and *contexts*. A context provides operations for:

- associating (binding) names to objects
- resolving names to objects
- removing bindings, listing names, renaming and so on.

A context contains a set of names to reference bindings. A reference contains a list of communication end-points. Every naming operation in the FNS interface is performed on a context object.

The federated naming system is formed by contexts from one naming system being bound in the contexts of another naming system. Resolution of a composite name proceeds from contexts within one naming system to those in the next, until the name is resolved.

XFN

XFN is *X/Open Federated Naming*. The programming interface and policies that FNS supports are specified by XFN. See `xfn`(3XFN) and `fns_policies`(5).

Composite Names

A composite name is a name that spans multiple naming systems. It consists of an ordered list of components. Each component is a name from the namespace of a single naming system. FNS defines the syntax for constructing a composite name using names from component naming systems. Individual naming systems are responsible for the syntax of each component.

The syntax for composite names is that components are composed left to right using the slash character (’`/`’) as the component separator. For example, the composite name `.../Wiz.Com/site/Oceanview.East` consists of four components: `...`, `Wiz.COM`, `site`, and `Oceanview.East`. See `fns_policies`(5) and `fns_initial_context`(5) for more examples of composite names.

Why FNS?

FNS is useful for the following reasons:

- A single uniform naming interface is provided to clients for accessing naming services. Consequently, the addition of new naming services does not require changes to applications or existing naming services.

Furthermore, applications that use FNS will be portable across platforms because the interface exported by FNS is XFN, a public, open interface endorsed by other vendors and by the X/Open Company.

- Names can be composed in a uniform way (that is, FNS supports a model in which composite names are constructed in a uniform syntactic way and can have any number of components).
- Coherent naming is encouraged through the use of shared contexts and shared names.

FNS and Naming Systems

FNS has support for NIS+, NIS, and files as enterprise-level naming services. This means that FNS implements the enterprise-level policies using NIS+, NIS, and files. FNS also supports DNS and X.500 (via DAP or LDAP) as global naming services, as well as support for federating NIS+ and NIS with DNS and X.500. See the corresponding individual man page for information about the implementation for a specific naming service.

SEE ALSO

`nis+`(1), `xfn`(3XFN), `fns_dns`(5), `fns_files`(5), `fns_initial_context`(5), `fns_nis`(5), `fns_nis+`(5), `fns_policies`(5), `fns_references`(5), `fns_x500`(5)

NAME

fns_dns – overview of FNS over DNS implementation

DESCRIPTION

Federated Naming Service (FNS) provides a method for federating multiple naming services under a single, simple interface for the basic naming operations. One of the naming services supported by FNS is the Internet Domain Name System, or DNS (see `in.named`(1M)). DNS is a hierarchical collection of name servers that provide the Internet community with host and domain name resolution. FNS uses DNS to name entities globally. Names can be constructed for any enterprise that is accessible on the Internet; consequently, names can also be constructed for objects exported by these enterprises.

FNS provides the XFN interface for performing naming resolution on DNS domains and hosts. In addition, enterprise namespaces such as those served by NIS+ and NIS can be federated with DNS by adding TXT records to DNS. To federate an NIS+ or NIS namespace under DNS, you first obtain the root reference for the NIS+ hierarchy or NIS domain. This reference is referred to as the *next naming system reference* because it refers to the *next* naming system beneath the DNS domain. This reference contains information about how to communicate with the NIS+ or NIS servers and has the following format:

<`domainname`> <*server name*> [<*server address*>]

where <`domainname`> is the fully qualified domain name. Notice that NIS+ and NIS have slightly different syntaxes for domain names. For NIS+, the fully qualified domain name is case-insensitive and terminated by a dot character ('`.`'). For NIS, the fully qualified domain name is case-sensitive and is *not* terminated by a dot character. For both NIS+ and NIS, <*server address*> is optional. If it is not supplied, a host name lookup will be performed to get the machine's address.

For example, if the machine `wiz-nisplus-server` with address `133.33.33.33` serves the NIS+ domain `wiz.com.`, the reference would look like this:

```
wiz.com. wiz-nisplus-server 133.33.33.33
```

For NIS, the reference information is of the form:

<`domainname`> <*server name*>

For example, if the machine `woz-nis-server` serves the NIS domain `Woz.COM`, the reference would look like this:

```
Woz.COM woz-nis-server
```

After obtaining this information, you then edit the DNS table (see `in.named`(1M)) and add a TXT record with this reference information. The TXT record must be associated with a DNS domain that includes an NIS record. For example, the reference information shown in the examples above would be entered as follows.

For NIS+:

```
TXT "XFNNISPLUS wiz.com.  wiz-nisplus-server 133.33.33.33"
```

For NIS:

```
TXT "XFNNIS woz.com woz-nis-server"
```

Note the mandatory double quotes (' " ') delimiting the contents of the TXT record. After making any changes to the DNS table, you must notify the server by either restarting it or sending it a signal to reread the table:

```
#kill -HUP ‘cat /etc/named.pid‘
```

This update effectively adds the next naming system reference to DNS. You can look up this reference using `fnlookup`(1) to see if the information has been added properly. For example, the following command looks up the next naming system reference of the DNS domain `Wiz.COM`:

```
#fnlookup -v .../Wiz.COM/
```

Note the mandatory trailing slash ('/').

After this administrative step has been taken, clients outside of the NIS+ hierarchy or NIS domain can access and perform operations on the contexts in the NIS+ hierarchy or NIS domain. Foreign NIS+ clients access the hierarchy as unauthenticated NIS+ clients. Continuing the example above, and assuming that NIS+ is federated underneath the DNS domain `Wiz.COM`, you can now list the root of the NIS+ enterprise using the command:

```
#fnlist .../Wiz.COM/
```

SEE ALSO

`fnlist`(1), `fnlookup`(1), `nis+`(1), `in.named`(1M), `ypserv`(1M), `xfn`(3XFN), `fns`(5), `fns_nis`(5), `fns_nis+`(5), `fns_references`(5), `fns_x500`(5)

NAME

fns_files – overview of FNS over files implementation

DESCRIPTION

The Federated Naming Service (FNS) provides a method for federating multiple naming services under a single, simple interface for the basic naming operations. One of the naming services supported by FNS is /etc files. FNS provides the XFN interface for performing naming and attribute operations on FNS enterprise objects (organization, site, user, host, and service objects), using files as the naming service. FNS stores bindings for these objects in files and uses them in conjunction with existing /etc files objects.

FNS Policies and /etc Files

FNS defines policies for naming objects in the federated namespace (see fns_policies(5)). At the enterprise level, FNS policies specify naming for organizations, hosts, users, sites, and services. The enterprise-level naming service provides contexts to allow other objects to be named relative to these objects.

The organizational unit namespace provides a hierarchical namespace for naming subunits of an enterprise. In /etc files, there is no concept of an organization. Hence, with respect to /etc files as the naming service, there is a single organizational unit context that represents the entire system. Users in an FNS organizational unit correspond to the users in the /etc/passwd file. FNS provides a context for each user in the /etc/passwd file.

Hosts in an FNS organizational unit correspond to the hosts in the /etc/hosts file. FNS provides a context for each host in the /etc/hosts file.

Security Considerations

Changes to the FNS information (using the commands fncreate(1M), fncreate_fs(1M), fnbind(1), fndestroy(1M) and fnunbind(1)) can be performed only by the privileged users on the system that exports the /var/fn directory. Also, based on the UNIX user IDs, users are allowed to modify their own contexts, bindings, and attributes, from any machine that mounts the /var/fn directory.

For example, the command fncreate(1M) creates FNS related files and directories in the system on which the command is executed. Hence, the invoker of the fncreate(1M) command must have super-user privileges in order to create the user, host, site, and service contexts. However, a user could use the fnunbind(1) command to create calendar bindings in the user’s own context, as in this example:

```
fnbind –r thisuser/service/calendar onc_calendar
onc_cal_str jsmith@beatrix
```

The files object name that corresponds to an FNS composite name can be obtained using fnlookup(1) and fnlist(1).

USAGE

The files used for storing FNS information are placed in the directory `/var/fn`. The machine on which `/var/fn` is located has access to the FNS file. The FNS information can be made accessible to other machines by exporting `/var/fn`. Client machines that NFS mount the `/var/fn` directory would then be able to access the FNS information.

SEE ALSO

`fnbind`(1), `fnlist`(1), `fnlookup`(1), `fnunbind`(1), `fncreate`(1M), `fncreate_fs`(1M), `fndestroy`(1M), `xfn`(3XFN), `fns`(5), `fns_initial_context`(5), `fns_nis`(5), `fns_nis+`(5), `fns_policies`(5), `fns_references`(5)

NAME

fns_initial_context – overview of the FNS Initial Context

DESCRIPTION

Every FNS name is interpreted relative to some context, and every FNS naming operation is performed on a context object. The FNS programming interface (XFN) provides a function that allows the client to obtain an *Initial Context* object. The Initial Context provides the initial pathway to other FNS contexts. FNS defines a set of bindings that the client can expect to find in this context,

FNS assumes that for every process:

1. There is a user associated with the process when `fn_ctx_handle_from_initial( )` is invoked. This association is based on the effective uid of the process. In the following discussion this user is denoted by *U*. The association of user to process may change during the life of a process but does not affect the context handle originally returned by `fn_ctx_handle_from_initial( )`.
2. The process is running on a host when `fn_ctx_handle_from_initial( )` is invoked. In the following discussion this host is denoted by *H*.

The following atomic names can appear in the Initial Context:

`...`	`thishost`	`thisorgunit`
`thisens`	`myself`	`myorgunit`
`myens`	`orgunit`	`site`
`user`	`host`	

Except for `...`, these names with an added underscore ('_') prefix are also in the Initial Context and have the same binding as their counterpart (for example, `thishost` and `_thishost` have the same binding). In addition, `org` has the same binding as `orgunit`, and `thisuser` has the same binding as `myself`. The bindings for these names are summarized in the following table.

Some of these names may not necessarily appear in all Initial Contexts. For example, a process owned by the super-user of a machine does not have any of the user-related bindings. Or, for another example, an installation that has not set up a site namespace will not have the site-related bindings.

...	global context for resolving DNS or X.500 names. Synonym: `/...`
thishost	*H*'s host context. Synonym: `_thishost`
thisens	the enterprise root of *H*. Synonym: `_thisens`

thisorgunit	*H*'s distinguished organizational unit context. In Solaris, this is *H*'s NIS+ home domain. Synonym: `_thisorgunit`
myself	*U*'s user context. Synonyms: `_myself`, `thisuser`
myens	the enterprise root of *U*. Synonym: `_myens`
myorgunit	*U*'s distinguished organizational unit context. In Solaris, this is *U*'s NIS+ home domain. Synonym: `_myorgunit`
user	the context in which users in the same organizational unit as *H* are named. Synonym: `_user`
host	the context in which hosts in the same organizational unit as *H* are named. Synonym: `_host`
org	the root context of the organizational unit namespace in *H*'s enterprise. In Solaris, this corresponds to the NIS+ root domain. Synonyms: `orgunit`, `_orgunit`
site	the root context of the site namespace in *H*'s enterprise, if the site namespace has been configured. Synonym: `_site`

EXAMPLES

EXAMPLE 1 Names beginning with the enterprise root

The types of objects that may be named relative to the enterprise root are user, host, service, organizational unit, file, and site. Here are some examples of names that begin with the enterprise root:

`thisens/orgunit/multimedia.servers.engineering`
: names an organizational unit `multimedia.servers.engineering` in *H*'s enterprise.

`thisens/site/northwing.floor3.admin`
: names the north wing site, on the third floor of the administrations building in *H*'s enterprise.

`myens/user/hdiffie`
: names the user `hdiffie` in *U*'s enterprise.

`myens/service/teletax`
: names the `teletax` service of *U*'s enterprise.

EXAMPLE 2 Names beginning with organizational unit names

The types of objects that may be named relative to an organizational unit name are: user, host, service, file, and site. Here are some examples of names that begin with organizational unit names (either explicitly via `org`, or implicitly via `thisorgunit` or `myorgunit`), and name objects relative to organizational unit names when resolved in the Initial Context:

`org/accounts_payable.finance/site/videoconference.northwing`
names a conference room `videoconference` located in the north wing of the site associated with the organizational unit `accounts_payable.finance`.

`org/finance/user/mjones`
names a user `mjones` in the organizational unit `finance`.

`org/finance/host/inmail`
names a machine `inmail` belonging to the organizational unit `finance`.

`org/accounts_payable.finance/fs/pub/blue-and-whites/FY92-124`
names a file `pub/blue-and-whites/FY92-124` belonging to the organizational unit `accounts_payable.finance`.

`org/accounts_payable.finance/service/calendar`
names the `calendar` service of the organizational unit `accounts_payable.finance`. This might manage the meeting schedules of the organizational unit.

`thisorgunit/user/cmead`
names the user `cmead` in *H*'s organizational unit.

`myorgunit/fs/pub/project_plans/widget.ps`
names the file `pub/project_plans/widget.ps` exported by *U*'s organizational unit's file system.

EXAMPLE 3 Names beginning with site names

The types of objects that may be named relative to a site name are users, hosts, services, and files. Here are some examples of names that begin with site names via `site`, and name objects relative to sites when resolved in the Initial Context:

`site/b5.mtv/service/printer/speedy`
names a printer `speedy` in the `b5.mtv` site.

`site/admin/fs/usr/dist`
names a file directory `usr/dist` available in the site `admin`.

EXAMPLE 4 Names beginning with user names

The types of objects that may be named relative to a user name are services and files. Here are some examples of names that begin with user names (explicitly via `user` or implicitly via `thisuser`), and name objects relative to users when resolved in the Initial Context:

`user/jsmith/service/calendar`
: names the `calendar` service of the user `jsmith`.

`user/jsmith/fs/bin/games/riddles`
: names the file `bin/games/riddles` of the user `jsmith`.

`thisuser/service/printer`
: names the `printer` service of *U*.

EXAMPLE 5 Names beginning with host names

The types of objects that may be named relative to a host name are services and files. Here are some examples of names that begin with host names (explicitly via `host or implicitly via thishost`), and name objects relative to hosts when resolved in the Initial Context:

`host/mailhop/service/mailbox`
: names the `mailbox` service associated with the machine `mailhop`.

`host/mailhop/fs/pub/saf/archives.91`
: names the directory `pub/saf/archives.91` found under the root directory of the machine `mailhop`.

`thishost/service/printer`
: names the `printer` service of *H*.

SEE ALSO

`nis+`(1), `geteuid`(2), `fn_ctx_handle_from_initial`(3XFN), `xfn`(3XFN), `fns`(5), `fns_policies`(5)

NAME

fns_nis+ – overview of FNS over NIS+ implementation

DESCRIPTION

Federated Naming Service (FNS) provides a method for federating multiple naming services under a single, simple interface for the basic naming operations. One of the naming services supported by FNS is NIS+, the enterprise-wide information service in Solaris (see `nis+`(1)). FNS provides the XFN interface for performing naming and attribute operations on FNS enterprise objects (organization, site, user, host, and service objects) using NIS+. FNS stores bindings for these objects in NIS+ and uses them in conjunction with existing NIS+ objects.

FNS Policies and NIS+

FNS defines policies for naming objects in the federated namespace (see `fns_policies`(5)). At the enterprise level, FNS policies specify naming for organizations, hosts, users, sites, and services. The enterprise-level naming service provides contexts to allow other objects to be named relative to these objects.

The organizational unit namespace provides a hierarchical namespace for naming subunits of an enterprise. An organizational unit maps to an NIS+ domain. Organizational unit names can be either fully qualified NIS+ domain names or relatively NIS+ domain names. If a terminal dot is present in the name, it is treated as a fully qualified name. Otherwise, the name is resolved relative to the root NIS+ domain.

Users in the NIS+ namespace are found in the `passwd.org_dir` table of an NIS+ domain. Users in an FNS organizational unit correspond to the users in the `passwd.org_dir` table of the corresponding NIS+ domain. FNS provides a context for each user in the `passwd.org_dir` table.

Hosts in the NIS+ namespace are found in the `hosts.org_dir` table of an NIS+ domain. Hosts in an FNS organizational unit correspond to the hosts in the `hosts.org_dir` table of the corresponding NIS+ domain. FNS provides a context for each host in the `hosts.org_dir` table.

In NIS+, users and hosts have a notion of a *home domain*. It is the primary NIS+ domain that maintains information associated with them. A user or host's home domain can be determined directly using its NIS+ principal name, which is composed of the atomic user (login) name or the atomic host name, and the name of the NIS+ home domain. For example, user `jsmith` with home domain `wiz.com` has an NIS+ principal name, `jsmith.wiz.com`.

A user's NIS+ home domain corresponds to the user's FNS organizational unit and determines the binding for `myens` and `myorgunit`.

A host's NIS+ home domain corresponds to the host's FNS organizational unit and determines the binding for `thisens`, `thisorgunit`, `user`, and `host`.

Federating NIS+ with DNS or X.500

Federating NIS+ with the global naming systems DNS or X.500 makes NIS+ contexts accessible outside of an NIS+ hierarchy. To enable the federation, the administrator must first add address information in either DNS or X.500 (see `fns_dns`(5) and `fns_x500`(5)). After this administrative step has been taken, clients outside of the NIS+ hierarchy can access contexts and perform operations from outside the hierarchy as an unauthenticated NIS+ client.

NIS+ Security

The command `fncreate`(1M) creates NIS+ tables and directories in the NIS+ hierarchy associated with the domain of the host on which it executes. The invoker of `fncreate`(1M) and other FNS commands is expected to have the necessary NIS+ credentials. (See `nis+`(1) and `nisdefaults`(1)). The environment variable `NIS_GROUP` of the process specifies the group owner for the NIS+ objects thus created. In order to facilitate administration of the NIS+ objects, `NIS_GROUP` should be set to the name of the NIS+ administration group for the domain prior to executing `fncreate`(1M) and other FNS commands. Changes to NIS+-related properties, including default access control rights, could be effected using NIS+ administration tools and interfaces after the context has been created. The NIS+ object name that corresponds to an FNS composite name can be obtained using `fnlookup`(1) and `fnlist`(1).

SEE ALSO

`fnlist`(1), `fnlookup`(1), `nis+`(1), `nischgrp`(1), `nischmod`(1), `nischown`(1), `nisdefaults`(1), `nisls`(1), `fncreate`(1M), `xfn`(3XFN), `fns`(5), `fns_dns`(5), `fns_files`(5), `fns_initial_context`(5), `fns_nis`(5), `fns_policies`(5), `fns_references`(5), `fns_x500`(5)

NAME

fns_nis – overview of FNS over NIS (YP) implementation

DESCRIPTION

Federated Naming Service (FNS) provides a method for federating multiple naming services under a single, simple interface for the basic naming operations. One of the naming services supported by FNS is NIS (YP), the enterprise-wide information services in Solaris (see `ypcat`(1), `ypmatch`(1), `ypfiles`(4)). FNS provides the XFN interface for performing naming and attribute operations on FNS enterprise objects (organization, site, user, host and service objects) using NIS. FNS stores bindings for these objects in NIS and uses them in conjunction with existing NIS objects.

FNS Policies and NIS

FNS defines policies for naming objects in the federated namespace (see `fns_policies`(5)). At the enterprise level, FNS policies specify naming for organizations, hosts, users, sites, and services. The enterprise-level naming service provides contexts to allow other objects to be named relative to these objects.

The FNS organizational unit namespace provides a hierarchical namespace for naming subunits of an enterprise. However, NIS does not support a hierarchical organizational structure. Therefore, a NIS domain maps to a single organizational unit in the FNS namespace.

Users in an FNS organizational unit correspond to the users in the `passwd.byname` map of the corresponding NIS domain. FNS provides a context for each user in the `passwd.byname` map.

Hosts in an FNS organizational unit correspond to the hosts in the `hosts.byname` map of the corresponding NIS domain. FNS provides a context for each host in the `hosts.byname` map.

Federating NIS with DNS or X.500

Federating NIS with the global naming systems DNS or X.500 makes NIS contexts accessible outside of an NIS domain. To enable the federation, the administrator must first add address information in either DNS or X.500 (see `fns_dns`(5) and `fns_x500`(5)). After this administrative step has been taken, clients outside of the NIS domain can access contexts and perform operations.

Security Considerations

Changes to the FNS information (using the commands `fncreate`(1M), `fncreate_fs`(1M), `fncreate_printer`(1M), `fnbind`(1), `fndestroy`(1M), `fncheck`(1M), and `fnunbind`(1)) can be performed only by the privileged users on the NIS master server that maintains the FNS information.

For example, the command `fncreate`(1M) creates the NIS map for the associated NIS domain in the system on which it is executed. Hence, the command must be run by a privileged user either on the NIS master server or on a system that will serve as a NIS master server for FNS.

The NIS object name that corresponds to an FNS composite name can be obtained using `fnlookup`(1) and `fnlist`(1).

SEE ALSO

fnbind(1), fnlist(1), fnlookup(1), fnunbind(1), ypcat(1), ypmatch(1), fncheck(1M), fncreate(1M), fncreate_fs(1M), fncreate_printer(1M), fndestroy(1M), xfn(3XFN), ypfiles(4), fns(5), fns_dns(5), fns_files(5), fns_initial_context(5), fns_nis+(5), fns_policies(5), fns_references(5), fns_x500(5)

NAME

fns_policies – overview of the FNS Policies

DESCRIPTION

FNS defines policies for naming objects in the federated namespace. The goal of these policies is to allow easy and uniform composition of names. The policies use the basic rule that objects with narrower scopes are named relative to objects with wider scopes.

FNS policies are described in terms of the following three categories: global, enterprise, and application.

Global naming service
: A global naming service is a naming service that has world-wide scope. Internet DNS and X.500 are examples of global naming services. The types of objects named at this global level are typically countries, states, provinces, cities, companies, universities, institutions, and government departments and ministries. These entities are referred to as *enterprises*.

Enterprise-level naming service
: Enterprise-level naming services are used to name objects within an enterprise. Within an enterprise, there are naming services that provide contexts for naming common entities such as organizational units, physical sites, human users, and computers. Enterprise-level naming services are bound below the global naming services. Global naming services provide contexts in which the root contexts of enterprise-level naming services can be bound.

Application-level naming service
: Application-level naming services are incorporated in applications offering services such as file service, mail service, print service, and so on. Application-level naming services are bound below enterprise naming services. The enterprise-level naming services provide contexts in which contexts of application-level naming services can be bound.

FNS has policies for global and enterprise naming. Naming within applications is left to individual applications or groups of related applications and not specified by FNS.

FNS policy specifies that DNS and X.500 are global naming services that are used to name enterprises. The global namespace is named using the name `...`. A DNS name or an X.500 name can appear after the `...`. Support for federating global naming services is planned for a future release of FNS.

Within an enterprise, there are namespaces for organizational units, sites, hosts, users, files and services, referred to by the names `orgunit`, `site`, `host`, `user`, `fs`, and `service`. In addition, these namespaces can be named using these names with an added underscore ('_') prefix (for example, `host` and `_host` have the same binding). The following table summarizes the FNS policies.

Context Type	Subordinate Context	Parent Context
org unit	site user host file system service	enterprise root
site	user host file system service	enterprise root org unit
user	service file system	enterprise root org unit
host	service file system	enterprise root org unit
service	not specified	enterprise root org unit site user host
file system	none	enterprise root

Context Type	Subordinate Context	Parent Context
		org unit site user host

In Solaris, an organizational unit name corresponds to an NIS+ domain name and is identified using either the fully-qualified form of its NIS+ domain name, or its NIS+ domain name relative to the NIS+ root. Fully-qualified NIS+ domain names have a terminal dot ('`.`'). For example, assume that the NIS+ root domain is "Wiz.COM." and "sales" is a subdomain of that. Then, the names `org/sales.Wiz.COM.` and `org/sales` both refer to the organizational unit corresponding to the same NIS+ domain `sales.Wiz.COM`.

User names correspond to names in the corresponding NIS+ *passwd.org_dir* table. The file system context associated with a user is obtained from his entry in the NIS+ *passwd.org_dir* table.

Host names correspond to names in the corresponding NIS+ *hosts.org_dir* table. The file system context associated with a host corresponds to the files systems exported by the host.

EXAMPLES

EXAMPLE 1 The types of objects that may be named relative to an organizational unit name are: user, host, service, file, and site. Here are some examples of names name objects relative to organizational unit names:

`org/accounts_payable.finance/site/videoconference.northwing`
names a conference room `videoconference` located in the north wing of the site associated with the organizational unit `accounts_payable.finance`.

`org/finance/user/mjones`
names a user `mjones` in the organizational unit `finance`.

`org/finance/host/inmail`
names a machine `inmail` belonging to the organizational unit `finance`.

`org/accounts_payable.finance/fs/pub/blue-and-whites/FY92-124`
names a file `pub/blue-and-whites/FY92-124` belonging to the organizational unit `accounts_payable.finance`.

`org/accounts_payable.finance/service/calendar` names the `calendar` service of the organizational unit `accounts_payable.finance`. This might manage the meeting schedules of the organizational unit.

EXAMPLE 2 The types of objects that may be named relative to a site name are services and files. Here are some examples of names that name objects relative to sites:

`site/b5.mtv/service/printer/speedy`
names a printer `speedy` in the `b5.mtv` site.

`site/admin/fs/usr/dist`
names a file directory `usr/dist` available in the site `admin`.

EXAMPLE 3 The types of objects that may be named relative to a user name are services and files. Here are some examples of names that name objects relative to users:

`user/jsmith/service/calendar`
names the `calendar` service of the user `jsmith`.

`user/jsmith/fs/bin/games/riddles`
names the file `bin/games/riddles` of the user `jsmith`.

EXAMPLE 4 The types of objects that may be named relative to a host name are services and files. Here are some examples of names that name objects relative to hosts:

`host/mailhop/service/mailbox`
names the `mailbox` service associated with the machine `mailhop`.

`host/mailhop/fs/pub/saf/archives.91`
names the directory `pub/saf/archives.91` found under the root directory of the machine `mailhop`.

SEE ALSO

`fncreate`(1M), `nis+`(1), `xfn`(3XFN), `fns`(5), `fns_initial_context`(5), `fns_references`(5)

NAME

fns_references – overview of FNS References

DESCRIPTION

Every composite name in FNS is bound to a *reference*. A reference consists of a type and a list of addresses. The reference type is used to identify the type of object.

An address is something that can be used with some communication mechanism to invoke operations on an object or service. Multiple addresses are intended to identify multiple communication endpoints for a single conceptual object or service. Each address in a reference consists of an address type and an opaque buffer. The address type determines the format and interpretation of the address data. Together, the address's type and data specify how to reach the object. Many communication mechanisms are possible; FNS does not place any restrictions on them.

The following summarizes the reference and address types that are currently defined. New types should be registered with the Federated Naming Group at SunSoft.

Reference Types

All reference types use the `FN_ID_STRING` identifier format unless otherwise qualified.

`onc_fn_enterprise`	Enterprise root context.
`onc_fn_organization`	A context for naming objects related to an organizational unit.
`onc_fn_hostname`	A context for naming hosts.
`onc_fn_username`	A context for naming users.
`onc_fn_user`	A context for naming objects related to a user.
`onc_fn_host`	A context for naming objects related to a computer.
`onc_fn_site`	A context for naming sites.
`onc_fn_service`	A context for naming services.
`onc_fn_nsid`	A context for naming namespace identifiers.
`onc_fn_generic`	A context for naming application-specific objects.
`onc_fn_fs`	A context for naming files, directories, and file systems.
`onc_fn_printername`	A context for naming printers.
`onc_printers`	A printer object. When implemented on top of NIS+, this could also be a context for naming printers.

`fn_link_ref`	An XFN link.
`inet_domain`	An Internet domain.

Address Types

All address types use the `FN_ID_STRING` identifier format unless otherwise qualified. The format of address contents is determined by the corresponding address type.

`onc_fn_nisplus`	For an FNS enterprise-level object implemented on top of NIS+. The address contains the context type, context representation type (either normal or merged), version number of the reference, and the NIS+ name of the object. The only intended use of this reference is that it be passed to `fn_ctx_handle_from_ref`(3XFN)
`onc_fn_nis`	For an FNS enterprise-level object implemented on top of NIS. The address contains the context type and version number of the reference, and the NIS name of the object. The only intended use of this reference is that it be passed to `fn_ctx_handle_from_ref`(3XFN).
`onc_fn_files`	For an FNS enterprise-level object implemented on top of `/etc` files. The address contains the context type and version number of the reference, and the location of the object in the `/etc` file system. The only intended use of this reference is that it be passed to `fn_ctx_handle_from_ref`(3XFN).
`onc_fn_fs_user`	For a user's home directory. The address contains the user's name and the name of the naming service password table where the user's home directory is stored.
`onc_fn_fs_user_nisplus`	For a user's home directory. The address contains the user's name and the name of the NIS+ password table

	where the user's home directory is stored.
`onc_fn_fs_host`	For all file systems exported by a host. The address contains the host's name.
`onc_fn_fs_mount`	For a single mount point. The address contains the mount options, the name of the servers and the exported path. See `mount`(1M).
`onc_fn_printer_files`	For a printer's address in the files naming service.
`onc_fn_printer_nis`	For a printer's address in the NIS naming service.
`onc_fn_printer_nisplus`	For a printer's address in the NIS+ naming service.
`fn_link_addr`	For an XFN link address. The contents is the string form of the composite name.
`inet_domain`	For an Internet domain. The address contains the fully-qualified domain name (for example, "Wiz.COM.")
`inet_ipaddr_string`	For an object with an Internet address. The address contains an internet IP address in dotted string form (for example, "192.144.2.3").
`x500`	For an X.500 object. The address contains an X.500 Distinguished Name, in the syntax specified in the `X/Open DCE: Directory Services.`
`osi_paddr`	For an object with an OSI presentation address. The address contains the string encoding of an OSI Presentation Address as defined in *A string encoding of Presentation Address* (RFC 1278).

`onc_printers_bsaddr`	For a printer that understands the BSD print protocol. The address contains the machine name and printer name used by the protocol.
`onc_printers_use`	For a printer alias. The address contains a printer name.
`onc_printers_all`	For a list of printers that are enumerated using the "all" option. The address contains a list of printer names.
`onc_printers_location`	For a printer's location. The address format is unspecified.
`onc_printers_type`	For a printer's type. The address format is unspecified.
`onc_printers_speed`	For a printer's speed. The address format is unspecified.

SEE ALSO

`mount`(1M), `fn_ctx_handle_from_ref`(3XFN), `xfn`(3XFN), `fns`(5), `fns_policies`(5)

Hardcastle-Kille, S.E., *A string encoding of Presentation Address*, RFC 1278, University College London, November 1991.

NAME

fns_x500 – overview of FNS over X.500 implementation

DESCRIPTION

Federated Naming Service (FNS) provides a method for federating multiple naming services under a single, simple interface for the basic naming operations. One of the naming services supported by FNS is the X.500 Directory Service (see ITU-T X.500 or ISO / IEC 9594). X.500 is a global directory service. Its components cooperate to manage information about a hierarchy of objects on a worldwide scope. Such objects include countries, organizations, people, services, and machines. FNS uses X.500 to name entities globally.

FNS provides the XFN interface for retrieval and modification of information stored in X.500. In addition, enterprise namespaces such as those served by NIS+ and NIS can be federated with X.500 by adding reference information to X.500 describing how to reach the desired next naming service. To federate a NIS+ or NIS namespace under X.500, perform the following steps:

1. Obtain the root reference for the NIS+ hierarchy or NIS domain.
2. Enhance the X.500 schema to support the addition of XFN references.
3. Create an X.500 entry to store the XFN reference.
4. Add the XFN reference.

The root reference is referred to as the *next naming system reference* because it refers to the *next* naming system beneath X.500. This reference contains information about how to communicate with the NIS+ or NIS servers and has the following format:

<*domainname*> <*server name*> [<*server address*>]

where <*domainname*> is the fully qualified domain name. Notice that NIS+ and NIS have slightly different syntaxes for domain names. For NIS+, the fully qualified domain name is case-insensitive and terminated by a dot character ('`.`'). For NIS, the fully qualified domain name is case-sensitive and *not* terminated by a dot character. For both NIS+ and NIS, <*server address*> is optional. If it is not supplied, a host name lookup will be performed to get the machine's address.

For example, if the machine `wiz-nisplus-server` with address `133.33.33.33` serves the NIS+ domain `wiz.com.`, the reference would look like this:

`wiz.com. wiz-nisplus-server 133.33.33.33`

For another example, if the machine `woz-nis-server` serves the NIS domain `Woz.COM`, the reference would look like this:

```
Woz.COM woz-nis-server
```

Before the next naming system reference can be added to X.500, the X.500 schema must be altered to include the following object class and associated attributes (defined in ASN.1 notation).

```
xFNSupplement OBJECT-CLASS ::= {
  SUBCLASS OF { top }
  KIND          auxiliary
  MAY CONTAIN { objectReferenceString  |  nNSReference-
String }
  ID            id-oc-xFNSupplement }

id-oc-xFNSupplement OBJECT IDENTIFIER ::= {
  iso member-body(2) ansi(840) sun(113536) 25 }

objectReferenceString ATTRIBUTE ::= {
  WITH SYNTAX            OCTET STRING
  EQUALITY MATCHING RULE octetStringMatch
  SINGLE VALUE           TRUE
  ID                     id-at-objectReferenceString }

id-at-objectReferenceString OBJECT IDENTIFIER ::= {
  iso member-body(2) ansi(840) sun(113536) 30 }

nNSReferenceString ATTRIBUTE ::= {
  WITH SYNTAX            OCTET STRING
  EQUALITY MATCHING RULE octetStringMatch
  SINGLE VALUE           TRUE
  ID                     id-at-nNSReferenceString }

id-at-nNSReferenceString OBJECT IDENTIFIER ::= {
  so member-body(2) ansi(840) sun(113536) 31 }
```

The procedures for altering the X.500 schema will vary from implementation to implementation. Consult *Solstice X.500* or the schema administration guide for your X.500 product.

Once X.500 supports XFN references, the next naming system reference can be added by first creating an X.500 object and then adding the new reference to it. For example, the following commands create entries for the `Wiz` and `Woz` organizations in the U.S.A. and add the reference information shown in the examples above to them.

For NIS+:

```
example% fnattr .../c=us/o=wiz -a objectclass \
        top organization xfnsupplement

example% fnbind -r .../c=us/o=wiz/ onc_fn_enterprise \
        onc_fn_nisplus_root "wiz.com. wiz-nisplus-server"
```

For NIS:

```
example% fnattr .../c=us/o=woz -a objectclass \
        top organization xfnsupplement
```

```
example% fnbind -r .../c=us/o=woz/ onc_fn_enterprise \
        onc_fn_nis_root "Woz.COM woz-nis-server"
```

Notice the mandatory trailing slash ('/') in the name argument to fnbind(1).

This modification effectively adds the next naming system reference to X.500. The reference may be retrieved using fnlookup(1) to see if the information has been added properly. For example, the following command looks up the next naming system reference of the Wiz organization:

```
example% fnlookup -v .../c=us/o=wiz/
```

Note the mandatory trailing slash.

After this administrative step has been taken, clients outside of the NIS+ hierarchy or NIS domain can access and perform operations on the contexts in the NIS+ hierarchy or NIS domain. Foreign NIS+ clients access the hierarchy as unauthenticated NIS+ clients. Continuing the example above, and assuming that NIS+ is federated underneath the Wiz organization, the root of the NIS+ enterprise may be listed using the command:

```
example% fnlist .../c=us/o=wiz/
```

Note the mandatory trailing slash.

The next naming system reference may be removed using the command:

```
example% fnunbind .../c=us/o=wiz/
```

Note the mandatory trailing slash.

SEE ALSO

fnattr(1), fnbind(1), fnlist(1), fnlookup(1), nis+(1), ypserv(1M), xfn(3XFN), fns(5), fns_dns(5), fns_nis(5), fns_nis+(5), fns_references(5)

Solstice X.500

NOTES

In a 64-bit XFN application, retrieval and modification of information stored in the X.500 directory service is not supported.

NAME

formats – file format notation

DESCRIPTION

Utility descriptions use a syntax to describe the data organization within files—stdin, stdout, stderr, input files, and output files—when that organization is not otherwise obvious. The syntax is similar to that used by the `printf`(3C) function. When used for stdin or input file descriptions, this syntax describes the format that could have been used to write the text to be read, not a format that could be used by the `scanf`(3C) function to read the input file.

Format

The description of an individual record is as follows:

"<`format`>", [<*arg1*>, <*arg2*>, . . ., <*argn*>]

The `format` is a character string that contains three types of objects defined below:

characters	Characters that are not *escape sequences* or *conversion specifications*, as described below, are copied to the output.
escape sequences	Represent non-graphic characters.
conversion specifications	Specifies the output format of each argument. (See below.)

The following characters have the following special meaning in the format string:

„ “	(An empty character position.) One or more blank characters.
/ \	Exactly one space character.

The notation for spaces allows some flexibility for application output. Note that an empty character position in `format` represents one or more blank characters on the output (not *white space*, which can include newline characters). Therefore, another utility that reads that output as its input must be prepared to parse the data using `scanf`(3C), `awk`(1), and so forth. The character is used when exactly one space character is output.

Escape Sequences

The following table lists escape sequences and associated actions on display devices capable of the action.

Sequence	Character	Terminal Action
`\\`	backslash	None.
`\a`	alert	Attempts to alert the user through audible or visible notification.
`\b`	backspace	Moves the printing position to one column before the current position, unless the current position is the start of a line.

Sequence	Character	Terminal Action
`\f`	form-feed	Moves the printing position to the initial printing position of the next logical page.
`\n`	newline	Moves the printing position to the start of the next line.
`\r`	carriage-return	Moves the printing position to the start of the current line.
`\t`	tab	Moves the printing position to the next tab position on the current line. If there are no more tab positions left on the line, the behavior is undefined.
`\v`	vertical-tab	Moves the printing position to the start of the next vertical tab position. If there are no more vertical tab positions left on the page, the behavior is undefined.

Conversion Specifications

Each conversion specification is introduced by the percent-sign character (%). After the character %, the following appear in sequence:

flags — Zero or more *flags*, in any order, that modify the meaning of the conversion specification.

field width — An optional string of decimal digits to specify a minimum *field width*. For an output field, if the converted value has fewer bytes than the field width, it is padded on the left (or right, if the left-adjustment flag (–), described below, has been given to the field width).

precision — Gives the minimum number of digits to appear for the d, o, i, u, x or X conversions (the field is padded with leading zeros), the number of digits to appear after the radix character for the e and f conversions, the maximum number of significant digits for the g conversion; or the maximum number of bytes to be written from a string in s conversion. The precision takes the form of a period (.) followed by a decimal digit string; a null digit string is treated as zero.

conversion characters — A conversion character (see below) that indicates the type of conversion to be applied.

flags

The *flags* and their meanings are:

–	The result of the conversion is left-justified within the field.
+	The result of a signed conversion always begins with a sign (+ or –).
<space>	If the first character of a signed conversion is not a sign, a space character is prefixed to the result. This means that if the space character and + flags both appear, the space character flag is ignored.
#	The value is to be converted to an alternative form. For c, d, i, u, and s conversions, the behaviour is undefined. For o conversion, it increases the precision to force the first digit of the result to be a zero. For x or X conversion, a non-zero result has 0x or 0X prefixed to it, respectively. For e, E, f, g, and G conversions, the result always contains a radix character, even if no digits follow the radix character. For g and G conversions, trailing zeros are not removed from the result as they usually are.
0	For d, i, o, u, x, X, e, E, f, g, and G conversions, leading zeros (following any indication of sign or base) are used to pad to the field width; no space padding is performed. If the 0 and – flags both appear, the 0 flag is ignored. For d, i, o, u, x and X conversions, if a precision is specified, the 0 flag is ignored. For other conversions, the behaviour is undefined.

Conversion Characters

Each conversion character results in fetching zero or more arguments. The results are undefined if there are insufficient arguments for the format. If the format is exhausted while arguments remain, the excess arguments are ignored.

The *conversion characters* and their meanings are:

d,i,o,u,x,X The integer argument is written as signed decimal (d or i), unsigned octal (o), unsigned decimal (u), or unsigned hexadecimal notation (x and X). The d and i specifiers convert to signed decimal in the style [–]*dddd*. The x conversion uses the numbers and letters 0123456789abcdef and the X conversion uses the numbers and letters 0123456789ABCDEF. The *precision* component of the argument specifies the minimum number of digits to appear. If the value being converted can be represented in fewer digits than the specified minimum, it is expanded with leading zeros. The default precision is 1. The result of converting a zero value with a precision of 0 is no characters. If both the field width and precision are omitted, the implementation may precede, follow or precede and follow numeric arguments of types d, i and u with blank characters; arguments of type o (octal) may be preceded with leading zeros.

The treatment of integers and spaces is different from the `printf`(3C) function in that they can be surrounded with blank characters. This was done so that, given a format such as:

"%d\n",<*foo*>

the implementation could use a `printf()` call such as:

printf("%6d\n", *foo*);

and still conform. This notation is thus somewhat like `scanf()` in addition to `printf()`.

f The floating point number argument is written in decimal notation in the style `[−]`*ddd.ddd*, where the number of digits after the radix character (shown here as a decimal point) is equal to the *precision* specification. The `LC_NUMERIC` locale category determines the radix character to use in this format. If the *precision* is omitted from the argument, six digits are written after the radix character; if the *precision* is explicitly 0, no radix character appears.

e,E The floating point number argument is written in the style `[−]`*d.ddd*e±`dd` (the symbol ± indicates either a plus or minus sign), where there is one digit before the radix character (shown here as a decimal point) and the number of digits after it is equal to the precision. The `LC_NUMERIC` locale category determines the radix character to use in this format. When the precision is missing, six digits are written after the radix character; if the precision is 0, no radix character appears. The E conversion character produces a number with E instead of e introducing the exponent. The exponent always contains at least two digits. However, if the value to be written requires an exponent greater than two digits, additional exponent digits are written as necessary.

g,G The floating point number argument is written in style f or e (or in style E in the case of a G conversion character), with the precision specifying the number of significant digits. The style used depends on the value converted: style g is used only if the exponent resulting from the conversion is less than –4 or greater than or equal to the precision. Trailing zeros are removed from the result. A radix character appears only if it is followed by a digit.

c The integer argument is converted to an `unsigned char` and the resulting byte is written.

s The argument is taken to be a string and bytes from the string are written until the end of the string or the number of bytes indicated by the *precision* specification of the argument is reached. If the precision is omitted from the argument, it is taken to be infinite, so all bytes up to the end of the string are written.

% Write a % character; no argument is converted.

In no case does a non-existent or insufficient *field width* cause truncation of a field; if the result of a conversion is wider than the field width, the field is simply expanded to contain the conversion result. The term *field width* should not be confused with the term *precision* used in the description of %s.

One difference from the C function `printf()` is that the l and h conversion characters are not used. There is no differentiation between decimal values for type `int`, type `long`, or type `short`. The specifications %d or %i should be interpreted as an arbitrary length sequence of digits. Also, no distinction is made between single precision and double precision numbers (`float` or `double` in C). These are simply referred to as floating point numbers.

Many of the output descriptions use the term `line`, such as:

"%s", <*input line*>

Since the definition of `line` includes the trailing newline character already, there is no need to include a `\n` in the format; a double newline character would otherwise result.

EXAMPLES

EXAMPLE 1 To represent the output of a program that prints a date and time in the form Sunday, July 3, 10:02, where <*weekday*> and <*month*> are strings:

"%s, / \ %s / \ %d, / \ %d:%.2d \ n",<*weekday*>,<*month*>,<*day*>,<*hour*>,<*min*>

EXAMPLE 2 To show pi written to 5 decimal places:

"pi / \ = / \%.5f \ n",<*value of pi*>

EXAMPLE 3 To show an input file format consisting of five colon-separated fields:

"%s:%s:%s:%s:%s \ n",<*arg1*>,<*arg2*>,<*arg3*>,<*arg4*>,<*arg5*>

SEE ALSO `awk`(1), `printf`(1), `printf`(3C), `scanf`(3C)

NAME

iconv_1250 – code set conversion tables for MS 1250 (Windows Latin 2)

DESCRIPTION

The following code set conversions are supported:

Code Set Conversions Supported				
Code	**Symbol**	**Target Code**	**Symbol**	**Target Output**
MS 1250	win2	ISO 8859-2	iso2	ISO Latin 2
MS 1250	win2	MS 852	dos2	MS-DOS Latin 2
MS 1250	win2	Mazovia	maz	Mazovia
MS 1250	win2	DHN	dhn	Dom Handlowy Nauki

CONVERSIONS

The conversions are performed according to the following tables. All values in the tables are given in octal.

MS 1250 to ISO 8859-2

For the conversion of MS 1250 to ISO 8859-2, all characters not in the following table are mapped unchanged.

Conversions Performed			
MS 1250	**ISO 8859-2**	**MS 1250**	**ISO 8859-2**
24-211	40	235	273
212	251	236	276
213	40	237	274
214	246	241	267
215	253	245	241
216	256	246-267	40
217	254	271	261
221-231	40	273	40
232	271	274	245
233	40	276	265
234	266	247	365

MS 1250 to MS 852

For the conversion of MS 1250 to MS 852, all characters not in the following table are mapped unchanged.

Conversions Performed			
MS 1250	**MS 852**	**MS 1250**	**MS 852**
200-211	40	311	220
212	346	312	250
213	40	313	323
214	227	314	267
215	233	315	326
216	246	316	327
217	215	317	322
220-231	40	320	321
232	347	321	343
233	40	322	325
234	230	323	340
235	234	324	342
236	247	325	212
237	253	326	231
240	377	327	236
241	363	330	374
242	364	331	336
243	235	332	351
244	317	333	353
245	244	334	232
246	40	335	355
247	365	336	335
250	371	337	341
251	40	340	352
252	270	341	240
253	256	342	203
254	252	343	307
255	360	344	204
256	40	345	222

Conversions Performed			
MS 1250	**MS 852**	**MS 1250**	**MS 852**
257	275	346	206
260	370	347	207
261	40	350	237
262	362	351	202
263	210	352	251
264	357	353	211
265-267	40	354	330
270	367	355	241
271	245	356	214
272	255	357	324
273	257	360	320
274	225	361	344
275	361	362	345
276	226	363	242
277	276	364	223
300	350	365	213
301	265	366	224
302	266	367	366
303	306	370	375
304	216	371	205
305	221	372	243
306	217	374	201
307	200	375	354
310	254	376	356

MS 1250 to Mazovia

For the conversion of MS 1250 to Mazovia, all characters not in the following table are mapped unchanged.

Conversions Performed			
MS 1250	**Mazovia**	**MS 1250**	**Mazovia**
200-213	40	310-311	40
214	230	312	220
215-216	40	313-320	40
217	240	321	245
220-233	40	322	40
234	236	323	243
235-236	40	324-325	40
237	246	326	231
240	377	327-333	40
241-242	40	334	232
243	234	335-336	40
244	40	337	341
245	217	340-341	40
246-252	40	342	203
253	256	343	40
254	252	344	204
255-256	40	345	40
257	241	346	215
260	370	347	207
261	361	350	40
262	40	351	202
263	222	352	221
264	40	353	211
265	346	354-355	40
266	40	356	214
267	372	357-360	40
270	40	361	244
271	206	362	40
272	40	363	242

Conversions Performed			
MS 1250	**Mazovia**	**MS 1250**	**Mazovia**
273	257	364	223
274-276	40	365	40
277	247	366	224
300-303	40	367	366
304	216	370-373	40
305	40	374	201
306	225	375-376	40
307	200		

MS 1250 to DHN

For the conversion of MS 1250 to DHN, all characters not in the following table are mapped unchanged.

Conversions Performed			
MS 1250	**DHN**	**MS 1250**	**DHN**
200-213	40	306	201
214	206	307-311	40
215-216	40	312	202
217	207	313-320	40
220-233	40	321	204
234	217	322	40
235-236	40	323	205
237	220	324-325	40
240	377	326	231
241-242	40	327-333	40
243	203	334	232
244	40	335-336	40
245	200	337	341
246-252	40	340	40
253	256	341	240
254	252	342-345	40
255-256	40	346	212

Conversions Performed			
MS 1250	**DHN**	**MS 1250**	**DHN**
257	210	347-351	40
260	370	352	213
261	361	353-354	40
262	40	355	241
263	214	356-360	40
264	40	361	215
265	346	362	40
266	40	363	216
267	372	364	223
270	40	365	40
271	211	366	224
272	40	367	366
273	257	370-371	40
274-276	40	372	243
277	221	373-376	40
300-305	40		

FILES

`/usr/lib/iconv/*.so` conversion modules

`/usr/lib/iconv/*.t` conversion tables

`/usr/lib/iconv/iconv_data` list of conversions supported by conversion tables

SEE ALSO

`iconv`(1), `iconv`(3C), `iconv`(5)

NAME

iconv_1251 – code set conversion tables for MS 1251 (Windows Cyrillic)

DESCRIPTION

The following code set conversions are supported:

Code Set Conversions Supported				
Code	**Symbol**	**Target Code**	**Symbol**	**Target Output**
MS 1251	win5	ISO 8859-5	iso5	ISO 8859-5 Cyrillic
MS 1251	win5	KOI8-R	koi8	KOI8-R
MS 1251	win5	PC Cyrillic	alt	Alternative PC Cyrillic
MS 1251	win5	Mac Cyrillic	mac	Macintosh Cyrillic

CONVERSIONS

The conversions are performed according to the following tables. All values in the tables are given in octal.

MS 1251 to ISO 8859-5

For the conversion of MS 1251 to ISO 8859-5, all characters not in the following table are mapped unchanged.

Conversions Performed			
MS 1251	**ISO 8859-5**	**MS 1251**	**ISO 8859-5**
24	4	310	270
200	242	311	271
201	243	312	272
202	40	313	273
203	363	314	274
204-207	40	315	275
210	255	316	276
211	40	317	277
212	251	320	300
213	40	321	301
214	252	322	302
215	254	323	303
216	253	324	304
217	257	325	305
220	362	326	306
221-227	40	327	307
230	255	330	310

Conversions Performed			
MS 1251	**ISO 8859-5**	**MS 1251**	**ISO 8859-5**
231	40	331	311
232	371	332	312
233	40	333	313
234	372	334	314
235	374	335	315
236	373	336	316
237	377	337	317
241	256	340	320
242	376	341	321
243	250	342	322
244-247	40	343	323
250	241	344	324
251	40	345	325
252	244	346	326
253-254	40	347	327
255	55	350	330
256	40	351	331
257	247	352	332
260-261	40	353	333
262	246	354	334
263	366	355	335
264-267	40	356	336
270	361	357	337
271	360	360	340
272	364	361	341
273	40	362	342
274	370	363	343
275	245	364	344
276	365	365	345

Conversions Performed			
MS 1251	**ISO 8859-5**	**MS 1251**	**ISO 8859-5**
277	367	366	346
300	260	367	347
301	261	370	350
302	262	371	351
303	263	372	352
304	264	373	353
305	265	374	354
306	266	375	355
307	267	376	356

MS 1251 to KOI8-R

For the conversion of MS 1251 to KOI8-R , all characters not in the following table are mapped unchanged.

Conversions Performed			
MS 1251	**KOI8-R**	**MS 1251**	**KOI8-R**
24	4	310	351
200	261	311	352
201	262	312	353
202	40	313	354
203	242	314	355
204-207	40	315	356
210	255	316	357
211	40	317	360
212	271	320	362
213	40	321	363
214	272	322	364
215	274	323	365
216	273	324	346
217	277	325	350
220	241	326	343
221-227	40	327	376

Conversions Performed			
MS 1251	**KOI8-R**	**MS 1251**	**KOI8-R**
230	255	330	373
231	40	331	375
232	251	332	377
233	40	333	371
234	252	334	370
235	254	335	374
236	253	336	340
237	257	337	361
241	276	340	301
242	256	341	302
243	270	342	327
244-247	40	343	307
250	263	344	304
251	40	345	305
252	264	346	326
253-254	40	347	332
255	55	350	311
256	40	351	312
257	267	352	313
260-261	40	353	314
262	266	354	315
263	246	355	316
264-267	40	356	317
270	243	357	320
271	260	360	322
272	244	361	323
273	40	362	324
274	250	363	325
275	265	364	306

Conversions Performed			
MS 1251	**KOI8-R**	**MS 1251**	**KOI8-R**
276	245	365	310
277	247	366	303
300	341	367	336
301	342	370	333
302	367	371	335
303	347	372	337
304	344	373	331
305	345	374	330
306	366	375	334
307	372	376	300

MS 1251 to PC Cyrillic

For the conversion of MS 1251 to PC Cyrillic, all characters not in the following table are mapped unchanged.

Conversions Performed			
MS 1251	**PC Cyrillic**	**MS 1251**	**PC Cyrillic**
24	4	332	232
200-207	40	333	233
210	260	334	234
211-227	40	335	235
230	260	336	236
231-247	40	337	237
250	360	340	240
251-254	40	341	241
255	55	342	242
256-267	40	343	243
270	361	344	244
271-277	40	345	245
300	200	346	246
301	201	347	247
302	202	350	250

Conversions Performed			
MS 1251	**PC Cyrillic**	**MS 1251**	**PC Cyrillic**
303	203	351	251
304	204	352	252
305	205	353	253
306	206	354	254
307	207	355	255
310	210	356	256
311	211	357	257
312	212	360	340
313	213	361	341
314	214	362	342
315	215	363	343
316	216	364	344
317	217	365	345
320	220	366	346
321	221	367	347
322	222	370	350
323	223	371	351
324	224	372	352
325	225	373	353
326	226	374	354
327	227	375	355
330	230	376	356
331	231		

MS 1251 to Mac Cyrillic

For the conversion of MS 1251 to Mac Cyrillic, all characters not in the following table are mapped unchanged.

Conversions Performed			
MS 1251	**Mac Cyrillic**	**MS 1251**	**Mac Cyrillic**
24	4	260	241
200	253	262	247
201	256	263	264
202	40	264	266
203	257	266	246
204	327	267	245
205	311	270	336
206	240	271	334
207-211	40	272	271
212	274	273	310
213	40	274	300
214	276	275	301
215	315	276	317
216	40	277	273
217	332	300	200
220	254	301	201
221	324	302	202
222	325	303	203
223	322	304	204
224	323	305	205
225	40	306	206
226	320	307	207
227	321	310	210
230	40	311	211
231	252	312	212
232	275	313	213
233	40	314	214
234	277	315	215
235	316	316	216

Conversions Performed			
MS 1251	**Mac Cyrillic**	**MS 1251**	**Mac Cyrillic**
236	40	317	217
237	333	320	220
240	312	321	221
241	330	322	222
242	331	323	223
243	267	324	224
244	377	325	225
245	242	326	226
246	40	327	227
247	244	330	230
250	335	331	231
252	270	332	232
253	307	333	233
254	302	334	234
255	55	335	235
256	250	336	236
257	272	337	237
355	316		

FILES

`/usr/lib/iconv/*.so`	conversion modules
`/usr/lib/iconv/*.t`	conversion tables
`/usr/lib/iconv/iconv_data`	list of conversions supported by conversion tables

SEE ALSO

iconv(1), iconv(3C), iconv(5)

NAME

iconv – code set conversion tables

DESCRIPTION

The following code set conversions are supported:

Code Set Conversions Supported				
Code	Symbol	Target Code	Symbol	Target Output
ISO 646	646	ISO 8859-1	8859	US ASCII
ISO 646de	646de	ISO 8859-1	8859	German
ISO 646da	646da	ISO 8859-1	8859	Danish
ISO 646en	646en	ISO 8859-1	8859	English ASCII
ISO 646es	646es	ISO 8859-1	8859	Spanish
ISO 646fr	646fr	ISO 8859-1	8859	French
ISO 646it	646it	ISO 8859-1	8859	Italian
ISO 646sv	646sv	ISO 8859-1	8859	Swedish
ISO 8859-1	8859	ISO 646	646	7 bit ASCII
ISO 8859-1	8859	ISO 646de	646de	German
ISO 8859-1	8859	ISO 646da	646da	Danish
ISO 8859-1	8859	ISO 646en	646en	English ASCII
ISO 8859-1	8859	ISO 646es	646es	Spanish
ISO 8859-1	8859	ISO 646fr	646fr	French
ISO 8859-1	8859	ISO 646it	646it	Italian
ISO 8859-1	8859	ISO 646sv	646sv	Swedish
ISO 8859-2	iso2	MS 1250	win2	Windows Latin 2
ISO 8859-2	iso2	MS 852	dos2	MS-DOS Latin 2
ISO 8859-2	iso2	Mazovia	maz	Mazovia
ISO 8859-2	iso2	DHN	dhn	Dom Handlowy Nauki
MS 1250	win2	ISO 8859-2	iso2	ISO Latin 2
MS 1250	win2	MS 852	dos2	MS-DOS Latin 2
MS 1250	win2	Mazovia	maz	Mazovia

Code Set Conversions Supported				
MS 1250	win2	DHN	dhn	Dom Handlowy Nauki
MS 852	dos2	ISO 8859-2	iso2	ISO Latin 2
MS 852	dos2	MS 1250	win2	Windows Latin 2
MS 852	dos2	Mazovia	maz	Mazovia
MS 852	dos2	DHN	dhn	Dom Handlowy Nauki

Code Set Conversions Supported				
Code	Symbol	Target Code	Symbol	Target Output
Mazovia	maz	ISO 8859-2	iso2	ISO Latin 2
Mazovia	maz	MS 1250	win2	Windows Latin 2
Mazovia	maz	MS 852	dos2	MS-DOS Latin 2
Mazovia	maz	DHN	dhn	Dom Handlowy Nauki
DHN	dhn	ISO 8859-2	iso2	ISO Latin 2
DHN	dhn	MS 1250	win2	Windows Latin 2
DHN	dhn	MS 852	dos2	MS-DOS Latin 2
DHN	dhn	Mazovia	maz	Mazovia
ISO 8859-5	iso5	KOI8-R	koi8	KOI8-R
ISO 8859-5	iso5	PC Cyrillic	alt	Alternative PC Cyrillic
ISO 8859-5	iso5	MS 1251	win5	Windows Cyrillic
ISO 8859-5	iso5	Mac Cyrillic	mac	Macintosh Cyrillic

Code Set Conversions Supported				
KOI8-R	koi8	ISO 8859-5	iso5	ISO 8859-5 Cyrillic
KOI8-R	koi8	PC Cyrillic	alt	Alternative PC Cyrillic
KOI8-R	koi8	MS 1251	win5	Windows Cyrillic
KOI8-R	koi8	Mac Cyrillic	mac	Macintosh Cyrillic
PC Cyrillic	alt	ISO 8859-5	iso5	ISO 8859-5 Cyrillic
PC Cyrillic	alt	KOI8-R	koi8	KOI8-R
PC Cyrillic	alt	MS 1251	win5	Windows Cyrillic
PC Cyrillic	alt	Mac Cyrillic	mac	Macintosh Cyrillic
MS 1251	win5	ISO 8859-5	iso5	ISO 8859-5 Cyrillic
MS 1251	win5	KOI8-R	koi8	KOI8-R
MS 1251	win5	PC Cyrillic	alt	Alternative PC Cyrillic
MS 1251	win5	Mac Cyrillic	mac	Macintosh Cyrillic
Mac Cyrillic	mac	ISO 8859-5	iso5	ISO 8859-5 Cyrillic
Mac Cyrillic	mac	KOI8-R	koi8	KOI8-R
Mac Cyrillic	mac	PC Cyrillic	alt	Alternative PC Cyrillic
Mac Cyrillic	mac	MS 1251	win5	Windows Cyrillic

CONVERSIONS

The conversions are performed according to the tables contained in the manual pages cross-referenced in the `Index of Conversion Code Tables` below.

Index of Conversion Code Tables		
Code	Target Code	See Manual Page
ISO 646	ISO 8859-1	iconv_646 (5)

Index of Conversion Code Tables		
ISO 646de	ISO 8859-1	
ISO 646da	ISO 8859-1	
ISO 646en	ISO 8859-1	
ISO 646es	ISO 8859-1	
ISO 646fr	ISO 8859-1	
ISO 646it	ISO 8859-1	
ISO 646sv	ISO 8859-1	
ISO 8859-1	ISO 646	iconv_8859-1 (5)
ISO 8859-1	ISO 646de	
ISO 8859-1	ISO 646da	
ISO 8859-1	ISO 646en	
ISO 8859-1	ISO 646es	
ISO 8859-1	ISO 646fr	
ISO 8859-1	ISO 646it	
ISO 8859-1	ISO 646sv	
ISO 8859-2	MS 1250	iconv_8859-2 (5)
ISO 8859-2	MS 852	
ISO 8859-2	Mazovia	
ISO 8859-2	DHN	
MS 1250	ISO 8859-2	iconv_1250 (5)
MS 1250	MS 852	
MS 1250	Mazovia	
MS 1250	DHN	
MS 852	ISO 8859-2	iconv_852 (5)
MS 852	MS 1250	
MS 852	Mazovia	
MS 852	DHN	
Mazovia	ISO 8859-2	iconv_maz (5)
Mazovia	MS 1250	

Index of Conversion Code Tables		
Mazovia	MS 852	
Mazovia	DHN	

Index of Conversion Code Tables		
Code	Target Code	See Manual Page
DHN	ISO 8859-2	iconv_dhn (5)
DHN	MS 1250	
DHN	MS 852	
DHN	Mazovia	
ISO 8859-5	KOI8-R	iconv_8859-5 (5)
ISO 8859-5	PC Cyrillic	
ISO 8859-5	MS 1251	
ISO 8859-5	Mac Cyrillic	
KOI8-R	ISO 8859-5	iconv_koi8-r (5)
KOI8-R	PC Cyrillic	
KOI8-R	MS 1251	
KOI8-R	Mac Cyrillic	
PC Cyrillic	ISO 8859-5	iconv_pc_cyr (5)
PC Cyrillic	KOI8-R	
PC Cyrillic	MS 1251	
PC Cyrillic	Mac Cyrillic	
MS 1251	ISO 8859-5	iconv_1251 (5)
MS 1251	KOI8-R	
MS 1251	PC Cyrillic	
MS 1251	Mac Cyrillic	
Mac Cyrillic	ISO 8859-5	iconv_mac_cyr (5)
Mac Cyrillic	KOI8-R	
Mac Cyrillic	PC Cyrillic	
Mac Cyrillic	MS 1251	

FILES

/usr/lib/iconv/*.so	conversion modules
/usr/lib/iconv/*.t	conversion tables
/usr/lib/iconv/iconv_data	list of conversions supported by conversion tables

SEE ALSO

iconv(1), iconv(3C), iconv_1250(5), iconv_1251(5), iconv_646(5), iconv_852(5), iconv_8859-1(5), iconv_8859-2(5), iconv_8859-5(5), iconv_dhn(5), iconv_koi8-r(5), iconv_mac_cyr(5), iconv_maz(5), iconv_pc_cyr(5), iconv_unicode(5)

NAME

iconv_646 – code set conversion tables for ISO 646

DESCRIPTION

The following code set conversions are supported:

Code Set Conversions Supported				
Code	**Symbol**	**Target Code**	**Symbol**	**Target Output**
ISO 646	646	ISO 8859-1	8859	US ASCII
ISO 646de	646de	ISO 8859-1	8859	German
ISO 646da	646da	ISO 8859-1	8859	Danish
ISO 646en	646en	ISO 8859-1	8859	English ASCII
ISO 646es	646es	ISO 8859-1	8859	Spanish
ISO 646fr	646fr	ISO 8859-1	8859	French
ISO 646it	646it	ISO 8859-1	8859	Italian
ISO 646sv	646sv	ISO 8859-1	8859	Swedish

CONVERSIONS

The conversions are performed according to the following tables. All values in the tables are given in octal.

ISO 646 (US ASCII) to ISO 8859-1

For the conversion of ISO 646 to ISO 8859-1, all characters in ISO 646 can be mapped unchanged to ISO 8859-1

ISO 646de (GERMAN) to ISO 8859-1

For the conversion of ISO 646de to ISO 8859-1, all characters not in the following table are mapped unchanged.

Conversions Performed			
ISO 646de	**ISO 8859-1**	**ISO 646de**	**ISO 8859-1**
100	247	173	344
133	304	174	366
134	326	175	374
135	334	176	337

ISO 646da (DANISH) to ISO 8859-1

For the conversion of ISO 646da to ISO 8859-1, all characters not in the following table are mapped unchanged.

Conversions Performed			
ISO 646da	**ISO 8859-1**	**ISO 646da**	**ISO 8859-1**
133	306	173	346
134	330	174	370
135	305	175	345

ISO 646en (ENGLISH ASCII) to ISO 8859-1

For the conversion of ISO 646en to ISO 8859-1, all characters not in the following table are mapped unchanged.

Conversions Performed	
ISO 646en	**ISO 8859-1**
043	243

ISO 646es (SPANISH) to ISO 8859-1

For the conversion of ISO 646es to ISO 8859-1, all characters not in the following table are mapped unchanged.

Conversions Performed			
ISO 646es	**ISO 8859-1**	**ISO 646es**	**ISO 8859-1**
100	247	173	260
133	241	174	361
134	321	175	347
135	277		

ISO 646fr (FRENCH) to ISO 8859-1

For the conversion of ISO 646fr to ISO 8859-1, all characters not in the following table are mapped unchanged.

Conversions Performed			
ISO 646fr	**ISO 8859-1**	**ISO 646fr**	**ISO 8859-1**
043	243	173	351
100	340	174	371
133	260	175	350
134	347	176	250
135	247		

ISO 646it (ITALIAN) to ISO 8859-1

For the conversion of ISO 646it to ISO 8859-1, all characters not in the following table are mapped unchanged.

Conversions Performed			
ISO 646it	**ISO 8859-1**	**ISO 646it**	**ISO 8859-1**
043	243	140	371
100	247	173	340
133	260	174	362
134	347	175	350
135	351	176	354

ISO 646sv (SWEDISH) to ISO 8859-1

For the conversion of ISO 646sv to ISO 8859-1, all characters not in the following table are mapped unchanged.

Conversions Performed			
ISO 646sv	**ISO 8859-1**	**ISO 646sv**	**ISO 8859-1**
100	311	140	351
133	304	173	344
134	326	174	366
135	305	175	345
136	334	176	374

FILES

`/usr/lib/iconv/*.so`	conversion modules
`/usr/lib/iconv/*.t`	conversion tables
`/usr/lib/iconv/iconv_data`	list of conversions supported by conversion tables

SEE ALSO

`iconv`(1), `iconv`(3C), `iconv`(5)

NAME

iconv_852 – code set conversion tables for MS 852 (MS-DOS Latin 2)

DESCRIPTION

The following code set conversions are supported:

Code Set Conversions Supported				
Code	**Symbol**	**Target Code**	**Symbol**	**Target Output**
MS 852	dos2	ISO 8859-2	iso2	ISO Latin 2
MS 852	dos2	MS 1250	win2	Windows Latin 2
MS 852	dos2	Mazovia	maz	Mazovia
MS 852	dos2	DHN	dhn	Dom Handlowy Nauki

CONVERSIONS

The conversions are performed according to the following tables. All values in the tables are given in octal.

MS 852 to ISO 8859-2

For the conversion of MS 852 to ISO 8859-2, all characters not in the following table are mapped unchanged.

Conversions Performed			
MS 852	**ISO 8859-2**	MS 852	**ISO 8859-2**
24-177	40	271-274	40
200	307	275	257
201	374	276	277
202	351	277-305	40
203	342	306	303
204	344	307	343
205	371	310-316	40
206	346	317	244
207	347	320	360
210	263	321	320
211	353	322	317
212	325	323	313
213	365	324	357
214	356	325	322
215	254	326	315
216	304	327	316
217	306	330	354

Conversions Performed			
MS 852	**ISO 8859-2**	MS 852	**ISO 8859-2**
220	311	331-334	40
221	305	335	336
222	345	336	331
223	364	337	40
224	366	340	323
225	245	341	337
226	265	342	324
227	246	343	321
230	266	344	361
231	326	345	362
232	334	346	251
233	253	347	271
234	273	350	300
235	243	351	332
236	327	352	340
237	350	353	333
240	341	354	375
241	355	355	335
242	363	356	376
243	372	357	264
244	241	360	255
245	261	361	275
246	256	362	262
247	276	363	267
250	312	364	242
251	352	365	247
252	40	366	367
253	274	367	270
254	310	370	260

Conversions Performed			
MS 852	**ISO 8859-2**	MS 852	**ISO 8859-2**
255	272	371	250
256-264	40	372	377
265	301	374	330
266	302	375	370
267	314	376	40
270	252		

MS 852 to MS 1250

For the conversion of MS 852 to MS 1250, all characters not in the following table are mapped unchanged.

Conversions Performed			
MS 852	**MS 1250**	**MS 852**	**MS 1250**
200	307	270	252
201	374	271-274	40
202	351	275	257
203	342	276	277
204	344	277-305	40
205	371	306	303
206	346	307	343
207	347	310-316	40
210	263	317	244
211	353	320	360
212	325	321	320
213	365	322	317
214	356	323	313
215	217	324	357
216	304	325	322
217	306	326	315
220	311	327	316
221	305	330	354
222	345	331-334	40

Conversions Performed			
MS 852	**MS 1250**	**MS 852**	**MS 1250**
223	364	335	336
224	366	336	331
225	274	337	40
226	276	340	323
227	214	341	337
230	234	342	324
231	326	343	321
232	334	344	361
233	215	345	362
234	235	346	212
235	243	347	232
236	327	350	300
237	350	351	332
240	341	352	340
241	355	353	333
242	363	354	375
243	372	355	335
244	245	356	376
245	271	357	264
246	216	360	255
247	236	361	275
250	312	362	262
251	352	363	241
252	254	364	242
253	237	365	247
254	310	366	367
255	272	367	270
256	253	370	260
257	273	371	250

Conversions Performed			
MS 852	**MS 1250**	**MS 852**	**MS 1250**
260-264	40	372	377
265	301	374	330
266	302	375	370
267	314	376	40

MS 852 to Mazovia

For the conversion of MS 852 to Mazovia, all characters not in the following table are mapped unchanged.

Conversions Performed			
MS 852	**Mazovia**	**MS 852**	**Mazovia**
205	40	246-247	40
206	215	250	220
210	222	251	221
212-213	40	253	246
215	240	254-270	40
217	225	275	241
220-226	40	276	247
227	230	306-336	40
230	236	340	243
233-234	40	342	40
235	234	343	245
236-243	40	344	244
244	217	345-375	40
245	206		

MS 852 to DHN

For the conversion of MS 852 to DHN, all characters not in the following table are mapped unchanged.

Conversions Performed			
MS 852	**DHN**	**MS 852**	**DHN**
200-205	40	244	200
206	212	245	211
207	40	246-247	40
210	214	250	202
211-214	40	251	213
215	207	253	220
216	40	254-270	40
217	201	275	210
220-226	40	276	221
227	206	306-336	40
230	217	340	205
233-234	40	342	40
235	203	343	204
236-237	40	344	215
242	216	345-375	40
252	254		

FILES

`/usr/lib/iconv/*.so` conversion modules

`/usr/lib/iconv/*.t` conversion tables

`/usr/lib/iconv/iconv_data` list of conversions supported by conversion tables

SEE ALSO

`iconv`(1), `iconv`(3C), `iconv`(5)

NAME

iconv_8859-1 – code set conversion tables for ISO 8859-1 (Latin 1)

DESCRIPTION

The following code set conversions are supported:

Code Set Conversions Supported				
Code	**Symbol**	**Target Code**	**Symbol**	**Target Output**
ISO 8859-1	8859	ISO 646	646	7 bit ASCII
ISO 8859-1	8859	ISO 646de	646de	German
ISO 8859-1	8859	ISO 646da	646da	Danish
ISO 8859-1	8859	ISO 646en	646en	English ASCII
ISO 8859-1	8859	ISO 646es	646es	Spanish
ISO 8859-1	8859	ISO 646fr	646fr	French
ISO 8859-1	8859	ISO 646it	646it	Italian
ISO 8859-1	8859	ISO 646sv	646sv	Swedish

CONVERSIONS

The conversions are performed according to the following tables. All values in the tables are given in octal.

ISO 8859-1 to ISO 646 (7-bit ASCII)

For the conversion of ISO 8859-1 to ISO 646, all characters not in the following table are mapped unchanged.

```
Converted to Underscore '_' (137)
---------------------------------
200 201 202 203 204 205 206 207
210 211 212 213 214 215 216 217
220 221 222 223 224 225 226 227
230 231 232 233 234 235 236 237
240 241 242 243 244 245 246 247
250 251 252 253 254 255 256 257
260 261 262 263 264 265 266 267
270 271 272 273 274 275 276 277
300 301 302 303 304 305 306 307
310 311 312 313 314 315 316 317
320 321 322 323 324 325 326 327
330 331 332 333 334 335 336 337
340 341 342 343 344 345 346 347
350 351 352 353 354 355 356 357
360 361 362 363 364 365 366 367
370 371 372 373 374 375 376 377
```

ISO 8859-1 to ISO 646de (GERMAN)

For the conversion of ISO 8859-1 to ISO 646de, all characters not in the following tables are mapped unchanged.

Conversions Performed			
ISO 8859-1	**ISO 646de**	**ISO 8859-1**	**ISO 646de**
247	100	337	176
304	133	344	173
326	134	366	174
334	135	374	175

```
Converted to Underscore '_' (137)
--------------------------------
100 133 134 135 173 174 175 176
200 201 202 203 204 205 206 207
210 211 212 213 214 215 216 217
220 221 222 223 224 225 226 227
230 231 232 233 234 235 236 237
240 241 242 243 244 245 246
250 251 252 253 254 255 256 257
260 261 262 263 264 265 266 267
270 271 272 273 274 275 276 277
300 301 302 303     305 306 307
310 311 312 313 314 315 316 317
320 321 322 323 324 325     327
330 331 332 333     335 336 337
340 341 342 343     345 346 347
350 351 352 353 354 355 356 357
360 361 362 363 364 365     367
370 371 372 373     375 376 377
```

ISO 8859-1 to ISO 646da (DANISH)

For the conversion of ISO 8859-1 to ISO 646da, all characters not in the following tables are mapped unchanged.

Conversions Performed			
ISO 8859-1	**ISO 646da**	**ISO 8859-1**	**ISO 646da**
305	135	345	175
306	133	346	173
330	134	370	174

```
Converted to Underscore '_' (137)
--------------------------------
133 134 135 173 174 175
200 201 202 203 204 205 206 207
210 211 212 213 214 215 216 217
220 221 222 223 224 225 226 227
230 231 232 233 234 235 236 237
240 241 242 243 244 245 246 247
250 251 252 253 254 255 256 257
```

```
260 261 262 263 264 265 266 267
270 271 272 273 274 275 276 277
300 301 302 303 304         307
310 311 312 313 314 315 316 317
320 321 322 323 324 325 326 327
    331 332 333 334 335 336 337
340 341 342 343 344         347
350 351 352 353 354 355 356 357
360 361 362 363 364 365 366 367
371 372 373 374         376 377
```

ISO 8859-1 to ISO 646en (ENGLISH ASCII)

For the conversion of ISO 8859-1 to ISO 646en, all characters not in the following tables are mapped unchanged.

Conversions Performed	
ISO 8859-1	**ISO 646en**
243	043

```
Converted to Underscore '_' (137)
---------------------------------
043
200 201 202 203 204 205 206 207
210 211 212 213 214 215 216 217
220 221 222 223 224 225 226 227
230 231 232 233 234 235 236 237
240 241 242     244 245 246 247
250 251 252 253 254 255 256 257
260 261 262 263 264 265 266 267
270 271 272 273 274 275 276 277
300 301 302 303 304 305 306 307
310 311 312 313 314 315 316 317
320 321 322 323 324 325 326 327
330 331 332 333 334 335 336 337
340 341 342 343 344 345 346 347
350 351 352 353 354 355 356 357
360 361 362 363 364 365 366 367
370 371 372 373 374 375 376 377
```

ISO 8859-1 to ISO 646fr (FRENCH)

For the conversion of ISO 8859-1 to ISO 646fr, all characters not in the following tables are mapped unchanged.

Conversions Performed			
ISO 8859-1	**ISO 646fr**	**ISO 8859-1**	**ISO 646fr**
243	043	347	134
247	135	350	175
250	176	351	173

Conversions Performed			
ISO 8859-1	**ISO 646fr**	**ISO 8859-1**	**ISO 646fr**
260	133	371	174
340	100		

```
Converted to Underscore '_' (137)
---------------------------------
043
100 133 134 135 173 174 175 176
200 201 202 203 204 205 206 207
210 211 212 213 214 215 216 217
220 221 222 223 224 225 226 227
230 231 232 233 234 235 236 237
240 241 242     244 245 246
    251 252 253 254 255 256 257
    261 262 263 264 265 266 267
270 271 272 273 274 275 276 277
300 301 302 303 304 305 306 307
310 311 312 313 314 315 316 317
320 321 322 323 324 325 326 327
330 331 332 333 334 335 336 337
    341 342 343 344 345 346
        352 353 354 355 356 357
360 361 362 363 364 365 366 367
370     372 373 374 375 376 377
```

ISO 8859-1 to ISO 646it (ITALIAN)

For the conversion of ISO 8859-1 to ISO 646it, all characters not in the following tables are mapped unchanged.

Conversions Performed			
ISO 8859-1	**ISO 646it**	**ISO 8859-1**	**ISO 646it**
243	043	350	175
247	100	351	135
260	133	354	176
340	173	362	174
347	134	371	140

```
Converted to Underscore '_' (137)
---------------------------------
043
100 133 134 135 173 174 175 176
200 201 202 203 204 205 206 207
210 211 212 213 214 215 216 217
220 221 222 223 224 225 226 227
230 231 232 233 234 235 236 237
```

```
240 241 242     244 245 246
250 251 252 253 254 255 256 257
    261 262 263 264 265 266 267
270 271 272 273 274 275 276 277
300 301 302 303 304 305 306 307
310 311 312 313 314 315 316 317
320 321 322 323 324 325 326 327
330 331 332 333 334 335 336 337
    341 342 343 344 345 346
        352 353 354 355 356 357
360 361 362 363 364 365 366 367
370     372 373 374 375 376 377
```

ISO 8859-1 to ISO 646es (SPANISH)

For the conversion of ISO 8859-1 to ISO 646es, all characters not in the following tables are mapped unchanged.

Conversions Performed			
ISO 8859-1	**ISO 646es**	**ISO 8859-1**	**ISO 646es**
241	133	321	134
247	100	347	175
260	173	361	174
277	135		

```
Converted to Underscore '_' (137)
---------------------------------
100 133 134 135 173 174 175
200 201 202 203 204 205 206 207
210 211 212 213 214 215 216 217
220 221 222 223 224 225 226 227
230 231 232 233 234 235 236 237
240     242 243 244 245 246
250 251 252 253 254 255 256 257
    261 262 263 264 265 266 267
270 271 272 273 274 275 276
300 301 302 303 304 305 306 307
310 311 312 313 314 315 316 317
320     322 323 324 325 326 327
330 331 332 333 334 335 336 337
340 341 342 343 344 345 346
350 351 352 353 354 355 356 357
360     362 363 364 365 366 367
370 371 372 373 374 375 376 377
```

ISO 8859-1 to ISO 646sv (SWEDISH)

For the conversion of ISO 8859-1 to ISO 646sv, all characters not in the following tables are mapped unchanged.

Conversions Performed			
ISO 8859-1	**ISO 646sv**	**ISO 8859-1**	**ISO 646sv**
304	133	344	173
305	135	345	175
311	100	351	140
326	134	366	174
334	136	374	176

```
Converted to Underscore '_' (137)
---------------------------------
100 133 134 135 136 140
173 174 175 176
200 201 202 203 204 205 206 207
210 211 212 213 214 215 216 217
220 221 222 223 224 225 226 227
230 231 232 233 234 235 236 237
240 241 242 243 244 245 246 247
250 251 252 253 254 255 256 257
260 261 262 263 264 265 266 267
270 271 272 273 274 275 276 277
300 301 302 303         306 307
310     312 313 314 315 316 317
320 321 322 323 324 325     327
330 331 332 333     335 336 337
340 341 342 343         346 347
350     352 353 354 355 356 357
360 361 362 363 364 365     367
370 371 372 373     375 376 377
```

FILES

`/usr/lib/iconv/*.so` conversion modules

`/usr/lib/iconv/*.t` conversion tables

`/usr/lib/iconv/iconv_data` list of conversions supported by conversion tables

SEE ALSO

`iconv`(1), `iconv`(3C), `iconv`(5)

NAME

iconv_8859-2 – code set conversion tables for ISO 8859-2 (Latin 2)

DESCRIPTION

The following code set conversions are supported:

Code Set Conversions Supported				
Code	**Symbol**	**Target Code**	**Symbol**	**Target Output**
ISO 8859-2	iso2	MS 1250	win2	Windows Latin 2
ISO 8859-2	iso2	MS 852	dos2	MS-DOS Latin 2
ISO 8859-2	iso2	Mazovia	maz	Mazovia
ISO 8859-2	iso2	DHN	dhn	Dom Handlowy Nauki

CONVERSIONS

The conversions are performed according to the following tables. All values in the tables are given in octal.

ISO 8859-2 to MS 1250

For the conversion of ISO 8859-2 to MS 1250, all characters not in the following table are mapped unchanged.

Conversions Performed			
ISO 8859-2	**MS 1250**	**ISO 8859-2**	**MS 1250**
24	4	261	271
177-237	40	265	276
241	245	266	234
245	274	267	241
246	214	271	232
251	212	273	235
253	215	274	237
254	217	276	236
256	216	266	236

ISO 8859-2 to MS 852

For the conversion of ISO 8859-2 to MS 852, all characters not in the following table are mapped unchanged.

Conversions Performed			
ISO 8859-2	**MS 852**	**ISO 8859-2**	**MS 852**
24	4	316	327
177-237	40	317	322
240	377	320	321

Conversions Performed			
ISO 8859-2	**MS 852**	**ISO 8859-2**	**MS 852**
241	244	321	343
242	364	322	325
243	235	323	340
244	317	324	342
245	225	325	212
246	227	326	231
247	365	327	236
250	371	330	374
251	346	331	336
252	270	332	351
253	233	333	353
254	215	334	232
255	360	335	355
256	246	336	335
257	275	337	341
260	370	340	352
261	245	341	240
262	362	342	203
263	210	343	307
264	357	344	204
265	226	345	222
266	230	346	206
267	363	347	207
270	367	350	237
271	347	351	202
272	255	352	251
273	234	353	211
274	253	354	330
275	361	355	241

Conversions Performed			
ISO 8859-2	**MS 852**	**ISO 8859-2**	**MS 852**
276	247	356	214
277	276	357	324
300	350	360	320
301	265	361	344
302	266	362	345
303	306	363	242
304	216	364	223
305	221	365	213
306	217	366	224
307	200	367	366
310	254	370	375
311	220	371	205
312	250	372	243
313	323	374	201
314	267	375	354
315	326	376	356
366	367		

ISO 8859-2 to Mazovia

For the conversion of ISO 8859-2 to Mazovia, all characters not in the following table are mapped unchanged.

Conversions Performed			
ISO 8859-2	**Mazovia**	**ISO 8859-2**	**Mazovia**
24	4	323	243
177-237	40	324-325	40
240	377	326	231
241	217	327-333	40
242	40	334	232
243	234	335-336	40
244-245	40	337	341
246	230	340-341	40

Conversions Performed			
ISO 8859-2	**Mazovia**	**ISO 8859-2**	**Mazovia**
247-253	40	342	203
254	240	343	40
255-256	40	344	204
257	241	345	40
260	370	346	215
261	206	347	207
262	40	350	40
263	222	351	202
264-265	40	352	221
266	236	353	211
267-273	40	354-355	40
274	246	356	214
275-276	40	357-360	40
277	247	361	244
300-303	40	362	40
304	216	363	242
305	40	364	223
306	225	365	40
307	200	366	224
310-311	40	367	366
312	220	370-373	40
313-320	40	374	201
321	245	375-376	40
322	40		

ISO 8859-2 to DHN

For the conversion of ISO 8859-2 to DHN, all characters not in the following table are mapped unchanged.

Conversions Performed			
ISO 8859-2	**DHN**	**ISO 8859-2**	**DHN**
24	4	322	40
177-237	40	323	205
240	377	324-325	40
241	200	326	231
242	40	327-333	40
243	203	334	232
244-245	40	335-336	40
246	206	337	341
247-253	40	340	40
254	207	341	240
255-256	40	342-345	40
257	210	346	212
260	370	347-351	40
261	211	352	213
262	40	353-354	40
263	214	355	241
264-265	40	356-360	40
266	217	361	215
267-273	40	362	40
274	220	363	216
275-276	40	364	223
277	221	365	40
300-305	40	366	224
306	201	367	366
307-311	40	370-371	40
312	202	372	243
313-320	40	373-376	40
321	204		

FILES

/usr/lib/iconv/*.so	conversion modules
/usr/lib/iconv/*.t	conversion tables
/usr/lib/iconv/iconv_data	list of conversions supported by conversion tables

SEE ALSO

iconv(1), iconv(3C), iconv(5)

NAME iconv_8859-5 – code set conversion tables for ISO 8859-5 (Cyrillic)

DESCRIPTION The following code set conversions are supported:

Code Set Conversions Supported				
Code	**Symbol**	**Target Code**	**Symbol**	**Target Output**
ISO 8859-5	iso5	KOI8-R	koi8	KOI8-R
ISO 8859-5	iso5	PC Cyrillic	alt	Alternative PC Cyrillic
ISO 8859-5	iso5	MS 1251	win5	Windows Cyrillic
ISO 8859-5	iso5	Mac Cyrillic	mac	Macintosh Cyrillic

CONVERSIONS The conversions are performed according to the following tables. All values in the tables are given in octal.

ISO 8859-5 to KOI8-R For the conversion of ISO 8859-5 to KOI8-R, all characters not in the following table are mapped unchanged.

Conversions Performed			
ISO 8859-5	**KOI8-R**	**ISO 8859-5**	**KOI8-R**
24	4	320	301
241	263	321	302
242	261	322	327
243	262	323	307
244	264	324	304
245	265	325	305
246	266	327	332
247	267	330	311
250	270	331	312
251	271	332	313
252	272	333	314
253	273	334	315
254	274	335	316

Conversions Performed			
ISO 8859-5	**KOI8-R**	**ISO 8859-5**	**KOI8-R**
256	276	336	317
257	277	337	320
260	341	340	322
261	342	341	323
262	367	342	324
263	347	343	325
264	344	344	306
265	345	345	310
266	366	346	303
267	372	347	336
270	351	350	333
271	352	351	335
272	353	352	337
273	354	353	331
274	355	354	330
275	356	355	334
276	357	356	300
277	360	357	321
300	362	360	260
301	363	361	243
302	364	362	241
303	365	363	242
304	346	364	244
305	350	365	245
306	343	366	246
307	376	367	247
310	373	370	250
311	375	371	251
312	377	372	252

Conversions Performed			
ISO 8859-5	**KOI8-R**	**ISO 8859-5**	**KOI8-R**
313	371	373	253
314	370	374	254
315	374	375	255
316	340	376	256
317	361		

ISO 8859-5 to PC Cyrillic

For the conversion of ISO 8859-5 to PC Cyrillic, all characters not in the following table are mapped unchanged.

Conversions Performed			
ISO 8859-5	**PC Cyrillic**	**ISO 8859-5**	**PC Cyrillic**
24	4	307	227
200-240	40	310	230
241	360	311	231
242-254	40	312	232
255	260	313	233
256-257	40	314	234
260	200	315	235
261	201	316	236
262	202	317	237
263	203	320	240
264	204	321	241
265	205	322	242
266	206	323	243
267	207	324	244
270	210	325	245
271	211	326	246
272	212	327	247
273	213	330	250
274	214	331	251
275	215	332	252

Conversions Performed			
ISO 8859-5	**PC Cyrillic**	**ISO 8859-5**	**PC Cyrillic**
276	216	333	253
277	217	334	254
300	220	335	255
301	221	336	256
302	222	337	257
303	223	360-374	40
304	224	375	260
305	225	376	40
306	226	365	40

ISO 8859-5 to MS 1251

For the conversion of ISO 8859-5 to MS 1251, all characters not in the following table are mapped unchanged.

Conversions Performed			
ISO 8859-5	**MS 1251**	**ISO 8859-5**	**MS 1251**
24	4	317	337
200-237	40	320	340
241	250	321	341
242	200	322	342
243	201	323	343
244	252	324	344
245	275	325	345
246	262	326	346
247	257	327	347
250	243	330	350
251	212	331	351
252	214	332	352
253	216	333	353
254	215	334	354
255	210	335	355
256	241	336	356

Conversions Performed			
ISO 8859-5	**MS 1251**	**ISO 8859-5**	**MS 1251**
257	217	337	357
260	300	340	360
261	301	341	361
262	302	342	362
263	303	343	363
264	304	344	364
265	305	345	365
266	306	346	366
267	307	347	367
270	310	350	370
271	311	351	371
272	312	352	372
273	313	353	373
274	314	354	374
275	315	355	375
276	316	356	376
277	317	357	377
300	320	360	271
301	321	361	270
302	322	362	220
303	323	363	203
304	324	364	272
305	325	365	276
306	326	366	263
307	327	367	277
310	330	370	274
311	331	371	232
312	332	372	234
313	333	373	236

Conversions Performed			
ISO 8859-5	**MS 1251**	**ISO 8859-5**	**MS 1251**
314	334	374	235
315	335	375	210
316	336	376	242
376	331		

ISO 8859-5 to Mac Cyrillic

For the conversion of ISO 8859-5 to Mac Cyrillic, all characters not in the following table are mapped unchanged.

Conversions Performed			
ISO 8859-5	**Mac Cyrillic**	**ISO 8859-5**	**Mac Cyrillic**
24	4	317	237
200-237	40	320	340
240	312	321	341
241	335	322	342
242	253	323	343
243	256	324	344
244	270	325	345
245	301	326	346
246	247	327	347
247	272	330	350
250	267	331	351
251	274	332	352
252	276	333	353
253	40	334	354
254	315	335	355
255	40	336	356
256	330	337	357
257	332	340	360
260	200	341	361
261	201	342	362
262	202	343	363

Conversions Performed			
ISO 8859-5	**Mac Cyrillic**	**ISO 8859-5**	**Mac Cyrillic**
263	203	344	364
264	204	345	365
265	205	346	366
266	206	347	367
267	207	350	370
270	210	351	371
271	211	352	372
272	212	353	373
273	213	354	374
274	214	355	375
275	215	356	376
276	216	357	337
277	217	360	334
300	220	361	336
301	221	362	254
302	222	363	257
303	223	364	271
304	224	365	317
305	225	366	264
306	226	367	273
307	227	370	300
310	230	371	275
311	231	372	277
312	232	373	40
313	233	374	316
314	234	375	40
315	235	376	331
316	236		

FILES

`/usr/lib/iconv/*.so`	conversion modules
`/usr/lib/iconv/*.t`	conversion tables
`/usr/lib/iconv/iconv_data`	list of conversions supported by conversion tables

SEE ALSO

iconv(1), iconv(3C), iconv(5)

NAME

iconv_dhn – code set conversion tables for DHN (Dom Handlowy Nauki)

DESCRIPTION

The following code set conversions are supported:

Code Set Conversions Supported				
Code	**Symbol**	**Target Code**	**Symbol**	**Target Output**
DHN	dhn	ISO 8859-2	iso2	ISO Latin 2
DHN	dhn	MS 1250	win2	Windows Latin 2
DHN	dhn	MS 852	dos2	MS-DOS Latin 2
DHN	dhn	Mazovia	maz	Mazovia

CONVERSIONS

The conversions are performed according to the following tables. All values in the tables are given in octal.

DHN to ISO 8859-2

For the conversion of DHN to ISO 8859-2, all characters not in the following table are mapped unchanged.

Conversions Performed			
DHN	**ISO 8859-2**	**DHN**	**ISO 8859-2**
24-177	40	222	40
200	241	223	364
201	306	224	366
202	312	225-230	40
203	243	231	326
204	321	232	334
205	323	233-237	40
206	246	240	341
207	254	241	355
210	257	242	363
211	261	243	372
212	346	244-340	40
213	352	341	337
214	263	342-365	40
215	361	366	367
216	363	367	40
217	266	370	260

Conversions Performed			
DHN	**ISO 8859-2**	**DHN**	**ISO 8859-2**
220	274	371-376	40
221	277		

DHN to MS 1250

For the conversion of DHN to MS 1250, all characters not in the following table are mapped unchanged.

Conversions Performed			
DHN	**MS 1250**	**DHN**	**MS 1250**
200	245	233-237	40
201	306	240	341
202	312	241	355
203	243	242	363
204	321	243	372
205	323	244-251	40
206	214	252	254
207	217	253-255	40
210	257	256	253
211	271	257	273
212	346	260-340	40
213	352	341	337
214	263	342-345	40
215	361	346	265
216	363	347-360	40
217	234	361	261
220	237	362-365	40
221	277	366	367
222	40	367	40
223	364	370	260
224	366	371	40
225-230	40	372	267

Conversions Performed			
DHN	**MS 1250**	**DHN**	**MS 1250**
231	326	373-376	40
232	334		

DHN to MS 852

For the conversion of DHN to MS 852, all characters not in the following table are mapped unchanged.

Conversions Performed			
DHN	**MS 852**	**DHN**	**MS 852**
200	244	212	206
201	217	213	251
202	250	214	210
203	235	215	344
204	343	216	242
205	340	217	230
206	227	220	253
207	215	221	276
210	275	222-375	40
211	245		

DHN to Mazovia

For the conversion of DHN to Mazovia, all characters not in the following table are mapped unchanged.

Conversions Performed			
DHN	**Mazovia**	**DHN**	**Mazovia**
200	217	212	215
201	225	213	221
202	220	214	222
203	234	215	244
204	245	216	242
205	243	217	236
206	230	220	246
207	240	221	247

Conversions Performed			
DHN	**Mazovia**	**DHN**	**Mazovia**
210	241	222-247	40
211	206		

FILES

/usr/lib/iconv/*.so conversion modules

/usr/lib/iconv/*.t conversion tables

/usr/lib/iconv/iconv_data list of conversions supported by conversion tables

SEE ALSO

iconv(1), iconv(3C), iconv(5)

NAME

iconv_koi8-r – code set conversion tables for KOI8-R

DESCRIPTION

The following code set conversions are supported:

Code Set Conversions Supported				
Code	**Symbol**	**Target Code**	**Symbol**	**Target Output**
KOI8-R	koi8	ISO 8859-5	iso5	ISO 8859-5 Cyrillic
KOI8-R	koi8	PC Cyrillic	alt	Alternative PC Cyrillic
KOI8-R	koi8	MS 1251	win5	Windows Cyrillic
KOI8-R	koi8	Mac Cyrillic	mac	Macintosh Cyrillic

CONVERSIONS

The conversions are performed according to the following tables. All values in the tables are given in octal.

KOI8-R to ISO 8859-5

For the conversion of KOI8-R to ISO 8859-5, all characters not in the following table are mapped unchanged.

Conversions Performed			
KOI8-R	**ISO 8859-5**	**KOI8-R**	**ISO 8859-5**
24	4	320	337
241	362	321	357
242	363	322	340
243	361	323	341
244	364	324	342
245	365	325	343
246	366	327	322
247	367	330	354
250	370	331	353
251	371	332	327
252	372	333	350
253	373	334	355
254	374	335	351
256	376	336	347
257	377	337	352
260	360	340	316
261	242	341	260

Conversions Performed			
KOI8-R	**ISO 8859-5**	**KOI8-R**	**ISO 8859-5**
262	243	342	261
263	241	343	306
264	244	344	264
265	245	345	265
266	246	346	304
267	247	347	263
270	250	350	305
271	251	351	270
272	252	352	271
273	253	353	272
274	254	354	273
275	255	355	274
276	256	356	275
277	257	357	276
300	356	360	277
301	320	361	317
302	321	362	300
303	346	363	301
304	324	364	302
305	325	365	303
306	344	366	266
307	323	367	262
310	345	370	314
311	330	371	313
312	331	372	267
313	332	373	310
314	333	374	315
315	334	375	311

Conversions Performed			
KOI8-R	**ISO 8859-5**	**KOI8-R**	**ISO 8859-5**
316	335	376	307
317	336		

KOI8-R to PC Cyrillic

For the conversion of KOI8-R to PC Cyrillic, all characters not in the following table are mapped unchanged.

Conversions Performed			
KOI8-R	**PC Cyrillic**	**KOI8-R**	**PC Cyrillic**
24	4	333	350
200-242	40	334	355
243	361	335	351
244-254	40	336	347
255	260	337	352
256-262	40	340	236
263	360	341	200
264-274	40	342	201
275	260	343	226
276-277	40	344	204
300	356	345	205
301	240	346	224
302	241	347	203
303	346	350	225
304	244	351	210
305	245	352	211
306	344	353	212
307	243	354	213
310	345	355	214
311	250	356	215
312	251	357	216
313	252	360	217
314	253	361	237

Conversions Performed			
KOI8-R	**PC Cyrillic**	**KOI8-R**	**PC Cyrillic**
315	254	362	220
316	255	363	221
317	256	364	222
320	257	365	223
321	357	366	206
322	340	367	202
323	341	370	234
324	342	371	233
325	343	372	207
326	246	373	230
327	242	374	235
330	354	375	231
331	353	376	227
332	247		

KOI8-R to MS 1251

For the conversion of KOI8-R to MS 1251, all characters not in the following table are mapped unchanged.

Conversions Performed			
KOI8-R	**MS 1251**	**KOI8-R**	**MS 1251**
24	4	317	356
200-237	40	320	357
241	220	321	377
242	203	322	360
243	270	323	361
244	272	324	362
245	276	325	363
246	263	326	346
247	277	327	342
250	274	330	374
251	232	331	373

Conversions Performed			
KOI8-R	**MS 1251**	**KOI8-R**	**MS 1251**
252	234	332	347
253	236	333	370
254	235	334	375
255	210	335	371
256	242	336	367
257	237	337	372
260	271	340	336
261	200	341	300
262	201	342	301
263	250	343	326
264	252	344	304
265	275	345	305
266	262	346	324
267	257	347	303
270	243	350	325
271	212	351	310
272	214	352	311
273	216	353	312
274	215	354	313
275	210	355	314
276	241	356	315
277	217	357	316
300	376	360	317
301	340	361	337
302	341	362	320
303	366	363	321
304	344	364	322
305	345	365	323
306	364	366	306

Conversions Performed			
KOI8-R	**MS 1251**	**KOI8-R**	**MS 1251**
307	343	367	302
310	365	370	334
311	350	371	333
312	351	372	307
313	352	373	330
314	353	374	335
315	354	375	331
316	355	376	327
376	227		

KOI8-R to Mac Cyrillic

For the conversion of KOI8-R to Mac Cyrillic, all characters not in the following table are mapped unchanged.

Conversions Performed			
KOI8-R	**Mac Cyrillic**	**KOI8-R**	**Mac Cyrillic**
24	4	317	356
200-237	40	320	357
240	312	321	337
241	254	322	360
242	257	323	361
243	336	324	362
244	271	325	363
245	317	326	346
246	264	327	342
247	273	330	374
250	300	331	373
251	275	332	347
252	277	333	370
253	40	334	375
254	316	335	371
255	40	336	367

Conversions Performed			
KOI8-R	**Mac Cyrillic**	**KOI8-R**	**Mac Cyrillic**
256	331	337	372
257	333	340	236
260	334	341	200
261	253	342	201
262	256	343	226
263	335	344	204
264	270	345	205
265	301	346	224
266	247	347	203
267	272	350	225
270	267	351	210
271	274	352	211
272	276	353	212
273	40	354	213
274	315	355	214
275	40	356	215
276	330	357	216
277	332	360	217
300	376	361	237
301	340	362	220
302	341	363	221
303	366	364	222
304	344	365	223
305	345	366	206
306	364	367	202
307	343	370	234
310	365	371	233
311	350	372	207
312	351	373	230

Conversions Performed			
KOI8-R	**Mac Cyrillic**	**KOI8-R**	**Mac Cyrillic**
313	352	374	235
314	353	375	231
315	354	376	227
316	355		

FILES

/usr/lib/iconv/*.so	conversion modules
/usr/lib/iconv/*.t	conversion tables
/usr/lib/iconv/iconv_data	list of conversions supported by conversion tables

SEE ALSO

iconv(1), iconv(3C), iconv(5)

NAME

iconv_mac_cyr – code set conversion tables for Macintosh Cyrillic

DESCRIPTION

The following code set conversions are supported:

Code Set Conversions Supported				
Code	**Symbol**	**Target Code**	**Symbol**	**Target Output**
Mac Cyrillic	mac	ISO 8859-5	iso5	ISO 8859-5 Cyrillic
Mac Cyrillic	mac	KOI8-R	koi8	KOI8-R
Mac Cyrillic	mac	PC Cyrillic	alt	Alternative PC Cyrillic
Mac Cyrillic	mac	MS 1251	win5	Windows Cyrillic

CONVERSIONS

The conversions are performed according to the following tables. All values in the tables are given in octal.

Mac Cyrillic to ISO 8859-5

For the conversion of Mac Cyrillic to ISO 8859-5, all characters not in the following table are mapped unchanged.

Conversions Performed			
Mac Cyrillic	**ISO 8859-5**	**Mac Cyrillic**	**ISO 8859-5**
24	4	276	252
200	260	277	372
201	261	300	370
202	262	301	245
203	263	302-311	40
204	264	312	240
205	265	313	242
206	266	314	362
207	267	315	254
210	270	316	374
211	271	317	365
212	272	320-327	40
213	273	330	256
214	274	331	376
215	275	332	257
216	276	333	377

Conversions Performed			
Mac Cyrillic	**ISO 8859-5**	**Mac Cyrillic**	**ISO 8859-5**
217	277	334	360
220	300	335	241
221	301	336	361
222	302	337	357
223	303	340	320
224	304	341	321
225	305	342	322
226	306	343	323
227	307	344	324
230	310	345	325
231	311	346	326
232	312	347	327
233	313	350	330
234	314	351	331
235	315	352	332
236	316	353	333
237	317	354	334
240-246	40	355	335
247	246	356	336
250-252	40	357	337
253	242	360	340
254	362	361	341
255	40	362	342
256	243	363	343
257	363	364	344
260-263	40	365	345
264	366	366	346
265-266	40	367	347
267	250	370	350

Conversions Performed			
Mac Cyrillic	**ISO 8859-5**	**Mac Cyrillic**	**ISO 8859-5**
270	244	371	351
271	364	372	352
272	247	373	353
273	367	374	354
274	251	375	355
275	371	376	356
375	370		

Mac Cyrillic to KOI8-R

For the conversion of Mac Cyrillic to KOI8-R, all characters not in the following table are mapped unchanged.

Conversions Performed			
Mac Cyrillic	**KOI8-R**	**Mac Cyrillic**	**KOI8-R**
24	4	276	272
200	341	277	252
201	342	300	250
202	367	301	265
203	347	302-311	40
204	344	312	240
205	345	313	261
206	366	314	241
207	372	315	274
210	351	316	254
211	352	317	245
212	353	320-327	40
213	354	330	276
214	355	331	256
215	356	332	277
216	357	333	257
217	360	334	260
220	362	335	263

Conversions Performed			
Mac Cyrillic	**KOI8-R**	**Mac Cyrillic**	**KOI8-R**
221	363	336	243
222	364	337	321
223	365	340	301
224	346	341	302
225	350	342	327
226	343	343	307
227	376	344	304
230	373	345	305
231	375	346	326
232	377	347	332
233	371	350	311
234	370	351	312
235	374	352	313
236	340	353	314
237	361	354	315
240-246	40	355	316
247	266	356	317
250-252	40	357	320
253	261	360	322
254	241	361	323
255	40	362	324
256	262	363	325
257	242	364	306
260-263	40	365	310
264	246	366	303
265-266	40	367	336
267	270	370	333
270	264	371	335
271	244	372	337

Conversions Performed			
Mac Cyrillic	**KOI8-R**	**Mac Cyrillic**	**KOI8-R**
272	267	373	331
273	247	374	330
274	271	375	334
275	251	376	300
375	370		

Mac Cyrillic to PC Cyrillic

For the conversion of Mac Cyrillic to PC Cyrillic, all characters not in the following table are mapped unchanged.

Conversions Performed			
Mac Cyrillic	**PC Cyrillic**	**Mac Cyrillic**	**PC Cyrillic**
24	4	355	255
240-334	40	356	256
335	360	357	257
336	361	360	340
337	357	361	341
340	240	362	342
341	241	363	343
342	242	364	344
343	243	365	345
344	244	366	346
345	245	367	347
346	246	370	350
347	247	371	351
350	250	372	352
351	251	373	353
352	252	374	354
353	253	375	355
354	254	376	356
303	366		

Mac Cyrillic to MS 1251

For the conversion of Mac Cyrillic to MS 1251, all characters not in the following table are mapped unchanged.

Conversions Performed			
Mac Cyrillic	**MS 1251**	**Mac Cyrillic**	**MS 1251**
24	4	255	40
200	300	256	201
201	301	257	203
202	302	260-263	40
203	303	264	263
204	304	266	264
205	305	267	243
206	306	270	252
207	307	271	272
210	310	272	257
211	311	273	277
212	312	274	212
213	313	275	232
214	314	276	214
215	315	277	234
216	316	300	274
217	317	301	275
220	320	302	254
221	321	303-306	40
222	322	307	253
223	323	310	273
224	324	311	205
225	325	312	240
226	326	313	200
227	327	314	220
230	330	315	215
231	331	316	235

Conversions Performed			
Mac Cyrillic	**MS 1251**	**Mac Cyrillic**	**MS 1251**
232	332	317	276
233	333	320	226
234	334	321	227
235	335	322	223
236	336	323	224
237	337	324	221
240	206	325	222
241	260	326	40
242	245	327	204
243	40	330	241
244	247	331	242
245	267	332	217
246	266	333	237
247	262	334	271
250	256	335	250
252	231	336	270
253	200	337	377
254	220	362	324

FILES

`/usr/lib/iconv/*.so`	conversion modules
`/usr/lib/iconv/*.t`	conversion tables
`/usr/lib/iconv/iconv_data`	list of conversions supported by conversion tables

SEE ALSO

`iconv`(1), `iconv`(3C), `iconv`(5)

NAME

iconv_maz – code set conversion tables for Mazovia

DESCRIPTION

The following code set conversions are supported:

Code Set Conversions Supported				
Code	**Symbol**	**Target Code**	**Symbol**	**Target Output**
Mazovia	maz	ISO 8859-2	iso2	ISO Latin 2
Mazovia	maz	MS 1250	win2	Windows Latin 2
Mazovia	maz	MS 852	dos2	MS-DOS Latin 2
Mazovia	maz	DHN	dhn	Dom Hanlowy Nauki

CONVERSIONS

The conversions are performed according to the following tables. All values in the tables are given in octal.

Mazovia to ISO 8859-2

For the conversion of Mazovia to ISO 8859-2, all characters not in the following table are mapped unchanged.

Conversions Performed			
Mazovia	**ISO 8859-2**	**Mazovia**	**ISO 8859-2**
24–177	40	230	246
200	307	231	326
201	374	232	334
202	351	233	40
203	342	234	243
204	344	235	40
205	40	236	266
206	261	237	40
207	347	240	254
210	40	241	257
211	353	242	363
212-213	40	243	323
214	356	244	361
215	346	245	321
216	304	246	274
217	241	247	277

Conversions Performed			
Mazovia	**ISO 8859-2**	**Mazovia**	**ISO 8859-2**
220	312	250-340	40
221	352	341	337
222	263	342-365	40
223	364	366	367
224	366	367	40
225	306	370	260
226-227	40	371-376	40
256	201		

Mazovia to MS 1250

For the conversion of Mazovia to MS 1250, all characters not in the following table are mapped unchanged.

Mazovia	MS 1250	Mazovia	MS 1250
200	307	236	234
201	374	237	40
202	351	240	217
203	342	241	257
204	344	242	363
205	40	243	323
206	271	244	361
207	347	245	321
210	40	246	237
211	353	247	277
212-213	40	250-251	40
214	356	252	254
215	346	253-255	40
216	304	256	253
217	245	257	273
220	312	260-340	40
221	352	341	337
222	263	342-345	40

Mazovia	MS 1250	Mazovia	MS 1250
223	364	346	265
224	366	347-360	40
225	306	361	261
226-227	40	362-365	0
230	214	366	367
231	326	367	40
232	334	370	260
233	40	371	40
234	243	372	267
235	40	373-376	40
274	212		

Mazovia to MS 852

For the conversion of Mazovia to MS 852, all characters not in the following table are mapped unchanged.

Conversions Performed			
Mazovia	**MS 852**	**Mazovia**	**MS 852**
205	40	234	235
206	245	235	40
210-213	40	236	230
215	206	237	40
217	244	240	215
220	250	241	275
221	251	243	340
222	210	244	344
225	217	245	343
226-227	40	246	253
230	227	247	276
233	40	250-375	40
227	327		

Mazovia to DHN

For the conversion of Mazovia to DHN, all characters not in the following table are mapped unchanged.

Conversions Performed			
Mazovia	**DHN**	**Mazovia**	**DHN**
200-205	40	234	203
206	211	236	217
207-214	40	240	207
215	212	241	210
216	40	242	216
217	200	243	205
220	202	244	215
221	214	246	220
225	201	247	221
230	206		

FILES

`/usr/lib/iconv/*.so`	conversion modules
`/usr/lib/iconv/*.t`	conversion tables
`/usr/lib/iconv/iconv_data`	list of conversions supported by conversion tables

SEE ALSO

`iconv`(1), `iconv`(3C), `iconv`(5)

NAME

iconv_pc_cyr – code set conversion tables for Alternative PC Cyrillic

DESCRIPTION

The following code set conversions are supported:

Code Set Conversions Supported				
Code	**Symbol**	**Target Code**	**Symbol**	**Target Output**
PC Cyrillic	alt	ISO 8859-5	iso5	ISO 8859-5 Cyrillic
PC Cyrillic	alt	KOI8-R	koi8	KOI8-R
PC Cyrillic	alt	MS 1251	win5	Windows Cyrillic
PC Cyrillic	alt	Mac Cyrillic	mac	Macintosh Cyrillic

CONVERSIONS

The conversions are performed according to the following tables. All values in the tables are given in octal.

PC Cyrillic to ISO 8859-5

For the conversion of PC Cyrillic to ISO 8859-5, all characters not in the following table are mapped unchanged.

Conversions Performed			
PC Cyrillic	**ISO 8859-5**	**PC Cyrillic**	**ISO 8859-5**
24	4	231	311
200	260	232	312
201	261	233	313
202	262	234	314
203	263	235	315
204	264	236	316
205	265	237	317
206	266	240	320
207	267	241	321
210	270	242	322
211	271	243	323
212	272	244	324
213	273	245	325
214	274	246	326
215	275	247	327
216	276	250	330
217	277	251	331

Conversions Performed			
PC Cyrillic	**ISO 8859-5**	**PC Cyrillic**	**ISO 8859-5**
220	300	252	332
221	301	253	333
222	302	254	334
223	303	255	335
224	304	256	336
225	305	257	337
226	306	260-337	255
227	307	360	241
230	310	362-376	255

PC Cyrillic to KOI8-R

For the conversion of PC Cyrillic to KOI8-R, all characters not in the following table are mapped unchanged.

Conversions Performed			
PC Cyrillic	**KOI8-R**	**PC Cyrillic**	**KOI8-R**
24	4	242	327
200	341	243	307
201	342	244	304
202	367	245	305
203	347	246	326
204	344	247	332
205	345	250	311
206	366	251	312
207	372	252	313
210	351	253	314
211	352	254	315
212	353	255	316
213	354	256	317
214	355	257	320
215	356	260-337	255
216	357	340	322

Conversions Performed			
PC Cyrillic	**KOI8-R**	**PC Cyrillic**	**KOI8-R**
217	360	341	323
220	362	342	324
221	363	343	325
222	364	344	306
223	365	345	310
224	346	346	303
225	350	347	336
226	343	350	333
227	376	351	335
230	373	352	337
231	375	353	331
232	377	354	330
233	371	355	334
234	370	356	300
235	374	357	321
236	340	360	263
237	361	361	243
240	301	362-376	255
241	302		

PC Cyrillic to MS 1251

For the conversion of PC Cyrillic to MS 1251, all characters not in the following table are mapped unchanged.

Conversions Performed			
PC Cyrillic	**MS 1251**	**PC Cyrillic**	**MS 1251**
24	4	242	342
200	300	243	343
201	301	244	344
202	302	245	345
203	303	246	346
204	304	247	347

Conversions Performed			
PC Cyrillic	**MS 1251**	**PC Cyrillic**	**MS 1251**
205	305	250	350
206	306	251	351
207	307	252	352
210	310	253	353
211	311	254	354
212	312	255	355
213	313	256	356
214	314	257	357
215	315	260-337	210
216	316	340	360
217	317	341	361
220	320	342	362
221	321	343	363
222	322	344	364
223	323	345	365
224	324	346	366
225	325	347	367
226	326	350	370
227	327	351	371
230	330	352	372
231	331	353	373
232	332	354	374
233	333	355	375
234	334	356	376
235	335	357	377
236	336	360	250
237	337	361	270
240	340	362-376	210
241	341		

PC Cyrillic to Mac Cyrillic

For the conversion of PC Cyrillic to Mac Cyrillic, all characters not in the following table are mapped unchanged.

Conversions Performed			
PC Cyrillic	**Mac Cyrillic**	**PC Cyrillic**	**Mac Cyrillic**
24	4	341	361
240	340	342	362
241	341	343	363
242	342	344	364
243	343	345	365
244	344	346	366
245	345	347	367
246	346	350	370
247	347	351	371
250	350	352	372
251	351	353	373
252	352	354	374
253	353	355	375
254	354	356	376
255	355	357	337
256	356	360	335
257	357	361	336
260-337	40	362-376	40
340	360		

FILES

`/usr/lib/iconv/*.so`	conversion modules
`/usr/lib/iconv/*.t`	conversion tables
`/usr/lib/iconv/iconv_data`	list of conversions supported by conversion tables

SEE ALSO

`iconv`(1), `iconv`(3C), `iconv`(5)

NAME

iconv_unicode – code set conversion tables for Unicode

DESCRIPTION

The following code set conversions are supported:

CODE SET CONVERSIONS SUPPORTED

FROM Code Set Code	FROM Filename Element	TO Code Set Target Code	TO Filename Element
ISO 8859-1 (Latin 1)	8859-1	UTF-8	UTF-8
ISO 8859-2 (Latin 2)	8859-2	UTF-8	UTF-8
ISO 8859-3 (Latin 3)	8859-3	UTF-8	UTF-8
ISO 8859-4 (Latin 4)	8859-4	UTF-8	UTF-8
ISO 8859-5 (Cyrillic)	8859-5	UTF-8	UTF-8
ISO 8859-6 (Arabic)	8859-6	UTF-8	UTF-8
ISO 8859-7 (Greek)	8859-7	UTF-8	UTF-8
ISO 8859-8 (Hebrew)	8859-8	UTF-8	UTF-8
ISO 8859-9 (Latin 5)	8859-9	UTF-8	UTF-8
ISO 8859-10 (Latin 6)	8859-10	UTF-8	UTF-8
Japanese EUC	eucJP	UTF-8	UTF-8
Chinese/PRC EUC (GB 2312-1980)	gb2312	UTF-8	UTF-8
ISO-2022	iso2022	UTF-8	UTF-8
Korean EUC	ko_KR-euc	Korean UTF-8	ko_KR-UTF-8
ISO-2022-KR	ko_KR-iso2022-7	Korean UTF-8	ko_KR_UTF-8
Korean Johap (KS C 5601-1987)	ko_KR-johap	Korean UTF-8	ko_KR-UTF-8
Korean Johap (KS C 5601-1992)	ko_KR-johap92	Korean UTF-8	ko_KR-UTF-8
Korean UTF-8	ko_KR-UTF-8	Korean EUC	ko_KR-euc
Korean UTF-8	ko_KR-UTF-8	Korean Johap (KS C 5601-1987)	ko_KR-johap
Korean UTF-8	ko_KR-UTF-8	Korean Johap (KS C 5601-1992)	ko_KR-johap92
KOI8-R (Cyrillic)	KOI8-R	UCS-2	UCS-2
KOI8-R (Cyrillic)	KOI8-R	UTF-8	UTF-8
PC Kanji (SJIS)	PCK	UTF-8	UTF-8
PC Kanji (SJIS)	SJIS	UTF-8	UTF-8
UCS-2	UCS-2	KOI8-R (Cyrillic)	KOI8-R
UCS-2	UCS-2	UCS-4	UCS-4

CODE SET CONVERSIONS SUPPORTED

FROM Code Set Code	FROM Filename Element	TO Code Set Target Code	TO Filename Element
UCS-2	UCS-2	UTF-7	UTF-7
UCS-2	UCS-2	UTF-8	UTF-8
UCS-4	UCS-4	UCS-2	UCS-2

UCS-4	UCS-4	UTF-16	UTF-16
UCS-4	UCS-4	UTF-7	UTF-7
UCS-4	UCS-4	UTF-8	UTF-8
UTF-16	UTF-16	UCS-4	UCS-4
UTF-16	UTF-16	UTF-8	UTF-8
UTF-7	UTF-7	UCS-2	UCS-2
UTF-7	UTF-7	UCS-4	UCS-4
UTF-7	UTF-7	UTF-8	UTF-8
UTF-8	UTF-8	ISO 8859-1 (Latin 1)	8859-1
UTF-8	UTF-8	ISO 8859-2 (Latin 2)	8859-2
UTF-8	UTF-8	ISO 8859-3 (Latin 3)	8859-3
UTF-8	UTF-8	ISO 8859-4 (Latin 4)	8859-4
UTF-8	UTF-8	ISO 8859-5 (Cyrillic)	8859-5
UTF-8	UTF-8	ISO 8859-6 (Arabic)	8859-6
UTF-8	UTF-8	ISO 8859-7 (Greek)	8859-7
UTF-8	UTF-8	ISO 8859-8 (Hebrew)	8859-8
UTF-8	UTF-8	ISO 8859-9 (Latin 5)	8859-9
UTF-8	UTF-8	ISO 8859-10 (Latin 6)	8859-10
UTF-8	UTF-8	Japanese EUC	eucJP
UTF-8	UTF-8	Chinese/PRC EUC (GB 2312-1980)	gb2312
UTF-8	UTF-8	ISO-2022	iso2022
UTF-8	UTF-8	KOI8-R (Cyrillic)	KOI8-R
UTF-8	UTF-8	PC Kanji (SJIS)	PCK
UTF-8	UTF-8	PC Kanji (SJIS)	SJIS
UTF-8	UTF-8	UCS-2	UCS-2
UTF-8	UTF-8	UCS-4	UCS-4
UTF-8	UTF-8	UTF-16	UTF-16
UTF-8	UTF-8	UTF-7	UTF-7
UTF-8	UTF-8	Chinese/PRC EUC (GB 2312-1980)	zh_CN.euc

CODE SET CONVERSIONS SUPPORTED

FROM Code Set Code	FROM Filename Element	TO Code Set Target Code	TO Filename Element
UTF-8	UTF-8	ISO 2022-CN	zh_CN.iso2022-7
UTF-8	UTF-8	Chinese/Taiwan Big5	zh_TW-big5
UTF-8	UTF-8	Chinese/Taiwan EUC (CNS 11643-1992)	zh_TW-euc
UTF-8	UTF-8	ISO 2022-TW	zh_TW-iso2022-7
Chinese/PRC EUC (GB 2312-1980)	zh_CN.euc	UTF-8	UTF-8
ISO 2022-CN	zh_CN.iso2022-7	UTF-8	UTF-8
Chinese/Taiwan Big5	zh_TW-big5	UTF-8	UTF-8
Chinese/Taiwan EUC (CNS 11643-1992)	zh_TW-euc	UTF-8	UTF-8
ISO 2022-TW	zh_TW-iso2022-7	UTF-8	UTF-8

EXAMPLES

EXAMPLE 1 The library module filename

In the conversion library, `/usr/lib/iconv` (see `iconv`(3C)), the library module filename is composed of two symbolic elements separated by the percent sign (%). The first symbol specifies the code set that is being converted; the second symbol specifies the *target code*, that is, the code set to which the first one is being converted.

In the conversion table above, the first symbol is termed the "FROM Filename Element". The second symbol, representing the target code set, is the "TO Filename Element".

For example, the library module filename to convert from the *Korean EUC* code set to the *Korean UTF-8* code set is

`ko_KR-euc%ko_KR-UTF-8`

FILES

`/usr/lib/iconv/*.so` conversion modules

SEE ALSO

`iconv`(1), `iconv`(3C), `iconv`(5)

Chernov, A., *Registration of a Cyrillic Character Set*, RFC 1489, RELCOM Development Team, July 1993.

Chon, K., H. Je Park, and U. Choi, *Korean Character Encoding for Internet Messages*, RFC 1557, Solvit Chosun Media, December 1993.

Goldsmith, D., and M. Davis, *UTF-7 – A Mail-Safe Transformation Format of Unicode*, RFC 1642, Taligent, Inc., July 1994.

Lee, F., *HZ – A Data Format for Exchanging Files of Arbitrarily Mixed Chinese and ASCII characters*, RFC 1843, Stanford University, August 1995.

Murai, J., M. Crispin, and E. van der Poel, *Japanese Character Encoding for Internet Messages*, RFC 1468, Keio University, Panda Programming, June 1993.

Nussbacher, H., and Y. Bourvine, *Hebrew Character Encoding for Internet Messages*, RFC 1555, Israeli Inter-University, Hebrew University, December 1993.

Ohta, M., *Character Sets ISO-10646 and ISO-10646-J-1*, RFC 1815, Tokyo Institute of Technology, July 1995.

Ohta, M., and K. Handa, *ISO-2022-JP-2: Multilingual Extension of ISO-2022-JP*, RFC 1554, Tokyo Institute of Technology, December 1993.

Reynolds, J., and J. Postel, *ASSIGNED NUMBERS*, RFC 1700, University of Southern California / Information Sciences Institute, October 1994.

Simonson, K., *Character Mnemonics & Character Sets*, RFC 1345, Rationel Almen Planlaegning, June 1992.

Spinelli, D., *Greek Character Encoding for Electronic Mail Messages*, RFC 1947, SENA S.A., May 1996.

The Unicode Consortium, *The Unicode Standard*, Version 2.0, Addison Wesley Developers Press, July 1996.

Wei, Y., Y. Zhang, J. Li, J. Ding, and Y. Jiang, *ASCII Printable Characters-Based Chinese Character Encoding for Internet Messages*, RFC 1842, AsiaInfo Services Inc., Harvard University, Rice University, University of Maryland, August 1995.

Yergeau, F., *UTF-8, a transformation format of Unicode and ISO 10646*, RFC 2044, Alis Technologies, October 1996.

Zhu, H., D. Hu, Z. Wang, T. Kao, W. Chang, and M. Crispin, *Chinese Character Encoding for Internet Messages*, RFC 1922, Tsinghua University, China Information Technology Standardization Technical Committee (CITS), Institute for Information Industry (III), University of Washington, March 1996.

NOTES

ISO 8859 character sets using Latin alphabetic characters are distinguished as follows:

`ISO 8859-1 (Latin 1)`

For most West European languages, including:

Albanian	Finnish	Italian
Catalan	French	Norwegian
Danish	German	Portuguese
Dutch	Galician	Spanish
English	Irish	Swedish
Faeroese	Icelandic	

`ISO 8859-2 (Latin 2)`

For most Latin-written Slavic and Central European languages:

Czech	Polish	Slovak
German	Rumanian	Slovene
Hungarian	Croatian	

`ISO 8859-3 (Latin 3)`

Popularly used for Esperanto, Galician, Maltese, and Turkish.

`ISO 8859-4 (Latin 4)`

Introduces letters for Estonian, Latvian, and Lithuanian. It is an incomplete predecessor of ISO 8859-10 (Latin 6).

`ISO 8859-9 (Latin 5)`
Replaces the rarely needed Icelandic letters in ISO 8859-1 (Latin 1) with the Turkish ones.

`ISO 8859-10 (Latin 6)`
Adds the last Inuit (Greenlandic) and Sami (Lappish) letters that were not included in ISO 8859-4 (Latin 4) to complete coverage of the Nordic area.

NAME

isalist – the native instruction sets known to Solaris software

DESCRIPTION

The possible instruction set names returned by `isalist`(1) and the `SI_ISALIST` command of `sysinfo`(2) are listed here.

The list is ordered within an instruction set family in the sense that later names are generally faster then earlier names; note that this is in the reverse order than listed by `isalist`(1) and `sysinfo`(2). In the following list of values, numbered entries generally represent increasing performance; lettered entries are either mutually exclusive or cannot be ordered.

SPARC Platforms

Where appropriate, correspondence with a given value of the –xarch option of Sun's C 4.0 compiler is indicated. Other compilers may have similar options.

1a. sparc — Indicates the SPARC V8 instruction set, as defined in The SPARC Architecture Manual, Version 8, Prentice-Hall, Inc., 1992. Some instructions (such as integer multiply and divide, FSMULD, and all floating point operations on quad operands) may be emulated by the kernel on certain systems.

1b. sparcv7 — Same as sparc. This corresponds to code produced with the –xarch=v7 option of Sun's C 4.0 compiler.

2. sparcv8–fsmuld — Like sparc, except that integer multiply and divide must be executed in hardware. This corresponds to code produced with the –xarch=v8a option of Sun's C 4.0 compiler.

3. sparcv8 — Like sparcv8–fsmuld, except that FSMULD must also be executed in hardware. This corresponds to code produced with the –xarch=v8 option of Sun's C 4.0 compiler.

4. sparcv8plus — Indicates the SPARC V8 instruction set plus those instructions in the SPARC V9 instruction set, as defined in The SPARC Architecture Manual, Version 9, Prentice-Hall, 1994, that can be used according to The V8+ Technical Specification. This corresponds to code produced with the –xarch=v8plus option of Sun's C 4.0 compiler.

5a. sparcv8plus+vis — Like sparcv8plus, with the addition of those UltraSPARC I Visualization Instructions that can be used according to The V8+ Technical Specification. This corresponds to code produced with the –xarch=v8plusa option of Sun's C 4.0 compiler.

	5b. sparcv8plus+fmuladd	Like sparcv8plus, with the addition of the Hal SPARC64 floating multiply-add and multiply-subtract instructions.
	6. sparcv9	Indicates the SPARC V9 instruction set, as defined in The SPARC Architecture Manual, Version 9, Prentice-Hall, 1994.
	7a. sparcv9+vis	Like sparcv9, with the addition of the UltraSPARC I Visualization Instructions.
	7b. sparcv9+fmuladd	Like sparcv9, with the addition of the Hal SPARC64 floating multiply-add and multiply-subtract instructions.
Intel Platforms	1. i386	The Intel 80386 instruction set, as described in The i386 Microprocessor Programmer's Reference Manual.
	2. i486	The Intel 80486 instruction set, as described in The i486 Microprocessor Programmer's Reference Manual. (This is effectively i386, plus the CMPXCHG, BSWAP, and XADD instructions.)
	3. pentium	The Intel Pentium instruction set, as described in The Pentium Processor User's Manual. (This is effectively i486, plus the CPU_ID instruction, and any features that the CPU_ID instruction indicates are present.)
	4. pentium+mmx	Like pentium, with the MMX instructions guaranteed present.
	5. pentium_pro	The Intel PentiumPro instruction set, as described in The PentiumPro Family Developer's Manual. (This is effectively pentium, with the CMOVcc, FCMOVcc, FCOMI, and RDPMC instructions guaranteed present.)
	6. pentium_pro+mmx	Like pentium_pro, with the MMX instructions guaranteed present.

SEE ALSO

`isalist`(1), `sysinfo`(2)

NAME

largefile – large file status of utilities

DESCRIPTION

A *large file* is a regular file whose size is greater than or equal to 2 Gbyte (2^{31} bytes). A *small file* is a regular file whose size is less than 2 Gbyte.

Large file aware utilities

A utility is called *large file aware* if it can process large files in the same manner as it does small files. A utility that is large file aware is able to handle large files as input and generate as output large files that are being processed. The exception is where additional files are used as system configuration files or support files that can augment the processing. For example, the `file` utility supports the `–m` option for an alternative "magic" file and the `–f` option for a support file that can contain a list of file names. It is unspecified whether a utility that is large file aware will accept configuration or support files that are large files. If a large file aware utility does not accept configuration or support files that are large files, it will cause no data loss or corruption upon encountering such files and will return an appropriate error.

The following `/usr/bin` utilities are large file aware:

adb	awk	bdiff	cat	chgrp
chmod	chown	cksum	cmp	compress
cp	csh	csplit	cut	dd
dircmp	du	egrep	fgrep	file
find	ftp	getconf	grep	head
join	jsh	ksh	ln	ls
mdb	mkdir	mkfifo	more	mv
nawk	page	paste	pathchk	pg
rcp	remsh	rksh	rm	rmdir
rsh	sed	sh	sort	split
sum	tail	tar	tee	test
touch	tr	uncompress	uudecode	uuencode
wc	zcat			

The following `/usr/xpg4/bin` utilities are large file aware:

awk	cp	du	egrep	fgrep
grep	ln	ls	more	mv
rm	sed	sh	sort	tail
tr				

The following `/usr/sbin` utilities are large file aware:

`install`	`mkfile`	`mknod`	`mvdir`	`swap`

See the USAGE section of the `swap`(1M) manual page for limitations of `swap` on block devices greater than 2 Gbyte on a 32–bit operating system.

The following `/usr/ucb` utilities are large file aware:

`chown`	`from`	`ln`	`ls`	`sed`
`sum`	`touch`			

The `/usr/bin/cpio` and `/usr/bin/pax` utilities are large file aware, but cannot archive a file whose size exceeds 8 Gbyte – 1 byte.

The `/usr/sbin/crash` and `/usr/bin/truss` utilities have been modified to read a dump file and display information relevant to large files, such as offsets.

cachefs file systems

The following `/usr/bin` utilities are large file aware for `cachefs` file systems:

`cachefspack`	`cachefsstat`

The following `/usr/sbin` utilities are large file aware for `cachefs` file systems:

`cachefslog`	`cachefswssize`	`cfsadmin`	`fsck`
`mount`	`umount`		

nfs file systems

The following utilities are large file aware for `nfs` file systems:

/usr/lib/autofs/automountd	/usr/sbin/mount

ufs file systems

The following `/usr/bin` utility is large file aware for `ufs` file systems:

`df`

The following `/usr/xpg4/bin` utility is large file aware for `ufs` file systems:

`df`

The following `/usr/sbin` utilities are large file aware for `ufs` file systems:

`clri`	`dcopy`	`edquota`	`ff`	`fsck`
`fsdb`	`fsirand`	`fstyp`	`labelit`	`lockfs`
`mkfs`	`mount`	`ncheck`	`newfs`	`quot`

`quota`	`quotacheck`	`quotaoff`	`quotaon`	`repquota`
`tunefs`	`ufsdump`	`ufsrestore`	`umount`	

Large file safe utilities

A utility is called *large file safe* if it causes no data loss or corruption when it encounters a large file. A utility that is large file safe is unable to process properly a large file, but returns an appropriate error.

The following `/usr/bin` utilities are large file safe:

`audioconvert`	`audioplay`	`audiorecord`	`comm`	`diff`
`diff3`	`diffmk`	`ed`	`lp`	`mail`
`mailcompat`	`mailstats`	`mailx`	`pack`	`pcat`
`red`	`rmail`	`sdiff`	`unpack`	`vi`
`view`				

The following `/usr/xpg4/bin` utilities are large file safe:

`ed`	`vi`	`view`

The following `/usr/sbin` utilities are large file safe:

lpfilter	lpforms

The following `/usr/ucb` utilities are large file safe:

`Mail`	`lpr`

The following `/usr/lib` utility is large file safe:

`sendmail`

SEE ALSO

`lf64`(5), `lfcompile`(5), `lfcompile64`(5)

NAME

lf64 – transitional interfaces for 64-bit file offsets

DESCRIPTION

The data types, interfaces, and macros described on this page provide explicit access to 64-bit file offsets. They are accessible through the transitional compilation environment described on the `lfcompile64`(5) manual page. The function prototype and semantics of a transitional interface are equivalent to those of the standard version of the call, except that relevant data types are 64-bit entities.

Data Types

The following tables list the standard data or struct types in the left-hand column and their corresponding explicit 64-bit file offset types in the right-hand column, grouped by header. The absence of an entry in the left-hand column indicates that there is no existing explicit 32-bit type that corresponds to the 64–bit type listed in the right—hand column. Note that in a 64-bit application, the standard definition is equivalent to the 64-bit file offset definition.

<aio.h>

struct `aiocb`	struct `aiocb64`
`off_t` aio_offset;	`off64_t` aio_offset;

<sys/dirent.h>

struct `dirent`	struct `dirent64`
`ino_t` d_ino;	`ino64_t` d_ino;
`off_t` d_off;	`off64_t` d_off;

<sys/fcntl.h>

struct `flock`	struct `flock64`
`off_t` l_start;	`off64_t` l_start;
`off_t` l_len;	`off64_t` l_len;
`F_SETLK`	`F_SETLK64`
`F_SETLKW`	`F_SETLKW64`
`F_GETLK`	`F_GETLK64`
`F_FREESP`	`F_FREESP64`
	`O_LARGEFILE`

<sys/stdio.h>

`fpos_t`	`fpos64_t`

<sys/resource.h>

`rlim_t`	`rlim64_t`
struct `rlimit`	struct `rlimit64`
`rlim_t` rlim_cur;	`rlim64_t` rlim_cur;
`rlim_t` rlim_max;	`rlim64_t` rlim_max;
`RLIM_INFINITY`	`RLIM64_INFINITY`
`RLIM_SAVED_MAX`	`RLIM64_SAVED_MAX`
`RLIM_SAVED_CUR`	`RLIM64_SAVED_CUR`

<sys/stat.h>

struct `stat`	struct `stat64`
`ino_t` st_ino;	`ino64_t` st_ino;
`off_t` st_size;	`off64_t` st_size;
`blkcnt_t` st_blocks;	`blkcnt64_t` st_blocks;

<sys/statvfs.h>

struct `statvfs`	struct `statvfs64`
`fsblkcnt_t` f_blocks;	`fsblkcnt64_t` f_blocks;
`fsblkcnt_t` f_bfree;	`fsblkcnt64_t` f_bfree;
`fsblkcnt_t` f_bavial;	`fsblkcnt64_t` f_bavial;
`fsfilcnt_t` f_files;	`fsfilcnt64_t` f_files;
`fsfilcnt_t` f_ffree;	`fsfilcnt64_t` f_ffree;
`fsfilcnt_t` f_favail;	`fsfilcnt64_t` f_favail;

<sys/types.h>

`off_t`;	`off64_t`;
`ino_t`;	`ino64_t`;
`blkcnt_t`;	`blkcnt64_t`;
`fsblkcnt_t`;	`fsblkcnt64_t`;
`fsfilcnt_t`;	`fsfilcnt64_t`;

<unistd.h>

_LFS64_LARGEFILE

_LFS64_STDIO

<sys/unistd.h>

_CS_LFS64_CFLAGS

_CS_LFS64_LDFLAGS

_CS_LFS64_LIBS

_CS_LFS64_LINTFLAGS

System Interfaces

The following tables display the standard API and the corresponding transitional interfaces for 64-bit file offsets. The interfaces are grouped by header. The interface name and the affected data types are displayed in courier font..

<aio.h>

int `aio_cancel`(. . .,	int `aio_cancel64`(. . .,
struct `aiocb` *);	struct `aiocb64` *);
int `aio_error`(	int `aio_error64`(
const struct `aiocb` *);	const struct `aiocb64` *);
int `aio_fsync`(. . .,	int `aio_fsync64`(. . .,
struct `aiocb` *);	struct `aiocb64` *);
int `aio_read`(struct `aiocb` *);	int `aio_read64`(struct `aiocb64` *);
int `aio_return`(struct `aiocb` *);	int `aio_return64`(struct `aiocb64` *);
int `aio_suspend`(	int `aio_suspend64`(
const struct `aiocb` *, . . .);	const struct `aiocb64` *, . . .);
int `aio_write`(struct `aiocb` *);	int `aio_write64`(struct `aiocb64` *);
int `lio_listio`(. . .,	int `lio_listio64`(. . .,
const struct `aiocb` *, . . .);	const struct `aiocb64` *, . . .);

<dirent.h>

struct `dirent *readdir`( );	struct `dirent64 *readdir64`( );
struct `dirent *readdir_r`( );	struct `dirent64 *readdir64_r`( );

<fcntl.h>

int creat();	int creat64();
int open();	int open64();

<ftw.h>

int ftw(. . .,	int ftw64(. . .,
const struct stat *,	const struct stat64 *,
. . .);	. . .);
int nftw(. .	int nftw64(. . .,
const struct stat *,	const struct stat64 *,
. . .);	. . .);

<libgen.h>

char *copylist(. . ., off_t);	char *copylist64(. . ., off64_t);

<stdio.h>

int fgetpos();	int fgetpos64();
FILE *fopen();	FILE *fopen64();
FILE *freopen();	FILE *freopen64();
int fseeko(. . .,	int fseeko64(. . .,
off_t, . . .);	off64_t, . . .);
int fsetpos(. . .,	int fsetpos64(. . .,
const fpos_t *);	const fpos64_t *);
off_t ftello();	off64_t ftello64();
FILE *tmpfile();	FILE *tmpfile64();

<stdlib.h>

int mkstemp();	int mkstemp64();

<sys/async.h>

int aioread(. . ., off_t,	int aioread64(. . ., off64_t,
. . .);	. . .);

int aiowrite(. . ., off_t,	int aiowrite64(. . .,
. . .);	off64_t, . . .);

<ucbinclude/sys/dir.h>

int alphasort(	int alphasort64(
struct direct **,	struct direct64 **,
struct direct **);	struct direct64 **);
struct direct *readdir()	struct direct64 *readdir64();
int scandir(. . .,	int scandir64(. . .,
struct direct *(*[ ]);,	struct direct64 *(*[ ]);,
. . .);	. . .);

<sys/dirent.h>

int getdents(. . ., dirent);	int getdents64(. . .,
	dirent64);

<sys/mman.h>

void mmap(. . ., off_t);	void mmap64(. . ., off64_t);

<sys/resource.h>

int getrlimit(. . .,	int getrlimit64(. . .,
struct rlimit *);	struct rlimit64 *);
int setrlimit(. . .,	int setrlimit64(. . .,
const struct rlimit *);	const struct rlimit64 *);

<sys/stat.h>

int fstat(. . .,	int fstat64(. . .,
struct stat *);	struct stat64 *);
int lstat(. . .,	int lstat64(. . .,
struct stat *);	struct stat64 *);

int stat(. . .,	int stat64(. . .,
struct stat *);	struct stat64 *);

<sys/statvfs.h>

int statvfs(. . .,	int statvfs64(. . .,
struct statvfs *);	struct statvfs64 *);
int fstatvfs(. . .,	int fstatvfs64(. . .,
struct statvfs *);	struct statvfs64 *);

<unistd.h>

int lockf(. . ., off_t);	int lockf64(. . .,
	off64_t);
off_t lseek(. . ., off_t,	off64_t lseek64(. . .,
. . .);	off64_t, . . .);
int ftruncate(. . ., off_t);	int ftruncate64(. . .,
	off64_t);
ssize_t pread(. . ., off_t);	ssize_t pread64(. . .,
	off64_t);
ssize_t pwrite(. . ., off_t);	ssize_t pwrite64(. . .,
	off64_t);
int truncate(. . ., off_t);	int truncate64(. . .,
	off64_t);

SEE ALSO

lfcompile(5), lfcompile64(5)

NAME

lfcompile – large file compilation environment for 32-bit applications

DESCRIPTION

All 64-bit applications can manipulate large files by default. The methods described on this page allow 32-bit applications to manipulate large files.

In the large file compilation environment, source interfaces are bound to appropriate 64-bit functions, structures, and types. Compiling in this environment allows 32-bit applications to access files whose size is greater than or equal to 2 Gbyte (2^{31} bytes).

Each interface named *xxx*`( )` that needs to access 64-bit entities to access large files maps to a *xxx*`64( )` call in the resulting binary. All relevant data types are defined to be of correct size (for example, `off_t` has a typedef definition for a 64-bit entity).

An application compiled in this environment is able to use the *xxx*`( )` source interfaces to access both large and small files, rather than having to explicitly utilize the transitional *xxx*`64( )` interface calls to access large files. See the `lfcompile64`(5) manual page for information regarding the transitional compilation environment.

Applications can be compiled in the large file compilation environment by using the following methods:

- Use the `getconf`(1) utility with one or more of the arguments listed in the table below. This method is recommended for portable applications.

argument	purpose
LFS_CFLAGS	obtain compilation flags necessary to enable the large file compilation environment
LFS_LDFLAGS	obtain link editor options
LFS_LIBS	obtain link library names
LFS_LINTFLAGS	obtain lint options

- Set the compile-time flag `_FILE_OFFSET_BITS` to 64 before including any headers. Applications may combine objects produced in the large file compilation environment with objects produced in the transitional compilation environment, but must be careful with respect to interoperability between those objects. Applications should not declare global variables of types whose sizes change between compilation environments.

Access to Additional Large File Interfaces

The `fseek( )` and `ftell( )` functions *do not* map to functions named `fseek64( )` and `ftell64( )`; rather, the large file additions `fseeko( )` and `ftello( )`, have functionality identical to `fseek( )` and `ftell( )` and *do* map to the 64-bit functions `fseeko64( )` and `ftello64( )`. Applications wishing to access large files should use `fseeko( )` and `ftello( )` in place of `fseek( )` and `ftell( )`. See the `fseek`(3C) and `ftell`(3C) manual pages for information about `fseeko( )` and `ftello( )`.

Applications wishing to access `fseeko( )` and `ftello( )` as well as the POSIX and X/Open specification-conforming interfaces should define the macro `_LARGEFILE_SOURCE` to be 1 and set whichever feature test macros are appropriate to obtain the desired environment (see `standards`(5)).

EXAMPLES

In the following examples, the large file compilation environment is accessed by invoking the `getconf` utility with one of the arguments listed in the table above. The additional large file interfaces are accessed by specifying `–D_LARGEFILE_SOURCE`.

The examples that use the form of command substitution specifying the command within parentheses preceded by a dollar sign can be executed only in a POSIX-conforming shell such as the Korn Shell (see `ksh`(1)). In a shell that is not POSIX-conforming, such as the Bourne Shell (see `sh`(1)) and the C Shell (see `csh`(1)), the `getconf` calls must be enclosed within grave accent marks, as shown in the second example.

EXAMPLE 1 An example of compiling a program with a “large” `off_t`, and that uses `fseeko( )`, `ftello( )`, and `yacc`(1)

```
$ c89 -D_LARGEFILE_SOURCE                \
      -D_FILE_OFFSET_BITS=64 -o foo      \
      $(getconf LFS_CFLAGS) y.tab.c b.o  \
      $(getconf LFS_LDFLAGS)             \
      -ly $(getconf LFS_LIBS)
```

CODE EXAMPLE 1 An example of compiling a program with a “large” `off_t` that does not use `fseeko( )` and `ftello( )` and has no application specific libraries:

```
% c89 -D_FILE_OFFSET_BITS=64     \
      `getconf LFS_CFLAGS` a.c   \
      `getconf LFS_LDFLAGS`      \
      `getconf LFS_LIBS`         \
```

CODE EXAMPLE 2 An example of compiling a program with a “default” `off_t` and that uses `fseeko( )` and `ftello( )`:

```
$ c89 –D_LARGEFILE_SOURCE  a.c
```

SEE ALSO

`csh`(1), `getconf`(1), `ksh`(1), `lint`(1B), `sh`(1), `fseek`(3C), `ftell`(3C), `lf64`(5), `lfcompile64`(5), `standards`(5)

NOTES

Certain system-specific or non-portable interfaces are not usable in the large file compilation environment. Known cases are:

- Kernel data structures read from `/dev/kmem`.
- Interfaces in the kernel virtual memory library, `–lkvm`.
- Interfaces in the ELF access library, `–lelf`.
- Interfaces to `/proc` defined in `<procfs.h>`.

Programs that use these interfaces should not be compiled in the large file compilation environment. As a partial safeguard against making this mistake, including either of the `<libelf.h>` or `<sys/procfs.h>` header files will induce a compilation error when the large file compilation environment is enabled.

In general, caution should be exercised when using any separately-compiled library whose interfaces include data items of type `off_t` or the other redefined types either directly or indirectly, such as with '`struct stat`'. (The redefined types are `off_t`, `rlim_t`, `ino_t`, `blkcnt_t`, `fsblkcnt_t`, and `fsfilcnt_t`.) For the large file compilation environment to work correctly with such a library, the library interfaces must include the appropriate *xxx*`64()` binary entry points and must have them mapped to the corresponding primary functions when `_FILE_OFFSET_BITS` is set to 64.

Care should be exercised using any of the `printf()` or `scanf()` routines on variables of the types mentioned above. In the large file compilation environment, these variables should be printed or scanned using `long long` formats.

BUGS

The `lint`(1B) utility will generate spurious error messages when `_FILE_OFFSET_BITS` is set to 64. This is because the binary `libc` lint library, `/usr/lib/llib-lc.ln`, is compiled only for the standard interfaces, not with `_FILE_OFFSET_BITS` set to 64. This deficiency hampers static error-checking for programs compiled in the large file compilation environment.

Symbolic formats analogous to those found in `<sys/int_fmtio.h>` do not exist for printing or scanning variables of the types that are redefined in the large file compilation environment.

NAME

lfcompile64 – transitional compilation environment

DESCRIPTION

All 64-bit applications can manipulate large files by default. The transitional interfaces described on this page can be used by 32-bit and 64-bit applications to manipulate large files.

In the transitional compilation environment, explicit 64-bit functions, structures, and types are added to the API. Compiling in this environment allows both 32-bit and 64-bit applications to access files whose size is greater than or equal to 2 Gbyte (2^{31} bytes).

The transitional compilation environment exports all the explicit 64-bit functions (*xxx*`64( )`) and types in addition to all the regular functions (*xxx*`( )`) and types. Both *xxx*`( )` and *xxx*`64( )` functions are available to the program source. A 32-bit application must use the *xxx*`64( )` functions in order to access large files. See the `lf64`(5) manual page for a complete listing of the 64-bit transitional interfaces.

The transitional compilation environment differs from the large file compilation environment, wherein the underlying interfaces are bound to 64-bit functions, structures, and types. An application compiled in the large file compilation environment is able to use the *xxx*`( )` source interfaces to access both large and small files, rather than having to explicitly utilize the transitional *xxx*`64( )` interface calls to access large files. See the `lfcompile`(5) manual page for more information regarding the large file compilation environment.

Applications may combine objects produced in the large file compilation environment with objects produced in the transitional compilation environment, but must be careful with respect to interoperability between those objects. Applications should not declare global variables of types whose sizes change between compilation environments.

For applications that do not wish to conform to the POSIX or X/Open specifications, the 64-bit transitional interfaces are available by default. No compile-time flags need to be set.

Access to Additional Large File Interfaces

Applications that wish to access the transitional interfaces as well as the POSIX or X/Open specification-conforming interfaces should use the following compilation methods and set whichever feature test macros are appropriate to obtain the desired environment (see `standards`(5)).

- Set the compile-time flag `_LARGEFILE64_SOURCE` to 1 before including any headers.
- Use the `getconf`(1) command with one or more of the following arguments:

argument	purpose
LFS64_CFLAGS	obtain compilation flags necessary to enable the transitional compilation environment
LFS64_LDFLAGS	obtain link editor options
LFS64_LIBS	obtain link library names
LFS64_LINTFLAGS	obtain lint options

EXAMPLES

In the following examples, the transitional compilation environment is accessed by invoking the `getconf` utility with one of the arguments listed in the table above. The additional large file interfaces are accessed either by specifying `–D_LARGEFILE64_SOURCE` or by invoking the `getconf` utility with the arguments listed above.

The example that uses the form of command substitution specifying the command within parentheses preceded by a dollar sign can be executed only in a POSIX-conforming shell such as the Korn Shell (see `ksh`(1)). In a shell that is not POSIX-conforming, such as the Bourne Shell (see `sh`(1)) and the C Shell (see `csh`(1)), the command must be enclosed within grave accent marks.

EXAMPLE 1 An example of compiling a program using transitional interfaces such as `lseek64( )` and `fopen64( )`:

```
$ c89 -D_LARGEFILE64_SOURCE        \
      $(getconf LFS64_CFLAGS) a.c  \
      $(getconf LFS64_LDFLAGS)     \
      $(getconf LFS64_LIBS)
```

CODE EXAMPLE 1 An example of running lint on a program using transitional interfaces:

```
% lint -D_LARGEFILE64_SOURCE        \
      `getconf LFS64_LINTFLAGS` … \
      `getconf LFS64_LIBS`
```

SEE ALSO

`getconf`(1), `lseek`(2), `fopen`(3C), `lf64`(5), `standards`(5)

NAME

locale – subset of a user's environment that depends on language and cultural conventions

DESCRIPTION

A `locale` is the definition of the subset of a user's environment that depends on language and cultural conventions. It is made up from one or more categories. Each category is identified by its name and controls specific aspects of the behavior of components of the system. Category names correspond to the following environment variable names:

`LC_CTYPE`	Character classification and case conversion.
`LC_COLLATE`	Collation order.
`LC_TIME`	Date and time formats.
`LC_NUMERIC`	Numeric formatting.
`LC_MONETARY`	Monetary formatting.
`LC_MESSAGES`	Formats of informative and diagnostic messages and interactive responses.

The standard utilities base their behavior on the current locale, as defined in the `ENVIRONMENT` section for each utility. The behavior of some of the C-language functions will also be modified based on the current locale, as defined by the last call to `setlocale`(3C).

Locales other than those supplied by the implementation can be created by the application via the `localedef`(1) utility. The value that is used to specify a locale when using environment variables will be the string specified as the *name* operand to `localedef` when the locale was created. The strings "C" and "POSIX" are reserved as identifiers for the POSIX locale.

Applications can select the desired locale by invoking the `setlocale()` function with the appropriate value. If the function is invoked with an empty string, such as:

```
setlocale(LC_ALL, "");
```

the value of the corresponding environment variable is used. If the environment variable is unset or is set to the empty string, the `setlocale()` function sets the appropriate environment.

Locale Definition

Locales can be described with the file format accepted by the `localedef` utility.

The locale definition file must contain one or more locale category source definitions, and must not contain more than one definition for the same locale category.

A category source definition consists of a category header, a category body and a category trailer. A category header consists of the character string naming of the category, beginning with the characters LC_. The category trailer consists of the string END, followed by one or more blank characters and the string used in the corresponding category header.

The category body consists of one or more lines of text. Each line contains an identifier, optionally followed by one or more operands. Identifiers are either keywords, identifying a particular locale element, or collating elements. Each keyword within a locale must have a unique name (that is, two categories cannot have a commonly-named keyword); no keyword can start with the characters LC_. Identifiers must be separated from the operands by one or more blank characters.

Operands must be characters, collating elements or strings of characters. Strings must be enclosed in double-quotes. Literal double-quotes within strings must be preceded by the <*escape character*>, described below. When a keyword is followed by more than one operand, the operands must be separated by semicolons; blank characters are allowed both before and after a semicolon.

The first category header in the file can be preceded by a line modifying the comment character. It has the following format, starting in column 1:

```
"comment_char %c\n",<comment character>
```

The comment character defaults to the number sign (#). Blank lines and lines containing the <*comment character*> in the first position are ignored.

The first category header in the file can be preceded by a line modifying the escape character to be used in the file. It has the following format, starting in column 1:

```
"escape_char %c\n",<escape character>
```

The escape character defaults to backslash.

A line can be continued by placing an escape character as the last character on the line; this continuation character will be discarded from the input. Although the implementation need not accept any one portion of a continued line with a length exceeding {LINE_MAX} bytes, it places no limits on the accumulated length of the continued line. Comment lines cannot be continued on a subsequent line using an escaped newline character.

Individual characters, characters in strings, and collating elements must be represented using symbolic names, as defined below. In addition, characters can be represented using the characters themselves or as octal, hexadecimal or decimal constants. When non-symbolic notation is used, the resultant locale definitions will in many cases not be portable between systems. The left angle bracket (<) is a reserved symbol, denoting the start of a symbolic name; when used to represent itself it must be preceded by the escape character. The following rules apply to character representation:

1. A character can be represented via a symbolic name, enclosed within angle brackets < and >. The symbolic name, including the angle brackets, must exactly match a symbolic name defined in the charmap file specified via the `localedef –f` option, and will be replaced by a character value determined from the value associated with the symbolic name in the charmap file. The use of a symbolic name not found in the charmap file constitutes an error, unless the category is `LC_CTYPE` or `LC_COLLATE`, in which case it constitutes a warning condition (see `localedef`(1) for a description of action resulting from errors and warnings). The specification of a symbolic name in a `collating-element` or `collating-symbol` section that duplicates a symbolic name in the charmap file (if present) is an error. Use of the escape character or a right angle bracket within a symbolic name is invalid unless the character is preceded by the escape character.

   `Example`:

   <c>;<c–cedilla> "<M><a><y>"

2. A character can be represented by the character itself, in which case the value of the character is implementation-dependent. Within a string, the double-quote character, the escape character and the right angle bracket character must be escaped (preceded by the escape character) to be interpreted as the character itself. Outside strings, the characters

   `, ; < >` *escape_char*

   must be escaped to be interpreted as the character itself.

   `Example`:

   `c` *beta-char* `"May"`

3. A character can be represented as an octal constant. An octal constant is specified as the escape character followed by two or more octal digits. Each constant represents a byte value. Multi-byte values can be represented by concatenated constants specified in byte order with the last constant specifying the least significant byte of the character.

`Example`:

```
\143;\347;\143\150 "\115\141\171"
```

4. A character can be represented as a hexadecimal constant. A hexadecimal constant is specified as the escape character followed by an `x` followed by two or more hexadecimal digits. Each constant represents a byte value. Multi-byte values can be represented by concatenated constants specified in byte order with the last constant specifying the least significant byte of the character.

   `Example`:

   ```
   \x63;\xe7;\x63\x68 "\x4d\x61\x79"
   ```

5. A character can be represented as a decimal constant. A decimal constant is specified as the escape character followed by a `d` followed by two or more decimal digits. Each constant represents a byte value. Multi-byte values can be represented by concatenated constants specified in byte order with the last constant specifying the least significant byte of the character.

   `Example`:

   ```
   \d99;\d231;\d99\d104 "\d77\d97\d121"
   ```

   Only characters existing in the character set for which the locale definition is created can be specified, whether using symbolic names, the characters themselves, or octal, decimal or hexadecimal constants. If a charmap file is present, only characters defined in the charmap can be specified using octal, decimal or hexadecimal constants. Symbolic names not present in the charmap file can be specified and will be ignored, as specified under item 1 above.

LC_CTYPE

The `LC_CTYPE` category defines character classification, case conversion and other character attributes. In addition, a series of characters can be represented by three adjacent periods representing an ellipsis symbol (. . .). The ellipsis specification is interpreted as meaning that all values between the values preceding and following it represent valid characters. The ellipsis specification is valid only within a single encoded character set; that is, within a group of characters of the same size. An ellipsis is interpreted as including in the list all characters with an encoded value higher than the encoded value of the character preceding the ellipsis and lower than the encoded value of the character following the ellipsis.

`Example`:

```
\x30;...;\x39;
```

includes in the character class all characters with encoded values between the endpoints.

The following keywords are recognized. In the descriptions, the term "automatically included" means that it is not an error either to include or omit any of the referenced characters.

The character classes `digit`, `xdigit`, `lower`, `upper`, and `space` have a set of automatically included characters. These only need to be specified if the character values (that is, encoding) differ from the implementation default values.

`cswidth`
: Moved to `extensions` file (see `extensions`(5)).

`upper`
: Define characters to be classified as upper-case letters.

  In the POSIX locale, the 26 upper-case letters are included:

  A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

  In a locale definition file, no character specified for the keywords `cntrl`, `digit`, `punct`, or `space` can be specified. The upper-case letters `A` to `Z` are automatically included in this class.

`lower`
: Define characters to be classified as lower-case letters. In the POSIX locale, the 26 lower-case letters are included:

  a b c d e f g h i j k l m n o p q r s t u v w x y z

  In a locale definition file, no character specified for the keywords `cntrl`, `digit`, `punct`, or `space` can be specified. The lower-case letters `a` to `z` of the portable character set are automatically included in this class.

`alpha`
: Define characters to be classified as letters.

  In the POSIX locale, all characters in the classes `upper` and `lower` are included.

In a locale definition file, no character specified for the keywords `cntrl`, `digit`, `punct`, or `space` can be specified. Characters classified as either `upper` or `lower` are automatically included in this class.

`digit` Define the characters to be classified as numeric digits.

In the POSIX locale, only

0 1 2 3 4 5 6 7 8 9

are included.

In a locale definition file, only the digits `0`, `1`, `2`, `3`, `4`, `5`, `6`, `7`, `8`, and `9` can be specified, and in contiguous ascending sequence by numerical value. The digits `0` to `9` of the portable character set are automatically included in this class.

The definition of character class `digit` requires that only ten characters; the ones defining digits can be specified; alternative digits (for example, Hindi or Kanji) cannot be specified here.

`space` Define characters to be classified as white-space characters.

In the POSIX locale, at a minimum, the characters `SPACE`, `FORMFEED`, `NEWLINE`, `CARRIAGE RETURN`, `TAB`, and `VERTICAL TAB` are included.

In a locale definition file, no character specified for the keywords `upper`, `lower`, `alpha`, `digit`, `graph`, or `xdigit` can be specified. The characters `SPACE`, `FORMFEED`, `NEWLINE`, `CARRIAGE RETURN`, `TAB`, and `VERTICAL TAB` of the portable character set, and any characters included in the class `blank` are automatically included in this class.

`cntrl` Define characters to be classified as control characters.

In the POSIX locale, no characters in classes `alpha` or `print` are included.

In a locale definition file, no character specified for the keywords `upper`, `lower`, `alpha`, `digit`, `punct`, `graph`, `print`, or `xdigit` can be specified.

`punct`
: Define characters to be classified as punctuation characters.

In the POSIX locale, neither the space character nor any characters in classes `alpha`, `digit`, or `cntrl` are included.

In a locale definition file, no character specified for the keywords `upper`, `lower`, `alpha`, `digit`, `cntrl`, `xdigit` or as the space character can be specified.

`graph`
: Define characters to be classified as printable characters, not including the space character.

In the POSIX locale, all characters in classes `alpha`, `digit`, and `punct` are included; no characters in class `cntrl` are included.

In a locale definition file, characters specified for the keywords `upper`, `lower`, `alpha`, `digit`, `xdigit`, and `punct` are automatically included in this class. No character specified for the keyword `cntrl` can be specified.

`print`
: Define characters to be classified as printable characters, including the space character.

In the POSIX locale, all characters in class `graph` are included; no characters in class `cntrl` are included.

In a locale definition file, characters specified for the keywords `upper`, `lower`, `alpha`, `digit`, `xdigit`, `punct`, and the space character are automatically included in this class. No character specified for the keyword `cntrl` can be specified.

`xdigit`
: Define the characters to be classified as hexadecimal digits.

In the POSIX locale, only:

0 1 2 3 4 5 6 7 8 9 A B C D E F a b c d e f

are included.

In a locale definition file, only the characters defined for the class `digit` can be specified, in contiguous ascending sequence by numerical value, followed by one or more sets of six characters representing the hexadecimal digits 10 to 15 inclusive, with each set in ascending order (for example `A`, `B`, `C`, `D`, `E`, `F`, `a`, `b`, `c`, `d`, `e`, `f`). The digits `0` to `9`, the upper-case letters `A` to `F` and the lower-case letters `a` to `f` of the portable character set are automatically included in this class.

The definition of character class `xdigit` requires that the characters included in character class `digit` be included here also.

`blank`
: Define characters to be classified as blank characters.

In the POSIX locale, only the space and tab characters are included.

In a locale definition file, the characters space and tab are automatically included in this class.

`charclass`
: Define one or more locale-specific character class names as strings separated by semi-colons. Each named character class can then be defined subsequently in the `LC_CTYPE` definition. A character class name consists of at least one and at most `{CHARCLASS_NAME_MAX}` bytes of alphanumeric characters from the portable filename character set. The first character of a character class name cannot be a digit. The name cannot match any of the `LC_CTYPE` keywords defined in this document.

`charclass-name`
: Define characters to be classified as belonging to the named locale-specific character class. In the POSIX locale, the locale-specific named character classes need not exist. If a class name is defined by a `charclass` keyword, but no characters are subsequently assigned to it, this is not an error; it represents a class without any characters belonging to it. The `charclass-name` can be used as the *property* argument to the

wctype(3C) function, in regular expression and shell pattern-matching bracket expressions, and by the tr(1) command.

toupper

Define the mapping of lower-case letters to upper-case letters.

In the POSIX locale, at a minimum, the 26 lower-case characters:

a b c d e f g h i j k l m n o p q r s t u v w x y z

are mapped to the corresponding 26 upper-case characters:

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

In a locale definition file, the operand consists of character pairs, separated by semicolons. The characters in each character pair are separated by a comma and the pair enclosed by parentheses. The first character in each pair is the lower-case letter, the second the corresponding upper-case letter. Only characters specified for the keywords lower and upper can be specified. The lower-case letters a to z, and their corresponding upper-case letters A to Z, of the portable character set are automatically included in this mapping, but only when the toupper keyword is omitted from the locale definition.

tolower

Define the mapping of upper-case letters to lower-case letters.

In the POSIX locale, at a minimum, the 26 upper-case characters:

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

are mapped to the corresponding 26 lower-case characters:

a b c d e f g h i j k l m n o p q r s t u v w x y z

In a locale definition file, the operand consists of character pairs, separated by semicolons. The characters in each character pair are separated by a comma and the pair enclosed by parentheses. The first character in each pair is the upper-case letter, the second the corresponding lower-case letter. Only characters specified for the keywords `lower` and `upper` can be specified. If the `tolower` keyword is omitted from the locale definition, the mapping will be the reverse mapping of the one specified for `toupper`.

LC_COLLATE

The `LC_COLLATE` category provides a collation sequence definition for numerous utilities (such as `sort`(1), `uniq`(1), and so forth), regular expression matching (see `regex`(5)), and the `strcoll`(3C), `strxfrm`(3C), `wcscoll`(3C), and `wcsxfrm`(3C) functions.

A collation sequence definition defines the relative order between collating elements (characters and multi-character collating elements) in the locale. This order is expressed in terms of collation values; that is, by assigning each element one or more collation values (also known as collation weights). At least the following capabilities are provided:

1. `Multi-character collating elements`. Specification of multi-character collating elements (that is, sequences of two or more characters to be collated as an entity).
2. `User-defined ordering of collating elements`. Each collating element is assigned a collation value defining its order in the character (or basic) collation sequence. This ordering is used by regular expressions and pattern matching and, unless collation weights are explicity specified, also as the collation weight to be used in sorting.
3. `Multiple weights and equivalence classes`. Collating elements can be assigned one or more (up to the limit `{COLL_WEIGHTS_MAX}`) collating weights for use in sorting. The first weight is hereafter referred to as the primary weight.
4. `One-to-Many mapping`. A single character is mapped into a string of collating elements.
5. `Equivalence class definition`. Two or more collating elements have the same collation value (primary weight).
6. `Ordering by weights`. When two strings are compared to determine their relative order, the two strings are first broken up into a series of collating elements; the elements in each successive pair of elements are then compared according to the relative primary weights for the elements.

If equal, and more than one weight has been assigned, then the pairs of collating elements are recompared according to the relative subsequent weights, until either a pair of collating elements compare unequal or the weights are exhausted. The following keywords are recognized in a collation sequence definition. They are described in detail in the following sections.

7. Define a collating-element symbol representing a multi-character collating element. This keyword is optional.
8. Define a collating symbol for use in collation order statements. This keyword is optional.
9. Define collation rules. This statement is followed by one or more collation order statements, assigning character collation values and collation weights to collating elements.
10. Specify the end of the collation-order statements.

collating-element *keyword*

In addition to the collating elements in the character set, the `collating-element` keyword is used to define multi-character collating elements. The syntax is:

`"collating-element %s from \"%s\"\n",`*<collating-symbol>,<string>*

The *<collating-symbol>* operand is a symbolic name, enclosed between angle brackets (< and >), and must not duplicate any symbolic name in the current charmap file (if any), or any other symbolic name defined in this collation definition. The string operand is a string of two or more characters that collates as an entity. A *<collating-element>* defined via this keyword is only recognized with the `LC_COLLATE` category.

```
Example:
   collating-element <ch> from "<c><h>"
   collating-element <e-acute> from "<acute><e>"
   collating-element <ll> from "ll"
```

collating-symbol *keyword*

This keyword will be used to define symbols for use in collation sequence statements; that is, between the `order_start` and the `order_end` keywords. The syntax is:

`"collating-symbol %s\n",`*<collating-symbol>*

The *<collating-symbol>* is a symbolic name, enclosed between angle brackets (< and >), and must not duplicate any symbolic name in the current charmap file (if any), or any other symbolic name defined in this collation definition.

A `collating-symbol` defined via this keyword is only recognized with the `LC_COLLATE` category.

Example:

```
collating-symbol <UPPER_CASE>
collating-symbol <HIGH>
```

The `collating-symbol` keyword defines a symbolic name that can be associated with a relative position in the character order sequence. While such a symbolic name does not represent any collating element, it can be used as a weight.

order_start *keyword*

The `order_start` keyword must precede collation order entries and also defines the number of weights for this collation sequence definition and other collation rules.

The syntax of the `order_start` keyword is:

`"order_start %s;%s;...;%s\n",` *<sort-rules>*,*<sort-rules>*

The operands to the `order_start` keyword are optional. If present, the operands define rules to be applied when strings are compared. The number of operands define how many weights each element is assigned; if no operands are present, one `forward` operand is assumed. If present, the first operand defines rules to be applied when comparing strings using the first (primary) weight; the second when comparing strings using the second weight, and so on. Operands are separated by semicolons (`;`). Each operand consists of one or more collation directives, separated by commas (`,`). If the number of operands exceeds the `{COLL_WEIGHTS_MAX}` limit, the utility will issue a warning message. The following directives will be supported:

`forward` Specifies that comparison operations for the weight level proceed from start of string towards the end of string.

`backward` Specifies that comparison operations for the weight level proceed from end of string towards the beginning of string.

`position` Specifies that comparison operations for the weight level will consider the relative position of elements in the strings not subject to `IGNORE.` The string containing an element not subject to `IGNORE` after the fewest collating elements subject to `IGNORE` from the start of the compare will collate first. If both strings contain a character not subject to `IGNORE` in the same relative position, the collating values assigned to the elements will determine the ordering. In case of equality, subsequent characters not subject to `IGNORE` are considered in the same manner.

The directives `forward` and `backward` are mutually exclusive.

```
Example:

order_start forward;backward
```

If no operands are specified, a single `forward` operand is assumed. The character (and collating element) order is defined by the order in which characters and elements are specified between the `order_start` and `order_end` keywords. This character order is used in range expressions in regular expressions (see `regex`(5)). Weights assigned to the characters and elements define the collation sequence; in the absence of weights, the character order is also the collation sequence. The `position` keyword provides the capability to consider, in a compare, the relative position of characters not subject to `IGNORE`. As an example, consider the two strings "o-ring" and "or-ing". Assuming the hyphen is subject to `IGNORE` on the first pass, the two strings will compare equal, and the position of the hyphen is immaterial. On second pass, all characters except the hyphen are subject to `IGNORE`, and in the normal case the two strings would again compare equal. By taking position into account, the first collates before the second.

Collation Order

The `order_start` keyword is followed by collating identifier entries. The syntax for the collating element entries is

`"%s %s;%s;. . .;%s\n"`<*collating-identifier*>`,`<*weight*>`,`<*weight*>`, . . .`

Each *collating-identifier* consists of either a character described in `Locale Definition` above, a <*collating-element*>, a <*collating-symbol*>, an ellipsis, or the special symbol `UNDEFINED`. The order in which collating elements are specified determines the character order sequence, such that each collating element compares less than the elements following it. The NUL character compares lower than any other character.

A <*collating-element*> is used to specify multi-character collating elements, and indicates that the character sequence specified via the <*collating-element*> is to be collated as a unit and in the relative order specified by its place.

A <*collating-symbol*> is used to define a position in the relative order for use in weights. No weights are specified with a <*collating-symbol*>.

The ellipsis symbol specifies that a sequence of characters will collate according to their encoded character values. It is interpreted as indicating that all characters with a coded character set value higher than the value of the character in the preceding line, and lower than the coded character set value for the character in the following line, in the current coded character set, will be placed in the character collation order between the previous and the following character in ascending order according to their coded character set values. An initial ellipsis is interpreted as if the preceding line specified the NUL character, and a trailing ellipsis as if the following line specified the highest coded character set value in the current coded character set. An ellipsis is treated as invalid if the preceding or following lines do not specify characters in the current coded character set.

The symbol `UNDEFINED` is interpreted as including all coded character set values not specified explicitly or via the ellipsis symbol. Such characters are inserted in the character collation order at the point indicated by the symbol, and in ascending order according to their coded character set values. If no `UNDEFINED` symbol is specified, and the current coded character set contains characters not specified in this section, the utility will issue a warning message and place such characters at the end of the character collation order.

The optional operands for each collation-element are used to define the primary, secondary, or subsequent weights for the collating element. The first operand specifies the relative primary weight, the second the relative secondary weight, and so on. Two or more collation-elements can be assigned the same weight; they belong to the same *equivalence class* if they have the same primary weight. Collation behaves as if, for each weight level, elements subject to `IGNORE` are removed, unless the `position` collation directive is specified for the corresponding level with the `order_start` keyword. Then each successive pair of elements is compared according to the relative weights for the elements. If the two strings compare equal, the process is repeated for the next weight level, up to the limit {`COLL_WEIGHTS_MAX`}.

Weights are expressed as characters described in `Locale Definition` above, <*collating-symbol*>s, <*collating-element*>s, an ellipsis, or the special symbol `IGNORE.` A single character, a <*collating–symbol*> or a <*collating–element*> represent the relative position in the character collating sequence of the character or symbol, rather than the character or characters themselves. Thus, rather than assigning absolute values to weights, a particular weight is expressed using the relative order value assigned to a collating element based on its order in the character collation sequence.

One-to-many mapping is indicated by specifying two or more concatenated characters or symbolic names. For example, if the character <`eszet`> is given the string "<`s`><`s`>" as a weight, comparisons are performed as if all occurrences of the character <`eszet`> are replaced by <`s`><`s`> (assuming that <`s`> has the collating weight <`s`>). If it is necessary to define <`eszet`> and <`s`><`s`> as an equivalence class, then a collating element must be defined for the string `ss`.

All characters specified via an ellipsis will by default be assigned unique weights, equal to the relative order of characters. Characters specified via an explicit or implicit `UNDEFINED` special symbol will by default be assigned the same primary weight (that is, belong to the same equivalence class). An ellipsis symbol as a weight is interpreted to mean that each character in the sequence has unique weights, equal to the relative order of their character in the character collation sequence. The use of the ellipsis as a weight is treated as an error if the collating element is neither an ellipsis nor the special symbol `UNDEFINED`.

The special keyword `IGNORE` as a weight indicates that when strings are compared using the weights at the level where `IGNORE` is specified, the collating element is ignored; that is, as if the string did not contain the collating element. In regular expressions and pattern matching, all characters that are subject to `IGNORE` in their primary weight form an equivalence class.

An empty operand is interpreted as the collating element itself.

For example, the order statement:

```
<a> <a>;<a>
```

is equal to:

```
<a>
```

An ellipsis can be used as an operand if the collating element was an ellipsis, and is interpreted as the value of each character defined by the ellipsis.

The collation order as defined in this section defines the interpretation of bracket expressions in regular expressions.

`Example`:

```
order_start                       forward;backward
UNDEFINED                         IGNORE;IGNORE
<LOW>
<space>                           <LOW>;<space>
. . .                             <LOW>;. . .
<a>                               <a>;<a>
<a-acute>                         <a>;<a-acute>
<a-grave>                         <a>;<a-grave>
<A>                               <a>;<A>
<A-acute>                         <a>;<A-acute>
```

```
<A-grave>          <a>;<A-grave>
<ch>               <ch>;<ch>
<Ch>               <ch>;<Ch>
<s>                <s>;<s>
<eszet>            "<s><s>";"<eszet><eszet>"
order_end
```

This example is interpreted as follows:

1. The UNDEFINED means that all characters not specified in this definition (explicitly or via the ellipsis) are ignored for collation purposes; for regular expression purposes they are ordered first.
2. All characters between <space> and <a> have the same primary equivalence class and individual secondary weights based on their ordinal encoded values.
3. All characters based on the upper– or lower–case character a belong to the same primary equivalence class.
4. The multi-character collating element <ch> is represented by the collating symbol <ch> and belongs to the same primary equivalence class as the multi-character collating element <Ch>.

order_end *keyword*

The collating order entries must be terminated with an order_end keyword.

LC_MONETARY

The LC_MONETARY category defines the rules and symbols that are used to format monetary numeric information. This information is available through the localeconv(3C) function

The following items are defined in this category of the locale. The item names are the keywords recognized by the localedef(1) utility when defining a locale. They are also similar to the member names of the lconv structure defined in <locale.h>. The localeconv function returns {CHAR_MAX} for unspecified integer items and the empty string ("") for unspecified or size zero string items.

In a locale definition file the operands are strings. For some keywords, the strings can contain only integers. Keywords that are not provided, string values set to the empty string (""), or integer keywords set to −1, are used to indicate that the value is not available in the locale.

int_curr_symbol — The international currency symbol. The operand is a four-character string, with the first three characters containing the alphabetic international currency symbol in accordance with those specified in the ISO 4217:1987 standard. The fourth character is the character used to separate the international currency symbol from the monetary quantity.

`currency_symbol` — The string used as the local currency symbol.

`mon_decimal_point` — The operand is a string containing the symbol that is used as the decimal delimiter (radix character) in monetary formatted quantities. In contexts where standards (such as the ISO C standard) limit the `mon_decimal_point` to a single byte, the result of specifying a multi-byte operand is unspecified.

`mon_thousands_sep` — The operand is a string containing the symbol that is used as a separator for groups of digits to the left of the decimal delimiter in formatted monetary quantities. In contexts where standards limit the `mon_thousands_sep` to a single byte, the result of specifying a multi-byte operand is unspecified.

`mon_grouping` — Define the size of each group of digits in formatted monetary quantities. The operand is a sequence of integers separated by semicolons. Each integer specifies the number of digits in each group, with the initial integer defining the size of the group immediately preceding the decimal delimiter, and the following integers defining the preceding groups. If the last integer is not –1, then the size of the previous group (if any) will be repeatedly used for the remainder of the digits. If the last integer is –1, then no further grouping will be performed.

The following is an example of the interpretation of the `mon_grouping` keyword. Assuming that the value to be formatted is `123456789` and the `mon_thousands_sep` is `'`, then the following table shows the result. The third column shows the equivalent string in the ISO C standard that would be used by the `localeconv` function to accommodate this grouping.

mon_grouping	Formatted Value	ISO C String
3;–1	123456’789	"\3\177"
3	123’456’789	"\3"
3;2;–1	1234’56’789	"\3\2\177"
3;2	12’34’56’789	"\3\2"
–1	123456789	"\177"

In these examples, the octal value of `{CHAR_MAX}` is 177.

`positive_sign` A string used to indicate a non-negative-valued formatted monetary quantity.

`negative_sign` A string used to indicate a negative-valued formatted monetary quantity.

`int_frac_digits` An integer representing the number of fractional digits (those to the right of the decimal delimiter) to be written in a formatted monetary quantity using `int_curr_symbol`.

`frac_digits` An integer representing the number of fractional digits (those to the right of the decimal delimiter) to be written in a formatted monetary quantity using `currency_symbol`.

`p_cs_precedes` An integer set to 1 if the `currency_symbol` or `int_curr_symbol` precedes the value for a monetary quantity with a non-negative value, and set to 0 if the symbol succeeds the value.

`p_sep_by_space` An integer set to 0 if no space separates the `currency_symbol` or `int_curr_symbol` from the value for a monetary quantity with a non-negative value, set to 1 if a space separates the symbol from the value, and set to 2 if a space separates the symbol and the sign string, if adjacent.

`n_cs_precedes` An integer set to 1 if the `currency_symbol` or `int_curr_symbol` precedes the value for a monetary quantity with a negative value, and set to 0 if the symbol succeeds the value.

`n_sep_by_space` — An integer set to 0 if no space separates the `currency_symbol` or `int_curr_symbol` from the value for a monetary quantity with a negative value, set to 1 if a space separates the symbol from the value, and set to 2 if a space separates the symbol and the sign string, if adjacent.

`p_sign_posn` — An integer set to a value indicating the positioning of the `positive_sign` for a monetary quantity with a non-negative value. The following integer values are recognized for both `p_sign_posn` and `n_sign_posn`:

- 0 — Parentheses enclose the quantity and the `currency_symbol` or `int_curr_symbol`.
- 1 — The sign string precedes the quantity and the `currency_symbol` or `int_curr_symbol`.
- 2 — The sign string succeeds the quantity and the `currency_symbol` or `int_curr_symbol`.
- 3 — The sign string precedes the `currency_symbol` or `int_curr_symbol`.
- 4 — The sign string succeeds the `currency_symbol` or `int_curr_symbol`.

`n_sign_posn` — An integer set to a value indicating the positioning of the `negative_sign` for a negative formatted monetary quantity.

The following table shows the result of various combinations:

		p_sep_by_space		
		2	1	0
p_cs_precedes= 1	p_sign_posn= 0	($1.25)	($1.25)	($1.25)
	p_sign_posn= 1	+$1.25	+$1.25	+$1.25
	p_sign_posn= 2	$1.25+	$1.25+	$1.25+
	p_sign_posn= 3	+$1.25	+$1.25	+$1.25

	p_sign_posn= 4	$+1.25	$+1.25	$+1.25
p_cs_precedes= 0	p_sign_posn= 0	(1.25 $)	(1.25 $)	(1.25$)
	p_sign_posn= 1	+1.25 $	+1.25 $	+1.25$
	p_sign_posn= 2	1.25$ +	1.25 $+	1.25$+
	p_sign_posn= 3	1.25+ $	1.25 +$	1.25+$
	p_sign_posn= 4	1.25$ +	1.25 $+	1.25$+

The monetary formatting definitions for the POSIX locale follow; the code listing depicting the `localedef`(1) input, the table representing the same information with the addition of `localeconv`(3C) and `nl_langinfo`(3C) formats. All values are unspecified in the POSIX locale.

```
LC_MONETARY
# This is the POSIX locale definition for
# the LC_MONETARY category.
#


int_curr_symbol             ""
currency_symbol             ""
mon_decimal_point           ""
mon_thousands_sep           ""
mon_grouping                -1
positive_sign               ""
negative_sign               ""
int_frac_digits             -1
p_cs_precedes               -1
p_sep_by_space              -1
n_cs_precedes               -1
n_sep_by_space              -1
p_sign_posn                 -1
n_sign_posn                 -1

# END LC_MONETARY
```

The entry `n/a` indicates that the value is not available in the POSIX locale.

LC_NUMERIC

The `LC_NUMERIC` category defines the rules and symbols that will be used to format non-monetary numeric information. This information is available through the `localeconv`(3C) function.

The following items are defined in this category of the locale. The item names are the keywords recognized by the `localedef` utility when defining a locale. They are also similar to the member names of the *lconv* structure defined in <`locale.h`>. The `localeconv( )` function returns `{CHAR_MAX}` for unspecified integer items and the empty string (`""`) for unspecified or size zero string items.

In a locale definition file the operands are strings. For some keywords, the strings only can contain integers. Keywords that are not provided, string values set to the empty string (`""`), or integer keywords set to –1, will be used to indicate that the value is not available in the locale. The following keywords are recognized:

`decimal_point`
: The operand is a string containing the symbol that is used as the decimal delimiter (radix character) in numeric, non-monetary formatted quantities. This keyword cannot be omitted and cannot be set to the empty string. In contexts where standards limit the `decimal_point` to a single byte, the result of specifying a multi-byte operand is unspecified.

`thousands_sep`
: The operand is a string containing the symbol that is used as a separator for groups of digits to the left of the decimal delimiter in numeric, non-monetary formatted monetary quantities. In contexts where standards limit the `thousands_sep` to a single byte, the result of specifying a multi-byte operand is unspecified.

`grouping`
: Define the size of each group of digits in formatted non-monetary quantities. The operand is a sequence of integers separated by semicolons. Each integer specifies the number of digits in each group, with the initial integer defining the size of the group immediately preceding the decimal delimiter, and the following integers defining the preceding groups. If the last integer is not –1, then the size of the previous group (if any) will be repeatedly used for the remainder of the digits. If the last integer is –1, then no further grouping will be performed. The non-monetary numeric

formatting definitions for the POSIX locale follow; the code listing depicting the `localedef` input, the table representing the same information with the addition of `localeconv` values and `nl_langinfo` constants.

```
LC_NUMERIC
# This is the POSIX locale definition for
# the LC_NUMERIC category.
#
decimal_point "<period>"
thousands_sep ""
grouping -1
#
END LC_NUMERIC
```

Item	POSIX locale Value	langinfo Constant	localeconv() Value	localedef Value
decimal_point	"."	RADIXCHAR	"."	.
thousands_sep	n/a	THOUSEP	""	""
grouping	n/a	-	""	–1

The entry `n/a` indicates that the value is not available in the POSIX locale.

LC_TIME

The `LC_TIME` category defines the interpretation of the field descriptors supported by `date`(1) and affects the behavior of the `strftime`(3C), `wcsftime`(3C), `strptime`(3C), and `nl_langinfo`(3C) functions. Because the interfaces for C-language access and locale definition differ significantly, they are described separately. For locale definition, the following mandatory keywords are recognized:

`abday` Define the abbreviated weekday names, corresponding to the `%a` field descriptor (conversion specification in the `strftime( )`, `wcsftime( )`, and `strptime( )` functions). The operand consists of seven semicolon-separated strings, each surrounded by double-quotes. The first string is the abbreviated name of the day corresponding to Sunday, the second the abbreviated name of the day corresponding to Monday, and so on.

`day` Define the full weekday names, corresponding to the `%A` field descriptor. The operand consists of seven semicolon-separated strings, each surrounded by double-quotes. The first string is the full name of the day corresponding to Sunday, the second the full name of the day corresponding to Monday, and so on.

`abmon` Define the abbreviated month names, corresponding to the `%b` field descriptor. The operand consists of twelve semicolon-separated strings, each surrounded by double-quotes. The first string is the abbreviated name of the first month of the year (January), the second the abbreviated name of the second month, and so on.

`mon` Define the full month names, corresponding to the `%B` field descriptor. The operand consists of twelve semicolon-separated strings, each surrounded by double-quotes. The first string is the full name of the first month of the year (January), the second the full name of the second month, and so on.

`d_t_fmt` Define the appropriate date and time representation, corresponding to the `%c` field descriptor. The operand consists of a string, and can contain any combination of characters and field descriptors. In addition, the string can contain the escape sequences \\, `\a`, `\b`, `\f`, `\n`, `\r`, `\t`, `\v`.

`date_fmt` Define the appropriate date and time representation, corresponding to the `%C` field descriptor. The operand consists of a string, and can contain any combination of characters and field descriptors. In addition, the string can contain the escape sequences `\\`, `\a`, `\b`, `\f`, `\n`, `\r`, `\t`, `\v`.

`d_fmt` Define the appropriate date representation, corresponding to the `%x` field descriptor. The operand consists of a string, and can contain any combination of characters and field descriptors. In addition, the string can contain the escape sequences `\\`, `\a`, `\b`, `\f`, `\n`, `\r`, `\t`, `\v`.

`t_fmt` Define the appropriate time representation, corresponding to the `%X` field descriptor. The operand consists of a string, and can contain any combination of characters and field descriptors. In addition, the string can contain the escape sequences `\\`, `\a`, `\b`, `\f`, `\n`, `\r`, `\t`, `\v`.

`am_pm` Define the appropriate representation of the *ante meridiem* and *post meridiem* strings, corresponding to the `%p` field descriptor. The operand consists of two strings, separated by a semicolon, each surrounded by double-quotes. The first string represents the *ante meridiem* designation, the last string the *post meridiem* designation.

`t_fmt_ampm` — Define the appropriate time representation in the 12-hour clock format with `am_pm`, corresponding to the `%r` field descriptor. The operand consists of a string and can contain any combination of characters and field descriptors. If the string is empty, the 12-hour format is not supported in the locale.

`era` — Define how years are counted and displayed for each era in a locale. The operand consists of semicolon-separated strings. Each string is an era description segment with the format:

direction:*offset*:*start_date*:*end_date*:*era_name*:*era_format*

according to the definitions below. There can be as many era description segments as are necessary to describe the different eras.

The start of an era might not be the earliest point For example, the Christian era B.C. starts on the day before January 1, A.D. 1, and increases with earlier time.

direction — Either a + or a – character. The + character indicates that years closer to the *start_date* have lower numbers than those closer to the *end_date*. The – character indicates that years closer to the *start_date* have higher numbers than those closer to the *end_date*.

offset — The number of the year closest to the *start_date* in the era, corresponding to the `%Eg` and `%Ey` field descriptors.

start_date — A date in the form *yyyy*/*mm*/`dd`, where *yyyy*, *mm*, and `dd` are the year, month and day numbers respectively of the start of the era. Years prior to A.D. 1 are represented as negative numbers.

end_date — The ending date of the era, in the same format as the *start_date*, or one of the two special values –* or +*. The value –* indicates that the ending date is the beginning of time. The value +* indicates that the ending date is the end of time.

era_name	A string representing the name of the era, corresponding to the `%EC` field descriptor.
era_format	A string for formatting the year in the era, corresponding to the `%EG` and `%EY` field descriptors.

`era_d_fmt` Define the format of the date in alternative era notation, corresponding to the `%Ex` field descriptor.

`era_t_fmt` Define the locale's appropriate alternative time format, corresponding to the `%EX` field descriptor.

`era_d_t_fmt` Define the locale's appropriate alternative date and time format, corresponding to the `%Ec` field descriptor.

`alt_digits` Define alternative symbols for digits, corresponding to the `%O` field descriptor modifier. The operand consists of semicolon-separated strings, each surrounded by double-quotes. The first string is the alternative symbol corresponding with zero, the second string the symbol corresponding with one, and so on. Up to 100 alternative symbol strings can be specified. The `%O` modifier indicates that the string corresponding to the value specified via the field descriptor will be used instead of the value.

LC_TIME *C-language* **Access**

The following information can be accessed. These correspond to constants defined in <`langinfo.h`> and used as arguments to the `nl_langinfo`(3C) function.

`ABDAY_`*x*	The abbreviated weekday names (for example Sun), where *x* is a number from 1 to 7.
`DAY_`*x*	The full weekday names (for example Sunday), where *x* is a number from 1 to 7.
`ABMON_`*x*	The abbreviated month names (for example Jan), where *x* is a number from 1 to 12.
`MON_`*x*	The full month names (for example January), where *x* is a number from 1 to 12.
`D_T_FMT`	The appropriate date and time representation.
`D_FMT`	The appropriate date representation.
`T_FMT`	The appropriate time representation.
`AM_STR`	The appropriate ante-meridiem affix.
`PM_STR`	The appropriate post-meridiem affix.

`T_FMT_AMPM` — The appropriate time representation in the 12-hour clock format with `AM_STR` and `PM_STR`.

`ERA` — The era description segments, which describe how years are counted and displayed for each era in a locale. Each era description segment has the format:

direction:*offset*:*start_date*:*end_date*:*era_name*:*era_format*

according to the definitions below. There will be as many era description segments as are necessary to describe the different eras. Era description segments are separated by semicolons.

The start of an era might not be the earliest point For example, the Christian era B.C. starts on the day before January 1, A.D. 1, and increases with earlier time.

direction — Either a + or a – character. The + character indicates that years closer to the start_date have lower numbers than those closer to the *end_date*. The – character indicates that years closer to the start_date have higher numbers than those closer to the *end_date*.

offset — The number of the year closest to the start_date in the era.

start_date — A date in the form *yyyy*/*mm*/*dd*, where *yyyy*, *mm*, and `dd` are the year, month and day numbers respectively of the start of the era. Years prior to AD 1 are represented as negative numbers.

end_date — The ending date of the era, in the same format as the *start_date*, or one of the two special values –* or +*. The value –* indicates that the ending date is the beginning of time. The value +* indicates that the ending date is the end of time.

era_name The era, corresponding to the %EC conversion specification.

era_format The format of the year in the era, corresponding to the %EY and %EY conversion specifications.

ERA_D_FMT The era date format.

ERA_T_FMT The locale's appropriate alternative time format, corresponding to the %EX field descriptor.

ERA_D_T_FMT The locale's appropriate alternative date and time format, corresponding to the %Ec field descriptor.

ALT_DIGITS The alternative symbols for digits, corresponding to the %O conversion specification modifier. The value consists of semicolon-separated symbols. The first is the alternative symbol corresponding to zero, the second is the symbol corresponding to one, and so on. Up to 100 alternative symbols may be specified. The following table displays the correspondence between the items described above and the conversion specifiers used by date(1) and the strftime(3C), wcsftime(3C), and strptime(3C) functions.

localedef Keyword	langinfo Constant	Conversion Specifier
abday	ABDAY_*x*	%a
day	DAY_*x*	%A
abmon	ABMON_*x*	%b
mon	MON	%B
d_t_fmt	D_T_FMT	%c
date_fmt	DATE_FMT	%C
d_fmt	D_FMT	%x

localedef Keyword	langinfo Constant	Conversion Specifier
t_fmt	T_FMT	%X
am_pm	AM_STR	%p
am_pm	PM_STR	%p
t_fmt_ampm	T_FMT_AMPM	%r
era	ERA	%EC, %Eg, %EG, %Ey, %EY
era_d_fmt	ERA_D_FMT	%Ex
era_t_fmt	ERA_T_FMT	%EX
era_d_t_fmt	ERA_D_T_FMT	%Ec
alt_digits	ALT_DIGITS	%O

LC_TIME *General* **Information**

Although certain of the field descriptors in the POSIX locale (such as the name of the month) are shown with initial capital letters, this need not be the case in other locales. Programs using these fields may need to adjust the capitalization if the output is going to be used at the beginning of a sentence.

The LC_TIME descriptions of abday, day, mon, and abmon imply a Gregorian style calendar (7-day weeks, 12-month years, leap years, and so forth). Formatting time strings for other types of calendars is outside the scope of this document set.

As specified under date in Locale Definition and strftime(3C), the field descriptors corresponding to the optional keywords consist of a modifier followed by a traditional field descriptor (for instance %Ex). If the optional keywords are not supported by the implementation or are unspecified for the current locale, these field descriptors are treated as the traditional field descriptor. For instance, assume the following keywords:

```
alt_digits "0th";"1st";"2nd";"3rd";"4th";"5th";\

"6th";"7th";"8th";"9th";"10th"

d_fmt "The %Od day of %B in %Y"
```

On 7/4/1776, the %x field descriptor would result in “The 4th day of July in 1776” while 7/14/1789 would come out as “The 14 day of July in 1789” It can be noted that the above example is for illustrative purposes only; the %O modifier is primarily intended to provide for Kanji or Hindi digits in date formats.

LC_MESSAGES

The LC_MESSAGES category defines the format and values for affirmative and negative responses.

The following keywords are recognized as part of the locale definition file. The nl_langinfo(3C) function accepts upper-case versions of the first four keywords.

yesexpr
: The operand consists of an extended regular expression (see regex(5)) that describes the acceptable affirmative response to a question expecting an affirmative or negative response.

noexpr
: The operand consists of an extended regular expression that describes the acceptable negative response to a question expecting an affirmative or negative response.

yesstr
: The operand consists of a fixed string (not a regular expression) that can be used by an application for composition of a message that lists an acceptable affirmative response, such as in a prompt.

nostr
: The operand consists of a fixed string that can be used by an application for composition of a message that lists an acceptable negative response. The format and values for affirmative and negative responses of the POSIX locale follow; the code listing depicting the localedef input, the table representing the same information with the addition of nl_langinfo( ) constants.

```
LC_MESSAGES
# This is the POSIX locale definition for
# the LC_MESSAGES category.
#
yesexpr "<circumflex><left-square-bracket><y><Y><right-square-bracket>"
#
noexpr  "<circumflex><left-square-bracket><n><N><right-square-bracket>"
#
yesstr "yes"
nostr "no"
END LC_MESSAGES
```

localedef Keyword	langinfo Constant	POSIX Locale Value
yesexpr	YESEXPR	"^[yY]"
noexpr	NOEXPR	"^[nN]"
yesstr	YESSTR	"yes"
nostr	NOSTR	"no"

SEE ALSO

date(1), locale(1), localedef(1), sort(1), tr(1), uniq(1), localeconv(3C), nl_langinfo(3C), setlocale(3C), strcoll(3C), strftime(3C), strptime(3C), strxfrm(3C), wcscoll(3C), wcsftime(3C), wcsxfrm(3C), wctype(3C), attributes(5), charmap(5), extensions(5), regex(5)

NAME

man – macros to format Reference Manual pages

SYNOPSIS

nroff –man *filename*…

troff –man *filename*…

DESCRIPTION

These macros are used to lay out the reference pages in this manual. Note: if *filename* contains format input for a preprocessor, the commands shown above must be piped through the appropriate preprocessor. This is handled automatically by the man(1) command. See the "Conventions" section.

Any text argument *t* may be zero to six words. Quotes may be used to include SPACE characters in a "word". If *text* is empty, the special treatment is applied to the next input line with text to be printed. In this way .I may be used to italicize a whole line, or .SB may be used to make small bold letters.

A prevailing indent distance is remembered between successive indented paragraphs, and is reset to default value upon reaching a non-indented paragraph. Default units for indents *i* are ens.

Type font and size are reset to default values before each paragraph, and after processing font and size setting macros.

These strings are predefined by –man:

*R ‘®’, ‘(Reg)’ in nroff.

*S Change to default type size.

Requests

* n.t.l. = next text line; p.i. = prevailing indent

Request	*Cause Break*	*If no Argument*	*Explanation*
.B *t*	no	*t*=n.t.l.*	Text is in bold font.
.BI *t*	no	*t*=n.t.l.	Join words, alternating bold and italic.
.BR *t*	no	*t*=n.t.l.	Join words, alternating bold and roman.
.DT	no	.5i 1i...	Restore default tabs.
.HP *i*	yes	*i*=p.i.*	Begin paragraph with hanging indent. Set prevailing indent to *i*.
.I *t*	no	*t*=n.t.l.	Text is italic.
.IB *t*	no	*t*=n.t.l.	Join words, alternating italic and bold.
.IP *x i*	yes	*x*=""	Same as .TP with tag *x*.
.IR *t*	no	*t*=n.t.l.	Join words, alternating italic and roman.

Request	*Cause Break*	*If no Argument*	*Explanation*
`.IX` *t*	no	-	Index macro, for SunSoft internal use.
`.LP`	yes	-	Begin left-aligned paragraph. Set prevailing indent to .5i.
`.P`	yes	-	Same as `.LP`.
`.PD` *d*	no	*d*=.4v	Set vertical distance between paragraphs.
`.PP`	yes	-	Same as `.LP`.
`.RE`	yes	-	End of relative indent. Restores prevailing indent.
`.RB` *t*	no	*t*=n.t.l.	Join words, alternating roman and bold.
`.RI` *t*	no	*t*=n.t.l.	Join words, alternating roman and italic.
`.RS` *i*	yes	*i*=p.i.	Start relative indent, increase indent by *i*. Sets prevailing indent to .5i for nested indents.
`.SB` *t*	no	-	Reduce size of text by 1 point, make text bold.
`.SH` *t*	yes	-	Section Heading.
`.SM` *t*	no	*t*=n.t.l.	Reduce size of text by 1 point.
`.SS` *t*	yes	*t*=n.t.l.	Section Subheading.
`.TH` *n s d f m*	yes	-	Begin reference page *n*, of of section *s*; *d* is the date of the most recent change. If present, *f* is the left page footer; *m* is the main page (center) header. Sets prevailing indent and tabs to .5i.
`.TP` *i*	yes	*i*=p.i.	Begin indented paragraph, with the tag given on the next text line. Set prevailing indent to *i*.
`.TX` *t p*	no	-	Resolve the title abbreviation *t*; join to punctuation mark (or text) *p*.

Conventions

When formatting a manual page, `man` examines the first line to determine whether it requires special processing. For example a first line consisting of:

```
'\" t
```

indicates that the manual page must be run through the `tbl`(1) preprocessor.

A typical manual page for a command or function is laid out as follows:

.TH *title* [1-9]
: The name of the command or function, which serves as the title of the manual page. This is followed by the number of the section in which it appears.

.SH NAME
: The name, or list of names, by which the command is called, followed by a dash and then a one-line summary of the action performed. All in roman font, this section contains no `troff`(1) commands or escapes, and no macro requests. It is used to generate the `windex` database, which is used by the `whatis`(1) command.

.SH SYNOPSIS
: Commands:

  The syntax of the command and its arguments, as typed on the command line. When in boldface, a word must be typed exactly as printed. When in italics, a word can be replaced with an argument that you supply. References to bold or italicized items are not capitalized in other sections, even when they begin a sentence.

  Syntactic symbols appear in roman face:

  [ ]
  : An argument, when surrounded by brackets is optional.

  |
  : Arguments separated by a vertical bar are exclusive. You can supply only one item from such a list.

  . . .
  : Arguments followed by an ellipsis can be repeated. When an ellipsis follows a bracketed set, the expression within the brackets can be repeated.

  Functions:

  If required, the data declaration, or `#include` directive, is shown first, followed by the function declaration. Otherwise, the function declaration is shown.

.SH DESCRIPTION	A narrative overview of the command or function's external behavior. This includes how it interacts with files or data, and how it handles the standard input, standard output and standard error. Internals and implementation details are normally omitted. This section attempts to provide a succinct overview in answer to the question, "what does it do?" Literal text from the synopsis appears in constant width, as do literal filenames and references to items that appear elsewhere in the reference manuals. Arguments are italicized. If a command interprets either subcommands or an input grammar, its command interface or input grammar is normally described in a `USAGE` section, which follows the `OPTIONS` section. The `DESCRIPTION` section only describes the behavior of the command itself, not that of subcommands.
.SH OPTIONS	The list of options along with a description of how each affects the command's operation.
.SH RETURN VALUES	A list of the values the library routine will return to the calling program and the conditions that cause these values to be returned.
.SH EXIT STATUS	A list of the values the utility will return to the calling program or shell, and the conditions that cause these values to be returned.
.SH FILES	A list of files associated with the command or function.
.SH SEE ALSO	A comma-separated list of related manual pages, followed by references to other published materials.
.SH DIAGNOSTICS	A list of diagnostic messages and an explanation of each.
.SH BUGS	A description of limitations, known defects, and possible problems associated with the command or function.

FILES

`/usr/share/lib/tmac/an`

`/usr/share/man/windex`

SEE ALSO

`man`(1), `nroff`(1), `troff`(1), `whatis`(1)

Dale Dougherty and Tim O'Reilly, *Unix Text Processing*

NAME

mansun – macros to format Reference Manual pages

SYNOPSIS

nroff –mansun *filename*…

troff –mansun *filename*…

DESCRIPTION

These macros are used to lay out the reference pages in this manual. Note: if *filename* contains format input for a preprocessor, the commands shown above must be piped through the appropriate preprocessor. This is handled automatically by man(1). See the “Conventions” section.

Any text argument *t* may be zero to six words. Quotes may be used to include SPACE characters in a “word”. If *text* is empty, the special treatment is applied to the next input line with text to be printed. In this way .I may be used to italicize a whole line, or .SB may be used to make small bold letters.

A prevailing indent distance is remembered between successive indented paragraphs, and is reset to default value upon reaching a non-indented paragraph. Default units for indents *i* are ens.

Type font and size are reset to default values before each paragraph, and after processing font and size setting macros.

These strings are predefined by –mansun:

*R ‘®’, ‘(Reg)’ in nroff.

*S Change to default type size.

Requests

* n.t.l. = next text line; p.i. = prevailing indent

Request	*Cause Break*	*If no Argument*	*Explanation*
.B *t*	no	*t*=n.t.l.*	Text is in bold font.
.BI *t*	no	*t*=n.t.l.	Join words, alternating bold and italic.
.BR *t*	no	*t*=n.t.l.	Join words, alternating bold and Roman.
.DT	no	.5i 1i...	Restore default tabs.
.HP *i*	yes	*i*=p.i.*	Begin paragraph with hanging indent. Set prevailing indent to *i*.
.I *t*	no	*t*=n.t.l.	Text is italic.
.IB *t*	no	*t*=n.t.l.	Join words, alternating italic and bold.
.IP *x i*	yes	*x*=""	Same as .TP with tag *x*.
.IR *t*	no	*t*=n.t.l.	Join words, alternating italic and Roman.

Request	*Cause Break*	*If no Argument*	*Explanation*
`.IX` *t*	no	-	Index macro, for SunSoft internal use.
`.LP`	yes	-	Begin left-aligned paragraph. Set prevailing indent to .5i.
`.P`	yes	-	Same as `.LP`.
`.PD` *d*	no	*d*=.4v	Set vertical distance between paragraphs.
`.PP`	yes	-	Same as `.LP`.
`.RE`	yes	-	End of relative indent. Restores prevailing indent.
`.RB` *t*	no	*t*=n.t.l.	Join words, alternating Roman and bold.
`.RI` *t*	no	*t*=n.t.l.	Join words, alternating Roman and italic.
`.RS` *i*	yes	*i*=p.i.	Start relative indent, increase indent by *i*. Sets prevailing indent to .5i for nested indents.
`.SB` *t*	no	-	Reduce size of text by 1 point, make text bold.
`.SH` *t*	yes	-	Section Heading.
`.SM` *t*	no	*t*=n.t.l.	Reduce size of text by 1 point.
`.SS` *t*	yes	*t*=n.t.l.	Section Subheading.
`.TH` *n s d f m*	yes	-	Begin reference page *n*, of of section *s*; *d* is the date of the most recent change. If present, *f* is the left page footer; *m* is the main page (center) header. Sets prevailing indent and tabs to .5i.
`.TP` *i*	yes	*i*=p.i.	Begin indented paragraph, with the tag given on the next text line. Set prevailing indent to *i*.
`.TX` *t p*	no	-	Resolve the title abbreviation *t*; join to punctuation mark (or text) *p*.

Conventions

When formatting a manual page, `mansun` examines the first line to determine whether it requires special processing. For example a first line consisting of:

```
'\" t
```

indicates that the manual page must be run through the `tbl`(1) preprocessor.

A typical manual page for a command or function is laid out as follows:

.TH *title* [1-8]
: The name of the command or function, which serves as the title of the manual page. This is followed by the number of the section in which it appears.

.SH NAME
: The name, or list of names, by which the command is called, followed by a dash and then a one-line summary of the action performed. All in Roman font, this section contains no `troff`(1) commands or escapes, and no macro requests. It is used to generate the `windex` database, which is used by the `whatis`(1) command.

.SH SYNOPSIS
: Commands:

  The syntax of the command and its arguments, as typed on the command line. When in boldface, a word must be typed exactly as printed. When in italics, a word can be replaced with an argument that you supply. References to bold or italicized items are not capitalized in other sections, even when they begin a sentence.

  Syntactic symbols appear in Roman face:

[ ]	An argument, when surrounded by brackets is optional.
|	Arguments separated by a vertical bar are exclusive. You can supply only one item from such a list.
. . .	Arguments followed by an ellipsis can be repeated. When an ellipsis follows a bracketed set, the expression within the brackets can be repeated.

  Functions:

  If required, the data declaration, or `#include` directive, is shown first, followed by the function declaration. Otherwise, the function declaration is shown.

.SH DESCRIPTION	A narrative overview of the command or function's external behavior. This includes how it interacts with files or data, and how it handles the standard input, standard output and standard error. Internals and implementation details are normally omitted. This section attempts to provide a succinct overview in answer to the question, "what does it do?" Literal text from the synopsis appears in constant width, as do literal filenames and references to items that appear elsewhere in the reference manuals. Arguments are italicized. If a command interprets either subcommands or an input grammar, its command interface or input grammar is normally described in a `USAGE` section, which follows the `OPTIONS` section. The `DESCRIPTION` section only describes the behavior of the command itself, not that of subcommands.
.SH OPTIONS	The list of options along with a description of how each affects the command's operation.
.SH FILES	A list of files associated with the command or function.
.SH SEE ALSO	A comma-separated list of related manual pages, followed by references to other published materials.
.SH DIAGNOSTICS	A list of diagnostic messages and an explanation of each.
.SH BUGS	A description of limitations, known defects, and possible problems associated with the command or function.

FILES

`/usr/share/lib/tmac/ansun`

`/usr/share/man/windex`

SEE ALSO

`man`(1), `nroff`(1), `troff`(1), `whatis`(1)

Dale Dougherty and Tim O'Reilly, *Unix Text Processing*

NAME

me – macros for formatting papers

SYNOPSIS

nroff –me [*options*] *filename*…

troff –me [*options*] *filename*…

DESCRIPTION

This package of `nroff` and `troff` macro definitions provides a canned formatting facility for technical papers in various formats. When producing 2-column output on a terminal, filter the output through `col`(1).

The macro requests are defined below. Many `nroff` and `troff` requests are unsafe in conjunction with this package, however, these requests may be used with impunity after the first `.pp`:

.bp begin new page

.br break output line here

.sp *n* insert *n* spacing lines

.ls *n* (line spacing) *n*=1 single, *n*=2 double space

.na no alignment of right margin

.ce *n* center next *n* lines

.ul *n* underline next *n* lines

.sz +*n* add *n* to point size

Output of the `eqn`(1), `neqn`(1), `refer`(1), and `tbl`(1) preprocessors for equations and tables is acceptable as input.

REQUESTS

In the following list, “initialization” refers to the first `.pp`, `.lp`, `.ip`, `.np`, `.sh`, or `.uh` macro. This list is incomplete.

Request	*Initial Value*	*Cause Break*	*Explanation*
`.(c`	-	yes	Begin centered block.
`.(d`	-	no	Begin delayed text.
`.(f`	-	no	Begin footnote.
`.(l`	-	yes	Begin list.
`.(q`	-	yes	Begin major quote.
`.(x`*x*	-	no	Begin indexed item in index *x*.
`.(z`	-	no	Begin floating keep.
`.)c`	-	yes	End centered block.

Request	*Initial Value*	*Cause Break*	*Explanation*
.)d	-	yes	End delayed text.
.)f	-	yes	End footnote.
.)l	-	yes	End list.
.)q	-	yes	End major quote.
.)x	-	yes	End index item.
.)z	-	yes	End floating keep.
.++ *m H*	-	no	Define paper section. *m* defines the part of the paper, and can be C (chapter), A (appendix), P (preliminary, for instance, abstract, table of contents, etc.), B (bibliography), RC (chapters renumbered from page one each chapter), or RA (appendix renumbered from page one).
.+c *T*	-	yes	Begin chapter (or appendix, etc., as set by .++). *T* is the chapter title.
.1c	1	yes	One column format on a new page.
.2c	1	yes	Two column format.
.EN	-	yes	Space after equation produced by eqn or neqn.
.EQ *x y*	-	yes	Precede equation; break out and add space. Equation number is *y*. The optional argument *x* may be *I* to indent equation (default), *L* to left-adjust the equation, or *C* to center the equation.
.GE	-	yes	End *gremlin* picture.

Request	*Initial Value*	*Cause Break*	*Explanation*
`.GS`	-	yes	Begin *gremlin* picture.
`.PE`	-	yes	End `pic` picture.
`.PS`	-	yes	Begin `pic` picture.
`.TE`	-	yes	End table.
`.TH`	-	yes	End heading section of table.
`.TS` *x*	-	yes	Begin table; if *x* is *H* table has repeated heading.
`.ac` *A N*	-	no	Set up for ACM style output. *A* is the Author's name(s), *N* is the total number of pages. Must be given before the first initialization.
`.b` *x*	no	no	Print *x* in boldface; if no argument switch to boldface.
`.ba` *+n*	0	yes	Augments the base indent by *n*. This indent is used to set the indent on regular text (like paragraphs).
`.bc`	no	yes	Begin new column.
`.bi` *x*	no	no	Print *x* in bold italics (nofill only).
`.bu`	-	yes	Begin bulleted paragraph.
`.bx` *x*	no	no	Print *x* in a box (nofill only).
`.ef` *'x'y'z*	'''''	no	Set even footer to *x y z*.
`.eh` *'x'y'z*	'''''	no	Set even header to *x y z*.
`.fo` *'x'y'z*	'''''	no	Set footer to *x y z*.
`.hx`	-	no	Suppress headers and footers on next page.
`.he` *'x'y'z*	'''''	no	Set header to *x y z*.
`.hl`	-	yes	Draw a horizontal line.

Request	*Initial Value*	*Cause Break*	*Explanation*
`.i` *x*	no	no	Italicize *x*; if *x* missing, italic text follows.
`.ip` *x y*	no	yes	Start indented paragraph, with hanging tag *x*. Indentation is *y* ens (default 5).
`.lp`	yes	yes	Start left-blocked paragraph.
`.lo`	-	no	Read in a file of local macros of the form `.*`*x*. Must be given before initialization.
`.np`	1	yes	Start numbered paragraph.
`.of` *'x'y'z*	'''''	no	Set odd footer to x y z.
`.oh` *'x'y'z*	'''''	no	Set odd header to x y z.
`.pd`	-	yes	Print delayed text.
`.pp`	no	yes	Begin paragraph. First line indented.
`.r`	yes	no	Roman text follows.
`.re`	-	no	Reset tabs to default values.
`.sc`	no	no	Read in a file of special characters and diacritical marks. Must be given before initialization.
`.sh` *n x*	-	yes	Section head follows, font automatically bold. *n* is level of section, *x* is title of section.
`.sk`	no	no	Leave the next page blank. Only one page is remembered ahead.
`.sm` *x*	-	no	Set *x* in a smaller pointsize.
`.sz` +*n*	10p	no	Augment the point size by *n* points.
`.th`	no	no	Produce the paper in thesis format. Must be given before initialization.

Request	*Initial Value*	*Cause Break*	*Explanation*
`.tp`	no	yes	Begin title page.
`.u` *x*	-	no	Underline argument (even in `troff`). (Nofill only).
`.uh`	-	yes	Like `.sh` but unnumbered.
`.xp` *x*	-	no	Print index *x*.

FILES

`/usr/share/lib/tmac/e`
`/usr/share/lib/tmac/*.me`

SEE ALSO

`col`(1), `eqn`(1), `nroff`(1), `refer`(1), `tbl`(1), `troff`(1)

NAME

mm – text formatting (memorandum) macros

SYNOPSIS

nroff –mm [*options*] *filename*…

troff –mm [*options*] *filename*…

DESCRIPTION

This package of `nroff`(1) and `troff`(1) macro definitions provides a formatting facility for various styles of articles, theses, and books. When producing 2-column output on a terminal or lineprinter, or when reverse line motions are needed, filter the output through `col`(1). All external –mm macros are defined below.

Note: this –mm macro package is an extended version written at Berkeley and is a superset of the standard –mm macro packages as supplied by Bell Labs. Some of the Bell Labs macros have been removed; for instance, it is assumed that the user has little interest in producing headers stating that the memo was generated at Whippany Labs.

Many `nroff` and `troff` requests are unsafe in conjunction with this package. However, the first four requests below may be used with impunity after initialization, and the last two may be used even before initialization:

`.bp` begin new page

`.br` break output line

`.sp`*n* insert n spacing lines

`.ce`*n* center next n lines

`.ls`*n* line spacing: *n*=`1` single, *n*=`2` double space

`.na` no alignment of right margin

Font and point size changes with `\f` and `\s` are also allowed; for example, `\fIword\fR` will italicize *word.* Output of the `tbl`(1), `eqn`(1) and `refer`(1) preprocessors for equations, tables, and references is acceptable as input.

REQUESTS

Here is a table of macros.

Macro Name	**Initial Value**	**Break? Reset?**	**Explanation**
`.1C`	on	y,y	one column format on a new page
`.2C [ l ]`	–	y,y	two column format *l*=line length
`.AE`	–	y	end abstract

Macro Name	Initial Value	Break? Reset?	Explanation
.AL [*t*] [*i*] [*s*]	*t*=1;*i*=.Li;*s*=0	y	Start automatic list type *t*=[1,A,a,I,i] 1=arabic numbers; A=uppercase letters a=lowercase letters; I=uppercase Roman numerals; i=lowercase Roman numerals indentation *i*; separation *s*
.AS *m* [*n*]	*n*=0	y	begin abstract
.AU	–	y	author's name
.AV *x*	–	y	signature and date line of verifier *x*
.B *x*	–	n	embolden *x*; if no *x*, switch to boldface
.BE	–	y	end block text
.BI *x y*	–	n	embolden *x* and underline *y*
.BL	–	y	bullet list
.BR *x y*	–	n	embolden *x* and use Roman font for *y*
.BS	–	n	start block text
.CN	–	y	same as .DE (nroff)
.CS	–	y	cover sheet
.CW	–	n	same as .DS I (nroff)
.DE	–	y	end display
.DF [*p*] [*f*] [*rp*]	*p*=L;*f*=N	y	start floating display; position *p*=[L,C,CB] L=left; I=indent; C=center; CB=center block fill *f*=[N,Y]; right position *rp* (fill only)
.DL [*i*] [*s*]	–	y	start dash list
.DS [*p*] [*f*] [*rp*]	*p*=L;*f*=N	y	begin static display (see .DF for argument descriptions)
.EC *x* [*n*]	*n*=1	y	equation title; equation *x*; number *n*
.EF *x*	–	n	even footer appears at the bottom of even-numbered pages; *x*="*l*'*c*'*r*" *l*=left; *c*=center; *r*=right

Macro Name	Initial Value	Break? Reset?	Explanation
.EH *x*	–	n	even header appears at the top of even-numbered pages; *x*="*l*'*c*'*r*" *l*=left; *c*=center; *r*=right
.EN	–	y	end displayed equation produced by eqn
.EQ	–	y	break out equation produced by eqn
.EX *x* [*n*]	*n*=1	y	exhibit title; exhibit *x*
			number *n*
.FD [*f*] [*r*]	*f*=10;*r*=1	n	set footnote style format *f*=[0-11]; renumber *r*=[0,1]
.FE	–	y	end footnote
.FG *x* [*n*]	*n*=1	y	figure title; figure *x*; number *n*
.FS	–	n	start footnote
.H *l* [*t*]	–	y	produce numbered heading level *l*=[1-7]; title *t*
.HU *t*	–	y	produce unnumbered heading; title *t*
.I *x*	–	n	underline *x*
.IB *x y*	–	n	underline *x* and embolden *y*
.IR *x y*	–	n	underline *x* and use Roman font on *y*
.LE [*s*]	*s*=0	y	end list; separation *s*
.LI [*m*] [*p*]	–	y	start new list item; mark *m*
			prefix *p* (mark only)
.ML *m* [*i*] [*s*]	*s*=0	y	start marked list; mark *m* indentation *i*; separation *s*=[0,1]
.MT *x*		y	memo title; title *x*
.ND *x*		n	no date in page footer; *x* is date on cover
.NE	–	y	end block text
.NS	–	y	start block text

Macro Name	Initial Value	Break? Reset?	Explanation
.OF *x*	–	n	odd footer appears at the bottom of odd-numbered pages; *x*="*l*'*c*'*r*" *l*=left; *c*=center; *r*=right
.OF *x*	–	n	odd header appears at the top of odd-numbered pages; *x*="*l*'*c*'*r*" *l*=left; *c*=center; *r*=right
.OP	–	y	skip to the top of an odd-number page
.P [*t*]	*t*=0	y,y	begin paragraph; *t*=[0,1] 0=justified; 1=indented
.PF *x*	–	n	page footer appears at the bottom of every page; *x*="*l*'*c*'*r*" *l*=left; *c*=center; *r*=right
.PH *x*	–	n	page header appears at the top of every page; *x*="*l*'*c*'*r*" *l*=left; *c*=center; *r*=right
.R	on	n	return to Roman font
.RB *x y*	–	n	use Roman on *x* and embolden *y*
.RI *x y*	–	n	use Roman on *x* and underline *y*
.RP *x*	-	y,y	released paper format ? *x*=no stops title on first
.RS	5n	y,y	right shift: start level of relative indentation
.S *m n*	–	n	set character point size & vertical space character point size *m*; vertical space *n*
.SA *x*	*x*=1	n	justification; *x*=[0,1]
.SK *x*	–	y	skip *x* pages
.SM	–	n	smaller; decrease point size by 2
.SP [*x*]	–	y	leave *x* blank lines
.TB *x* [*n*]	*n*=1	y	table title; table *x*; number *n*
.TC	–	y	print table of contents (put at end of input file)
.TE	–	y	end of table processed by tbl
.TH	–	y	end multi-page header of table

Macro Name	Initial Value	Break? Reset?	Explanation
`.TL`	–	n	title in boldface and two points larger
`.TM`	–	n	UC Berkeley thesis mode
`.TP` *i*	y	y	*i*=p.i. Begin indented paragraph, with the tag given on the next text line. Set prevailing indent to *i*.
`.TS` *x*	–	y,y	begin table; if *x*=`H` table has multi-page header
`.TY`	–	y	display centered title `CONTENTS`
`.VL` *i* [*m*] [*s*]	*m*=`0`;*s*=`0`	y	start variable-item list; indentation *i* mark-indentation *m*; separation *s*

REGISTERS

Formatting distances can be controlled in `–mm` by means of built-in number registers. For example, this sets the line length to 6.5 inches:

```
.nr  LL  6.5i
```

Here is a table of number registers and their default values:

Name	Register Controls	Takes Effect	Default
`Cl`	contents level	table of contents	2
`De`	display eject	display	0
`Df`	display floating	display	5
`Ds`	display spacing	display	1v
`Hb`	heading break	heading	2
`Hc`	heading centering	heading	0
`Hi`	heading indent	heading	1
`Hi`	heading spacing	heading	1
`Hu`	heading unnumbered	heading	2
`Li`	list indentation	list	6 (`nroff`) 5 (`troff`)
`Ls`	list spacing	list	6

Name	Register Controls	Takes Effect	Default
Pi	paragraph indent	paragraph	5
Pt	paragraph type	paragraph	1
Si	static indent	display	5 (nroff) 3 (troff)

When resetting these values, make sure to specify the appropriate units. Setting the line length to 7, for example, will result in output with one character per line. Setting Pi to 0 suppresses paragraph indentation

Here is a list of string registers available in –mm; they may be used anywhere in the text:

Name	String's Function
*Q	quote (" in nroff, ‘‘ in troff)
*U	unquote (" in nroff, ’’ in troff)
*-	dash (-- in nroff, — in troff)
*(MO	month (month of the year)
*(DY	day (current date)
**	automatically numbered footnote
*′	acute accent (before letter)
*‘	grave accent (before letter)
*^	circumflex (before letter)
*,	cedilla (before letter)
*:	umlaut (before letter)
*~	tilde (before letter)
\(BU	bullet item
\(DT	date (*month day, yr*)
\(EM	em dash
\(Lf	LIST OF FIGURES title
\(Lt	LIST OF TABLES title
\(Lx	LIST OF EXHIBITS title

Name	String's Function
`\(Le`	`LIST OF EQUATIONS` title
`\(Rp`	`REFERENCES` title
`\(Tm`	trademark character (TM)

When using the extended accent mark definitions available with `.AM`, these strings should come after, rather than before, the letter to be accented.

FILES

`/usr/share/lib/tmac/m`

`/usr/share/lib/tmac/mm.[nt]` `nroff` and `troff` definitions of `mm`.

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWdoc

SEE ALSO

`col`(1), `eqn`(1), `nroff`(1), `refer`(1), `tbl`(1), `troff`(1), `attributes`(5)

BUGS

Floating keeps and regular keeps are diverted to the same space, so they cannot be mixed together with predictable results.

NAME

ms – text formatting macros

SYNOPSIS

nroff –`ms` [*options*] *filename*…

troff –`ms` [*options*] *filename*…

DESCRIPTION

This package of `nroff`(1) and `troff`(1) macro definitions provides a formatting facility for various styles of articles, theses, and books. When producing 2-column output on a terminal or lineprinter, or when reverse line motions are needed, filter the output through `col`(1). All external –`ms` macros are defined below.

Note: this –`ms` macro package is an extended version written at Berkeley and is a superset of the standard –`ms` macro packages as supplied by Bell Labs. Some of the Bell Labs macros have been removed; for instance, it is assumed that the user has little interest in producing headers stating that the memo was generated at Whippany Labs.

Many `nroff` and `troff` requests are unsafe in conjunction with this package. However, the first four requests below may be used with impunity after initialization, and the last two may be used even before initialization:

`.bp` begin new page

`.br` break output line

`.sp` *n* insert n spacing lines

`.ce` *n* center next n lines

`.ls` *n* line spacing: *n*=`1` single, *n*=`2` double space

`.na` no alignment of right margin

Font and point size changes with `\f` and `\s` are also allowed; for example, `\fIword\fR` will italicize *word.* Output of the `tbl`(1), `eqn`(1) and `refer`(1) preprocessors for equations, tables, and references is acceptable as input.

REQUESTS

Macro Name	Initial Value	Break? Reset?	Explanation
`.AB` *x*	–	y	begin abstract; if *x*=no do not label abstract
`.AE`	–	y	end abstract
`.AI`	–	y	author's institution
`.AM`	–	n	better accent mark definitions
`.AU`	–	y	author's name

Macro Name	Initial Value	Break? Reset?	Explanation
`.B` *x*	–	n	embolden *x*; if no *x*, switch to boldface
`.B1`	–	y	begin text to be enclosed in a box
`.B2`	–	y	end boxed text and print it
`.BT`	date	n	bottom title, printed at foot of page
`.BX` *x*	–	n	print word *x* in a box
`.CM`	if t	n	cut mark between pages
`.CT`	–	y,y	chapter title: page number moved to CF (TM only)
`.DA` *x*	if n	n	force date *x* at bottom of page; today if no *x*
`.DE`	–	y	end display (unfilled text) of any kind
`.DS` *x y*	I	y	begin display with keep; *x*=I, L, C, B; *y*=indent
`.ID` *y*	8n,.5i	y	indented display with no keep; *y*=indent
`.LD`	–	y	left display with no keep
`.CD`	–	y	centered display with no keep
`.BD`	–	y	block display; center entire block
`.EF` *x*	–	n	even page footer *x* (3 part as for `.tl`)
`.EH` *x*	–	n	even page header *x* (3 part as for `.tl`)
`.EN`	–	y	end displayed equation produced by `eqn`
`.EQ` *x y*	–	y	break out equation; *x*=L,I,C; *y*=equation number
`.FE`	–	n	end footnote to be placed at bottom of page
`.FP`	–	n	numbered footnote paragraph; may be redefined
`.FS` *x*	–	n	start footnote; *x* is optional footnote label
`.HD`	undef	n	optional page header below header margin
`.I` *x*	–	n	italicize *x*; if no *x*, switch to italics
`.IP` *x y*	–	y,y	indented paragraph, with hanging tag *x*; *y*=indent
`.IX` *x y*	–	y	index words *x y* and so on (up to 5 levels)
`.KE`	–	n	end keep of any kind
`.KF`	–	n	begin floating keep; text fills remainder of page
`.KS`	–	y	begin keep; unit kept together on a single page

Macro Name	**Initial Value**	**Break? Reset?**	**Explanation**
`.LG`	–	n	larger; increase point size by 2
`.LP`	–	y,y	left (block) paragraph.
`.MC` *x*	–	y,y	multiple columns; *x*=column width
`.ND` *x*	if t	n	no date in page footer; *x* is date on cover
`.NH` *x y*	–	y,y	numbered header; *x*=level, *x*=0 resets, *x*=S sets to *y*
`.NL`	10p	n	set point size back to normal
`.OF` *x*	–	n	odd page footer *x* (3 part as for `.tl`)
`.OH` *x*	–	n	odd page header *x* (3 part as for `.tl`)
`.P1`	if TM	n	print header on first page
`.PP`	–	y,y	paragraph with first line indented
`.PT`	- % -	n	page title, printed at head of page
`.PX` *x*	–	y	print index (table of contents); *x*=no suppresses title
`.QP`	–	y,y	quote paragraph (indented and shorter)
`.R`	on	n	return to Roman font
`.RE`	5n	y,y	retreat: end level of relative indentation
`.RP` *x*	–	n	released paper format; *x*=no stops title on first page
`.RS`	5n	y,y	right shift: start level of relative indentation
`.SH`	–	y,y	section header, in boldface
`.SM`	–	n	smaller; decrease point size by 2
`.TA`	8n,5n	n	set TAB characters to 8n 16n ... (`nroff`) or 5n 10n ... (`troff`)
`.TC` *x*	–	y	print table of contents at end; *x*=no suppresses title
`.TE`	–	y	end of table processed by `tbl`
`.TH`	–	y	end multi-page header of table
`.TL`	–	y	title in boldface and two points larger
`.TM`	off	n	UC Berkeley thesis mode
`.TS` *x*	–	y,y	begin table; if *x*=H table has multi-page header

Macro Name	**Initial Value**	**Break? Reset?**	**Explanation**
`.UL` *x*	–	n	underline *x*, even in `troff`
`.UX` *x*	–	n	UNIX; trademark message first time; *x* appended
`.XA` *x y*	–	y	another index entry; *x*=page or no for none; y=indent
`.XE`	–	y	end index entry (or series of `.IX` entries)
`.XP`	–	y,y	paragraph with first line indented, others indented
`.XS` *x y*	–	y	begin index entry; *x*=page or no for none; *y*=indent
`.1C`	on	y,y	one column format, on a new page
`.2C`	–	y,y	begin two column format
`.]–`	–	n	beginning of `refer` reference
`.[0`	–	n	end of unclassifiable type of reference
`.[N`	–	n	N= 1:journal-article, 2:book, 3:book-article, 4:report

REGISTERS

Formatting distances can be controlled in `–ms` by means of built-in number registers. For example, this sets the line length to 6.5 inches:

```
.nr  LL  6.5i
```

Here is a table of number registers and their default values:

Name	**Register Controls**	**Takes Effect**	**Default**
`PS`	point size	paragraph	10
`VS`	vertical spacing	paragraph	12
`LL`	line length	paragraph	6i
`LT`	title length	next page	same as `LL`
`FL`	footnote length	next `.FS`	5.5i
`PD`	paragraph distance	paragraph	1v (if n), .3v (if t)
`DD`	display distance	displays	1v (if n), .5v (if t)
`PI`	paragraph indent	paragraph	5n
`QI`	quote indent	next `.QP`	5n

Name	Register Controls	Takes Effect	Default
FI	footnote indent	next .FS	2n
PO	page offset	next page	0 (if n), ≈1i (if t)
HM	header margin	next page	1i
FM	footer margin	next page	1i
FF	footnote format	next .FS	0 (1, 2, 3 available)

When resetting these values, make sure to specify the appropriate units. Setting the line length to 7, for example, will result in output with one character per line. Setting FF to 1 suppresses footnote superscripting; setting it to 2 also suppresses indentation of the first line; and setting it to 3 produces an .IP-like footnote paragraph.

Here is a list of string registers available in –ms; they may be used anywhere in the text:

Name	String’s Function
*Q	quote (" in nroff, “ in troff)
*U	unquote (" in nroff, ” in troff)
*-	dash (-- in nroff, — in troff)
*(MO	month (month of the year)
*(DY	day (current date)
**	automatically numbered footnote
*′	acute accent (before letter)
*‘	grave accent (before letter)
*^	circumflex (before letter)
*,	cedilla (before letter)
*:	umlaut (before letter)
*~	tilde (before letter)

When using the extended accent mark definitions available with .AM, these strings should come after, rather than before, the letter to be accented.

FILES

/usr/share/lib/tmac/s
/usr/share/lib/tmac/ms.???

SEE ALSO

col(1), eqn(1), nroff(1), refer(1), tbl(1), troff(1)

BUGS

Floating keeps and regular keeps are diverted to the same space, so they cannot be mixed together with predictable results.

NAME

nfssec – overview of NFS security modes

DESCRIPTION

The `mount_nfs`(1M) and `share_nfs`(1M) commands each provide a way to specify the security mode to be used on an NFS file system through the `sec=`*mode* option. *mode* can be either `sys`, `dh`, `krb4`, or `none.` These security modes may also be added to the automount maps. Note that `mount_nfs`(1M) and `automount`(1M) do not support `sec=`*none* at this time.

The `sec=`*mode* option on the `share_nfs`(1M) command line establishes the security mode of NFS servers. If the NFS connection uses the NFS Version 3 protocol, the NFS clients must query the server for the appropriate *mode* to use. If the NFS connection uses the NFS Version 2 protocol, then the NFS client will use the default security mode, which is currently `sys`. NFS clients may force the use of a specific security mode by specifying the `sec=`*mode* option on the command line. However, if the file system on the server is not shared with that security mode, the client may be denied access.

If the NFS client wants to authenticate the NFS server using a particular (stronger) security mode, the client will want to specify the security mode to be used, even if the connection uses the NFS Version 3 protocol. This guarantees that an attacker masquerading as the server does not compromise the client.

The NFS security modes are described as follows:

`sys` Use `AUTH_SYS` authentication. The user's UNIX user-id and group-ids are passed in the clear on the network, unauthenticated by the NFS server. This is the simplest security method and requires no additional administration. It is the default used by Solaris NFS Version 2 clients and Solaris NFS servers.

`dh` Use a Diffie-Hellman public key system (`AUTH_DES,` which is referred to as `AUTH_DH` in the forthcoming Internet RFC).

`krb4` Use the Kerberos Version 4 authentication system (`AUTH_KERB,` which is referred to as `AUTH_KERB4` in a forthcoming Internet RFC).

`none` Use null authentication (`AUTH_NONE).` NFS clients using `AUTH_NONE` have no identity and are mapped to the anonymous user `nobody` by NFS servers. A client using a security mode other than the one with which a Solaris NFS server shares the file system will have its security mode mapped to `AUTH_NONE.` In this case, if the file system is shared with `sec=`*none*, users from the client will be mapped to the anonymous user. The NFS security mode `none` is supported by `share_nfs`(1M), but not by `mount_nfs`(1M) or `automount`(1M).

FILES

`/etc/nfssec.conf`	NFS security service configuration file.

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

`automount`(1M), `mount_nfs`(1M), `share_nfs`(1M), `rpc_clnt_auth`(3NSL), `secure_rpc`(3NSL), `attributes`(5)

NOTES

`/etc/nfssec.conf` lists the NFS security services. Do not edit this file. It is not intended to be user-configurable.

NAME

pam_dial_auth – authentication management PAM module for dialups

SYNOPSIS

`/usr/lib/security/pam_dial_auth.so.1`

DESCRIPTION

The dialup PAM module, `/usr/lib/security/pam_dial_auth.so.1`, authenticates a user according to the `/etc/dialups` and `/etc/d_passwd` files. Only `pam_sm_authenticate( )` is implemented within this module. `pam_sm_setcred( )` is a null function. `/usr/lib/security/pam_dial_auth.so.1` is designed to be stacked immediately below the `/usr/lib/security/pam_unix.so.1` module for the login service.

`pam_sm_authenticate( )` performs authentication only if both the `/etc/dialups` and `/etc/d_passwd` files exist. The user's terminal line is checked against entries in the `/etc/dialups` file. If there is a match, the user's shell is compared against entries in the `/etc/d_passwd` file. If there is a matching entry, the user is prompted for a password which is validated against the entry in the `/etc/d_passwd` file. If the passwords match, the user is authenticated. The following option may be passed in to this service module:

`debug` `syslog`(3C) debugging information at `LOG_DEBUG` level.

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
MT Level	MT-Safe with exceptions

SEE ALSO

`pam`(3PAM), `pam_authenticate`(3PAM), `d_passwd`(4), `dialups`(4), `libpam`(3LIB), `pam.conf`(4), `attributes`(5)

NOTES

The interfaces in `libpam( )` are MT-Safe only if each thread within the multi-threaded application uses its own PAM handle.

NAME

pam_krb5 – authentication, account, session, and password management PAM modules for Kerberos V5

SYNOPSIS

`/usr/lib/security/pam_krb5.so.1`

DESCRIPTION

The Kerberos V5 service module for PAM, `/usr/lib/security/pam_krb5.so.1`, provides functionality for all four PAM modules: authentication, account management, session management,and password management. The `pam_krb5.so.1` module is a shared object that can be dynamically loaded to provide the necessary functionality upon demand. Its path is specified in the PAM configuration file.

Kerberos Authentication Module

The Kerberos V5 authentication component provides functions to verify the identity of a user, (`pam_sm_authenticate()`) and to refresh the Kerberos credentials cache (`pam_sm_setcred()`). `pam_sm_authenticate()` authenticates a user principal though the Kerberos authentication service. If the authentication request is successful, the authentication service will send a ticket-granting ticket (tgt) back to the `pam_krb5.so.1` module, which will store the tgt in the credentials cache for later use by Kerberized network applications.

The following options may be passed to the Kerberos V5 authentication module:

`acceptor`
: Prevent the PAM module from performing the authentication service exchange used to obtain the initial ticket-granting ticket. This should be used on Kerberos application servers since the initial ticket is not needed.

`debug`
: `syslog`(3C) debugging information at `LOG_DEBUG` level.

`nowarn`
: Turn off warning messages.

`use_first_pass`
: Request Kerberos V5 authentication with the user’s initial password (entered when the user authenticated to the first authentication module in the stack). If Kerberos V5 authentication fails, or if no password has been entered, it quits and does not prompt the user for a password. This option should only be used if the authentication service is designated as *optional* in the `pam.conf` configuration file.

`try_first_pass`
: Request Kerberos V5 authentication with the user’s initial password (entered when the user authenticated to the first authentication module in the stack). If Kerberos V5 authentication fails, or if no password has been entered,

	prompt the user for a password with the prompt "`Kerberos Password:`".
`use_xfn_pass`	Request Kerberos V5 authentication with a mapped password that has been stored under XFN. If Kerberos V5 authentication fails, or if no password has been entered, it quits and does not prompt the user for a password. This option should only be used if the authentication service is designated as *optional* in the `pam.conf` configuration file.
`try_xfn_Pass`	Request Kerberos V5 authentication with a mapped password that has been stored under XFN. If Kerberos V5 authentication fails, or if no password has been stored, prompt the user for a password with the prompt "`Kerberos Password:`".

Kerberos V5 Account Management Module

The account management module returns success and performs no funtions. This component is a null function.

Kerberos V5 Session Management Module

The Kerberos V5 session management component provides functions to initiate `pam_sm_open_session()` and terminate `pam_sm_close_session()` Kerberos V5 sessions. For Kerberos V5, `pam_sm_open_session` is a null function. `pam_close_session` will destory a principal's credential cache as well as the in kernel Kerberos credentials if the session being closed is the last open session on this server for the calling principal.

Kerberos V5 Password Management Module

The Kerberos V5 password management component provides a function to change passwords `pam_sm_chauthtok()` in the Key Distribution Center (KDC) database. The following options may be passed in to the Kerberos V5 password module:

`debug`	`syslog`(3C) Debugging information at `LOG_DEBUG` level.
`nowarn`	Turn off warning messages.
`use_first_pass`	Request Kerberos V5 authentication with the user's initial password (entered when the user authenticated to the first authentication module in the stack). If Kerberos V5 authentication fails, or if no password has been entered, it quits and does not prompt the user for a password. If authentication succeeds, the user is prompted by "`New KRB5 password:`" for a new password. The user is then prompted a second time for the

new password for verification and the KDC database is updated with the new password if both responses match.

`try_first_pass` Request Kerberos V5 authentication with the user's initial password (entered when the user authenticated to the first authentication module in the stack). If Kerberos V5 authentication fails, or if no password has been entered, prompt the user for a password with the prompt "`Old KRB5 Password:`". If authentication succeeds, the user is prompted by "New KRB5 password:" for a new password. The user is then prompted a second time for the new password for verification and the KDC database is updated with the new password if both responses match.

`use_xfn_pass` Request Kerberos V5 authentication with a mapped password that has been stored under XFN. If Kerberos V5 authentication fails, or if no password has been stored, it quits and does not prompt the user for a password. If authentication succeeds, the user is prompted by "`New KRB5 password:`" for a new password. The user is then prompted a second time for the new password for verification and the KDC database is updated with the new password if both responses match.

`try_xfn_pass` Request Kerberos V5 authentication with a mapped password that has been stored under XFN. If Kerberos V5 authentication fails, or if no password has been stored, prompt the user for a password with the prompt "`Old KRB5 Password:`". If authentication succeeds, the user is prompted by "`New KRB5 password:`" for a new password. The user is then prompted a second time for the new password for verification and the KDC database is updated with the new password if both responses match.

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
MT Level	MT-Safe with exceptions

SEE ALSO

keylogin(1), pam(3PAM), pam_authenticate(3PAM), syslog(3C), libpam(4), pam.conf(4), attributes(5), SEAM(5)

NAME

pam_ldap – authentication and password management PAM modules for LDAP

SYNOPSIS

`/usr/lib/security/pam_ldap.so.1`

DESCRIPTION

The LDAP service module for PAM, `/usr/lib/security/pam_ldap.so.1`, provides functionality for two PAM modules: authentication and password management. The `pam_ldap.so.1` module is a shared object that can be dynamically loaded to provide the necessary functionality upon demand. Its path is specified in the PAM configuration file.

The `pam_ldap.so.1` module must be used in conjunction with pam_unix.so.1 module (see `pam_unix`(5)). The latter supports UNIX authentication and the `pam_ldap.so.1` module supports stronger authentication mechanisms such as CRAM-MD5. When this moudule is used for authentication and password management, it is designed to be stacked directly below the `pam_unix.so.1` module. If any other module was designed to be stacked in this manner, it can be stacked under the `pam_ldap.so.1` module. If this design is not followed, UNIX authentication or password management will not work. See the `EXAMPLES` section below to see how the modules are to be stacked when using this module.

The `pam_ldap.so.1` module supports two components: the Authentication component and the Password management component.

LDAP Authentication Component

The LDAP authentication component provides functions to verify the identity of a user (`pam_sm_authenticate`(3PAM)) and to set user specific credentials (`pam_sm_setcred`(3PAM)). The `pam_sm_authenticate( )` function uses the password entered by the user to attempt to authenticate to the LDAP server. If successful, the user is authenticated.

At present, the `pam_sm_setcred( )` function succeeds all the time without setting any credentials.

The following options may be passed to the LDAP service module:

debug
: `syslog`(3C) debugging information at `LOG_DEBUG` level.

nowarn
: Turn off warning messages.

use_first_pass
: Compare the password in the password database with the user’s initial password (entered when the user authenticated to the first authentication module in the stack). If the passwords do not match, or if no password has been entered, it quits and does not prompt the user for a password.

try_first_pass
: Compare the password in the password database with the user’s initial password (entered when the user authenticated to the first authentication module in the stack). If the passwords do not match, or if no password has been entered, prompt the user for a password.

These options are case sensitive and must be used exactly as presented here.

LDAP Password Management Component

The LDAP password management component provides the pam_sm_chauthtok(3PAM) function to change passwords in the LDAP password database. The following options may be passed in to the LDAP service module:

debug — syslog(3C) debugging information at LOG_DEBUG level.

nowarn — Turn off warning messages.

use_first_pass — Compare the password in the password database with the user's old password (entered to the first password module in the stack). If the passwords do not match, or if no password has been entered, it quits and does not prompt the user for the old password. It also attempts to use the new password (entered to the first password module in the stack) as the new password for this module. If the new password fails, it quits and does not prompt the user for a new password.

try_first_pass — Compare the password in the password database with the user's old password (entered to the first password module in the stack). If the passwords do not match, or if no password has been entered, it prompts the user for the old password. It also attempts to use the new password (entered to the first password module in the stack) as the new password for this module. If the new password fails, it prompts the user for a new password.

EXAMPLES

EXAMPLE 1 Use pam_ldap.so.1 with authentication in pam.conf.

The following is a configuration for the login service when using pam_ldap.so.1. The service name 'login' can be substituted for any other authentication service such as dtlogin or su. Lines that begin with the # symbol are comments and are ignored.

```
# Authentication management for login service is stacked.
# If pam_unix succeeds, pam_ldap is not invoked.
login   auth sufficient /usr/lib/security/pam_unix.so.1
login   auth required   /usr/lib/security/pam_ldap.so.1 try_first_pass
```

Note that the pam_unix.so.1 is qualified with the sufficient control flag.

EXAMPLE 2 Use pam_ldap.so.1 with password in pam.conf

The following is a configuration for the password management when using pam_ldap.so.1. Lines that begin with the # symbol are ignored.

```
# Password management
#
other   password sufficient     /usr/lib/security/pam_unix.so.1
other   password required       /usr/lib/security/pam_ldap.so.1
```

ATTRIBUTES

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

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

SEE ALSO

ldap(1), libpam(3LIB), pam(3PAM), pam_sm_authenticate(3PAM), pam_sm_chauthtok(3PAM), pam_sm_setcred(3PAM), syslog(3C), pam.conf(4), attributes(5)

NOTES

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

NAME

pam_rhosts_auth – authentication management PAM module using ruserok()

SYNOPSIS

`/usr/lib/security/pam_rhosts_auth.so.1`

DESCRIPTION

The rhosts PAM module, `/usr/lib/security/pam_rhosts_auth.so.1`, authenticates a user via the `rlogin` authentication protocol. Only `pam_sm_authenticate()` is implemented within this module. `pam_sm_authenticate()` uses the `ruserok`(3SOCKET) library function to authenticate the `rlogin` or `rsh` user. `pam_sm_setcred()` is a null function.

`/usr/lib/security/pam_rhosts_auth.so.1` is designed to be stacked on top of the `/usr/lib/security/pam_unix.so.1` module for both the `rlogin` and `rsh` services. This module is normally configured as *sufficient* so that subsequent authentication is performed only on failure of `pam_sm_authenticate()`. The following option may be passed in to this service module:

`debug` `syslog`(3C) debugging information at LOG_DEBUG level.

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
MT Level	MT-Safe with exceptions

SEE ALSO

`pam`(3PAM), `pam_authenticate`(3PAM), `ruserok`(3SOCKET), `syslog`(3C), `libpam`(3LIB), `pam.conf`(4), `attributes`(5)

NOTES

The interfaces in `libpam()` are MT-Safe only if each thread within the multi-threaded application uses its own PAM handle.

NAME

pam_roles – Role Account Management PAM module for Solaris

SYNOPSIS

/usr/lib/security/pam_roles.so.1

DESCRIPTION

The Role Account Management module for PAM, /usr/lib/security/pam_roles.so.1, provides functionality for one PAM module: Account management. The pam_roles.so.1 is a shared object that can be dynamically loaded to provide the necessary functionality upon demand. Its path is specified in the PAM configuration file.

Role Account Management module

The Role account management component provides a function to check for authorization to assume a role. It prevents direct logins to a role. It uses the user_attr(4) database to specify which users can assume which roles.

The following options may be passed to the Role Authentication service module:

debug syslog(3C) debugging information at LOG_DEBUG level.

If PAM_USER (see pam_set_item(3PAM)) is specified as type normal in the user_attr(4) database, the module returns PAM_IGNORE.

If PAM_RUSER (see pam_set_item(3PAM)) is not set, the uid of the process loading the module is used to determine PAM_RUSER.

The module returns success if the user_attr(4) entry for PAM_RUSER has an entry in the roles field for PAM_USER; otherwise it returns PAM_PERM_DENIED.

This module is generally stacked above the account management module pam_unix.so.1. The error messages indicating that roles cannot be logged into correctly are only issued if the user has entered the correct password.

Here are some sample entries from pam.conf(4) demonstrating the use of the pam_roles.so.1 module:

```
dtlogin account requisite /usr/lib/security/$ISA/pam_roles.so.1
dtlogin account required /usr/lib/security/$ISA/pam_unix.so.1
#
su account requisite /usr/lib/security/$ISA/pam_roles.so.1
su account requisite /usr/lib/security/$ISA/pam_roles.so.1
#
rlogin account requisite /usr/lib/security/$ISA/pam_roles.so.1
rlogin account required /usr/lib/security/$ISA/pam_unix.so.1
#
```

The dtlogin program invokes pam_roles.so.1. PAM_RUSER is the username corresponding to the uid of the dtlogin process, which is 0. The user_attr entry for root user (uid 0) is empty, so all role logins are prevented through dtlogin. The same rule applies to login.

The su program invokes pam_roles.so.1. PAM_RUSER is the username of the userid of the shell that invokes su. A user needs the appropriate entry in the roles list in user_attr(4) to be able to su to another user.

In the example above, the rlogin program invokes the pam_roles.so.1 module. The module checks for PAM_RUSER and determines whether the role being assumed, PAM_RUSER, is in the roles list of the userattr entry for PAM_RUSER. If it is in the roles list, the module returns PAM_SUCCESS; otherwise it returns PAM_PERM_DENIED.

SEE ALSO

keylogin(1), libpam(3LIB), pam(3PAM), pam_acct_mgmt(3PAM), pam_setcred(3PAM), pam_set_item(3PAM), syslog(3C), pam.conf(4), user_attr(4), attributes(5)

NOTES

The interfaces in libpam(3LIB) are MT-Safe only if each thread within the multi-threaded application uses its own PAM handle.

NAME

pam_sample – a sample PAM module

SYNOPSIS

`/usr/lib/security/pam_sample.so.1`

DESCRIPTION

The SAMPLE service module for PAM is divided into four components: authentication, account management, password management, and session management. The sample module is a shared object that is dynamically loaded to provide the necessary functionality.

SAMPLE Authentication Component

The SAMPLE authentication module, typically `/usr/lib/security/pam_sample.so.1`, provides functions to test the PAM framework functionality using the `pam_sm_authenticate`(3PAM) call. The SAMPLE module implementation of the `pam_sm_authenticate`(3PAM) function compares the user entered password with the password set in the `pam.conf`(4) file, or the string "test" if a default test password has not been set. The following options may be passed in to the SAMPLE Authentication module:

`debug` Syslog debugging information at the `LOG_DEBUG` level.

`passwd=newone` Sets the password to be "newone."

`first_pass_good` The first password is always good when used with the use_first_pass or try_first_pass option.

`first_pass_bad` The first password is always bad when used with the use_first_pass or try_first_pass option.

`always_fail` Always returns `PAM_AUTH_ERR.`

`always_succeed` Always returns `PAM_SUCCESS.`

`always_ignore` Always returns `PAM_IGNORE.`

`use_first_pass` Use the user's initial password (entered when the user is authenticated to the first authentication module in the stack) to authenticate with the SAMPLE module. If the passwords do not match, or if this is the first authentication module in the stack, quit and do not prompt the user for a password. It is recommended that this option only be used if the SAMPLE authentication module is designated as *optional* in the `pam.conf` configuration file.

`try_first_pass` Use the user's initial password (entered when the user is authenticated to the first authentication module in the stack) to authenticate with the SAMPLE module. If the passwords do not match, or if this is the first authentication module in the stack, prompt the user for a password. The SAMPLE module `pam_sm_setcred`(3PAM) function always returns `PAM_SUCCESS.`

SAMPLE Account Management Component

The SAMPLE Account Management Component, typically `pam_sample.so.1`, implements a simple access control scheme that limits machine access to a list of authorized users. The list of authorized users is supplied as option arguments to the entry for the SAMPLE account management PAM module in the `pam.conf` file. Note that the module always permits access to the root super user.

The option field syntax to limit access is shown below: allow= *name[,name]* allow= *name [allow=name]*

The example `pam.conf` show below permits only larry to `login` directly. `rlogin` is allowed only for don and larry. Once a user is logged in, the user can use `su` if the user are sam or eric.

login	account	require	pam_sample.so.1 allow=larry
dtlogin	account	require	pam_sample.so.1 allow=larry
rlogin	account	require	pam_sample.so.1 allow=don allow=larry
su	account	require	pam_sample.so.1 allow=sam,eric

The debug and nowarn options are also supported.

SAMPLE Password Management Component

The SAMPLE Password Management Component function (`pam_sm_chauthtok`(3PAM)), always returns `PAM_SUCCESS.`

SAMPLE Session Management Component

The SAMPLE Session Management Component functions (`pam_sm_open_session`(3PAM), `pam_sm_close_session`(3PAM)) always return `PAM_SUCCESS.`

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
MT Level	MT-Safe with exceptions

SEE ALSO

`pam`(3PAM), `pam_sm_authenticate`(3PAM), `pam_sm_chauthtok`(3PAM), `pam_sm_close_session`(3PAM), `pam_sm_open_session`(3PAM), `pam_sm_setcred`(3PAM), `libpam`(3LIB), `pam.conf`(4), `attributes`(5)

NOTES

The interfaces in `libpam()` are MT-Safe only if each thread within the multi-threaded application uses its own PAM handle.

NAME

pam_smartcard – PAM authentication module for Smart Card

SYNOPSIS

/usr/lib/security/pam_smartcard.so

DESCRIPTION

The Smart Card service module for PAM, /usr/lib/security/pam_smartcard.so, provides functionality for PAM smart card authentication. The pam_smartcard.so module is a shared object that can be dynamically loaded to provide the necessary functionality upon demand. Its path is specified in the PAM configuration file pam.conf.

Smart Card Authentication Module

The Smart Card authentication component provides functions to verify the identity of a smart card user, pam_sm_authenticate(3PAM).

The pam_sm_authenticate( ) function collects the user input such as user name, PIN number, password and related authentication tokens. It passes this data back to its underlying layer, OCF to perform card based authentication except password. The password is passed from the smart card module to a user-specified PAM module. This PAM module compares the password either entered by the user or downloaded from the card with the password that this module associates with the user. If all the authentication steps are successful, the user is authenticated and this module returns PAM_SUCCESS.

The following options may be passed to the Smart Card service module:

debug	sysolg(3C) debugging information at LOG_DEBUG level.
nowarn	Turn off warning messages.
verbose	Turn on verbose authentication failure reporting to the user.
password=*xxx*	Specify name of PAM client to use when password authentication is required. This option is used to specify the client name to use when a password authentication is required. This is the name that will be used in a call to pam_start(3PAM) from inside the pam_smartcard module. The appropriate entries in /etc/pam.conf must exist to use this facility. The default value of this option is smartcard_unix.

EXAMPLES

EXAMPLE 1 pam.conf entries

The following pam.conf entries illustrate the use of the password option:

```
service        type  ctrl-flag Module_path         Options

dtlogin        auth  required  /usr/lib/security/  password=smartcard_unix
                                pam_smartcard.so.1
smartcard_unix auth  required  /usr/lib/security/  use_first_pass
                                pam_unix.so.1
```

This example shows that when `pam_smartcard` needs to perform a password authentication, it will register with PAM as the client name smartcard_unix, and PAM in turn will use the `pam_unix`(5) module to perform the password authentication, returning the results to pam_smartcard

This module provides the ability to specify the number of retries that the user is allowed when entering an invalid password or invalid PIN. To specify the retry values, edit the `/etc/default/login` file and add or modify the appropriate parameter. See `FILES`.

FILES

`/etc/default/login`

Values can be set for the following retry parameters in the file /etc/default/login. If a parameter is not specified, the default value is used.

SCPW_RETRIES — Sets the number of invalid password retries allowed. The default is 0.

SCPIN_RETRIES — Sets the number of invalid PIN retries allowed. The default is 0.

SEE ALSO

`smartcard`(1M), `libpam`(3LIB), `pam`(3PAM), `pam_authenticate`(3PAM), `pam_start`(3PAM), `pam.conf`(4)

NOTES

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

NAME

pam_unix – authentication, account, session, and password management PAM modules for UNIX

SYNOPSIS

`/usr/lib/security/pam_unix.so.1`

DESCRIPTION

The UNIX service module for PAM, `/usr/lib/security/pam_unix.so.1`, provides functionality for all four PAM modules: authentication, account management, session management and password management. The `pam_unix.so.1` module is a shared object that can be dynamically loaded to provide the necessary functionality upon demand. Its path is specified in the PAM configuration file.

Unix Authentication Module

The UNIX authentication component provides functions to verify the identity of a user, (`pam_sm_authenticate( )`) and to set user specific credentials (`pam_sm_setcred( )`). `pam_sm_authenticate( )` compares the user entered password with the password from the UNIX password database. If the passwords match, the user is authenticated. If the user also has secure RPC credentials and the secure RPC password is the same as the UNIX password, then the secure RPC credentials are also obtained.

The following options may be passed to the UNIX service module:

`debug` — `syslog`(3C) debugging information at `LOG_DEBUG` level.

`nowarn` — Turn off warning messages.

`use_first_pass` — It compares the password in the password database with the user's initial password (entered when the user authenticated to the first authentication module in the stack). If the passwords do not match, or if no password has been entered, it quits and does not prompt the user for a password. This option should only be used if the authentication service is designated as *optional* in the `pam.conf` configuration file.

`try_first_pass` — It compares the password in the password database with the user's initial password (entered when the user authenticated to the first authentication module in the stack). If the passwords do not match, or if no password has been entered, prompt the user for a password. When prompting for the current password, the UNIX authentication module will use the prompt, "password:" unless one of the following scenarios occur:

1. The option `try_first_pass` is specified and the password entered for the first module in the stack fails for the UNIX module.
2. The option `try_first_pass` is not specified, and the earlier authentication modules listed in the `pam.conf` file have prompted the user for the password.

In these two cases, the UNIX authentication module will use the prompt "SYSTEM password:". The `pam_sm_setcred()` function sets user specific credentials. If the user had secure RPC credentials, but the secure RPC password was not the same as the UNIX password, then a warning message is printed. If the user wants to get secure RPC credentials, then `keylogin`(1) needs to be run.

Unix Account Management Module

The UNIX account management component provides a function to perform account management, `pam_sm_acct_mgmt()`. The function retrieves the user's password entry from the UNIX password database and verifies that the user's account and password have not expired. The following options may be passed in to the UNIX service module:

`debug` `syslog`(3C) debugging information at `LOG_DEBUG` level.

`nowarn` Turn off warning messages.

Unix Session Management Module

The UNIX session management component provides functions to initiate `pam_sm_open_session()` and terminate `pam_sm_close_session()` UNIX sessions. For UNIX, `pam_open_session` updates the `/var/adm/lastlog` file. The account management module reads this file to determine the previous time the user logged in. The following options may be passed in to the UNIX service module:

`debug` `syslog`(3C) debugging information at `LOG_DEBUG` level.

`nowarn` Turn off warning messages. `pam_close_session` is a null function.

Unix Password Management Module

The UNIX password management component provides a function to change passwords `pam_sm_chauthtok()` in the UNIX password database. This module must be *required* in `pam.conf`. It cannot be *optional* or *sufficient*. The following options may be passed in to the UNIX service module:

`debug` `syslog`(3C) Debugging information at `LOG_DEBUG` level.

`nowarn` Turn off warning messages.

`use_first_pass` It compares the password in the password database with the user’s old password (entered to the first password module in the stack). If the passwords do not match, or if no password has been entered, it quits and does not prompt the user for the old password. It also attempts to use the new password (entered to the first password module in the stack) as the new password for this module. If the new password fails, it quits and does not prompt the user for a new password.

`try_first_pass` It compares the password in the password database with the user’s old password (entered to the first password module in the stack). If the passwords do not match, or if no password has been entered, it prompts the user for the old password. It also attempts to use the new password (entered to the first password module in the stack) as the new password for this module. If the new password fails, it prompts the user for a new password. If the user’s password has expired, the UNIX account module saves this information in the authentication handle using `pam_set_data()`, with a unique name, `SUNW_UNIX_AUTHOK_DATA.` The UNIX password module retrieves this information from the authentication handle using `pam_get_data()` to determine whether or not to force the user to update the user’s password.

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
MT Level	MT-Safe with exceptions

SEE ALSO

`keylogin`(1), `pam`(3PAM), `pam_authenticate`(3PAM), `pam_setcred`(3PAM), `syslog`(3C), `libpam`(3LIB), `pam.conf`(4), `attributes`(5)

NOTES

The interfaces in `libpam()` are MT-Safe only if each thread within the multi-threaded application uses its own PAM handle.

NAME

prof – profile within a function

SYNOPSIS

#define MARK
#include <prof.h>
void **MARK**(*name*);

DESCRIPTION

MARK introduces a mark called *name* that is treated the same as a function entry point. Execution of the mark adds to a counter for that mark, and program-counter time spent is accounted to the immediately preceding mark or to the function if there are no preceding marks within the active function.

name may be any combination of letters, numbers, or underscores. Each *name* in a single compilation must be unique, but may be the same as any ordinary program symbol.

For marks to be effective, the symbol MARK must be defined before the header `prof.h` is included, either by a preprocessor directive as in the synopsis, or by a command line argument:

```
cc -p -DMARK work.c
```

If MARK is not defined, the MARK(*name*) statements may be left in the source files containing them and are ignored. `prof -g` must be used to get information on all labels.

EXAMPLES

In this example, marks can be used to determine how much time is spent in each loop. Unless this example is compiled with MARK defined on the command line, the marks are ignored.

```
#include <prof.h>
work( )
{
        int i, j;
        . . .
        MARK(loop1);
        for (i = 0; i < 2000; i++) {
                . . .
        }
        MARK(loop2);
        for (j = 0; j < 2000; j++) {
                . . .
        }
}
```

SEE ALSO

`profil`(2), `monitor`(3C)

NAME

rbac – role-based access control

DESCRIPTION

The addition of role-based access control (RBAC) to the Solaris operating environment gives developers the opportunity to deliver fine-grained security in new and modified applications. RBAC is an alternative to the all-or-nothing security model of traditional superuser-based systems. With RBAC, an administrator can assign privileged functions to specific user accounts (or special accounts called roles).

There are two ways of giving applications privileges:

1. Administrators can assign special attributes such as setUID to applications.
2. Administrators can assign special attributes such as setUID to applications.

Special attribute assignment along with the theory behind RBAC is discussed in detail in “Role Based Access Control” chapter of the System Administration Guide, Volume 2. This section describes what authorizations are and how to code for them.

Authorizations

An authorization is a unique string that represents a user’s right to perform some operation or class of operations. Authorization definitions are stored in a database called auth_attr(4). For programming authorization checks, only the authorization name is significant.

Some typical values in an auth_attr database are shown below.

```
solaris.jobs.:::Cron and At Jobs::help=JobHeader.html
solaris.admin:::Cron & At Administrator::help=JobsAdmin.html
solaris.grant:::Delegate Cron & At Administration::help=JobsGrant.html
solaris.jobs.user:::Cron & At User::help=JobsUser.html
```

Note the authorization name strings ending with the grant suffix. These are special authorizations; they give a user the ability to delegate authorizations with the same prefix and functional area to other users.

Creating Authorization Checks

To check authorizations, use the `chkauthattr`(3SECDB) library function, which verifies whether or not a user has a given authorization. The synopsis is:

```
int chkauthattr(const char *authname, const char *username);
```

The `chkauthattr()` function checks the policy.conf(4), user_attr(4), and prof_attr(4) databases in order for a match to the given authorization.

If you are modifying existing code that tests for root UID, you should find the test in the code and replace it with the chkauthattr function. A typical root UID check is shown in Figure 1, with the test underlined. An authorization check replacing it is shown in Figure 2; it uses the solaris.jobs.admin authorization and a variable called real_login representing the user.

EXAMPLE 1 Standard root check

```
ruid = getuid();

if ((eflag || lflag || rflag) && argc == 1) {
        if ((pwp = getpwnam(*argv)) == NULL)
                crabort(INVALIDUSER);

        if (ruid != 0) {
                if (pwp->pw_uid != ruid)
                        crabort(NOTROOT);
                else
                        pp = getuser(ruid);
        } else
                pp = *argv++;
} else {
```

EXAMPLE 2 Authorization check

```
ruid = getuid();
if ((pwp = getpwuid(ruid)) == NULL)
        crabort(INVALIDUSER);

strcpy(real_login, pwp->pw_name);

if ((eflag || lflag || rflag) && argc == 1) {
        if ((pwp = getpwnam(*argv)) == NULL)
                crabort(INVALIDUSER);

        if (!chkauthattr("solaris.jobs.admin", real_login)) {
                if (pwp->pw_uid != ruid)
                        crabort(NOTROOT);
                else
                        pp = getuser(ruid);
        } else
                pp = *argv++;
} else {
```

For new applications, find an appropriate location for the test and use `chkauthattr()` as shown above. Typically the authorization check makes an access decision based on the identity of the calling user to determine if a privileged action (for example, a system call) should be taken on behalf of that user.

Applications that perform a test to restrict who can perform their security-relevant functionality are generally `setuid` to root. Programs that were written prior to RBAC and that are only available to the root user may not have such checks. Remember that in most cases, the kernel requires an effective user ID of root in order to override policy enforcement. Therefore, authorization checking is most useful in programs that are setuid to root.

For instance, if you want to write a program that allows authorized users to set the system date, the command must be run with an effective user ID of root. Typically, this means that the file modes for the file would be `-rwsr-xr-x` with root ownership.

Use caution, though, when making programs setuid to root. For example, the effective UID should be set to the real UID as early as possible in the program's initialization function. The effective UID can then be set back to root after the authorization check is performed and before the system call is made. On return from the system call, the effective UID should be set back to the real UID again to adhere to the principle of least privilege.

Another consideration is that LD_LIBRARY path is ignored for setuid programs (see SECURITY section in `ld.so.1`(1)) and that shell scripts must be modified to work properly when the effective and real UIDs are different. For example, the –p flag in Bourne shell is required to avoid resetting the effective UID back to the real UID.

Using an effective UID of root instead of the real UID requires extra care when writing shell scripts. For example, many shell scripts check to see if the user is root before executing their functionality. With RBAC, these shell scripts may be running with the effective UID of root and with a real UID of a user or role. Thus, the shell script should check euid instead of uid. For example,

```
WHO=‘id | cut -f1 -d" "‘
if [ ! "$WHO" = "uid=0(root)" ]
then
       echo "$PROG: ERROR: you must be super-user to run this script."
       exit 1
fi
```

should be changed to

```
WHO=‘/usr/xpg4/bin/id -n -u‘
if [ ! "$WHO" = "root" ]
then
       echo "$PROG: ERROR: you are not authorized to run this script."
       exit 1
fi
```

Authorizations can be explicitly checked in shell scripts by piping the output of the `auths`(1) utility to `grep`(1). For example,

```
AUTHS=‘auths‘
echo $AUTHS|grep "^solaris.date$"
if [ $? -ne 0 ]
then
        echo "$PROG: ERROR: you are not authorized to set the date."
        exit 1
fi
```

SEE ALSO

`ld.so.1`(1), `chkauthattr`(3SECDB), `policy.conf`(4), `prof_attr`(4), `user_attr`(4)

NAME

regex – internationalized basic and extended regular expression matching

DESCRIPTION

Regular Expressions (REs) provide a mechanism to select specific strings from a set of character strings. The Internationalized Regular Expressions described below differ from the Simple Regular Expressions described on the `regexp`(5) manual page in the following ways:

- both Basic and Extended Regular Expressions are supported
- the Internationalization features—character class, equivalence class, and multi-character collation—are supported.

The Basic Regular Expression (BRE) notation and construction rules described in the `BASIC REGULAR EXPRESSIONS` section apply to most utilities supporting regular expressions. Some utilities, instead, support the Extended Regular Expressions (ERE) described in the `EXTENDED REGULAR EXPRESSIONS` section; any exceptions for both cases are noted in the descriptions of the specific utilities using regular expressions. Both BREs and EREs are supported by the Regular Expression Matching interfaces `regcomp`(3C) and `regexec`(3C).

BASIC REGULAR EXPRESSIONS

BREs Matching a Single Character

A BRE ordinary character, a special character preceded by a backslash, or a period matches a single character. A bracket expression matches a single character or a single collating element. See `RE Bracket Expression`, below.

BRE Ordinary Characters

An ordinary character is a BRE that matches itself: any character in the supported character set, except for the BRE special characters listed in `BRE Special Characters`, below.

The interpretation of an ordinary character preceded by a backslash (\) is undefined, except for:

1. the characters), (, {, and }
2. the digits 1 to 9 inclusive (see `BREs Matching Multiple Characters`, below)
3. a character inside a bracket expression.

BRE Special Characters

A BRE *special character* has special properties in certain contexts. Outside those contexts, or when preceded by a backslash, such a character will be a BRE that matches the special character itself. The BRE special characters and the contexts in which they have their special meaning are:

. [\ The period, left-bracket, and backslash are special except when used in a bracket expression (see `RE Bracket Expression`, below). An expression containing a [that is not preceded by a backslash and is not part of a bracket expression produces undefined results.

* The asterisk is special except when used:

- in a bracket expression
- as the first character of an entire BRE (after an initial ^, if any)
- as the first character of a subexpression (after an initial ^, if any); see `BREs Matching Multiple Characters`, below.

^ The circumflex is special when used:

- as an anchor (see `BRE Expression Anchoring`, below).
- as the first character of a bracket expression (see `RE Bracket Expression`, below).

$ The dollar sign is special when used as an anchor.

Periods in BREs

A period (`.`), when used outside a bracket expression, is a BRE that matches any character in the supported character set except NUL.

RE Bracket Expression

A bracket expression (an expression enclosed in square brackets, [ ]) is an RE that matches a single collating element contained in the non-empty set of collating elements represented by the bracket expression.

The following rules and definitions apply to bracket expressions:

1. A *bracket expression* is either a matching list expression or a non-matching list expression. It consists of one or more expressions: collating elements, collating symbols, equivalence classes, character classes, or range expressions (see rule 7 below). Portable applications must not use range expressions, even though all implementations support them. The right-bracket (]) loses its special meaning and represents itself in a bracket expression if it occurs first in the list (after an initial circumflex (^), if any). Otherwise, it terminates the bracket expression, unless it appears in a collating symbol (such as [.].]) or is the ending right-bracket for a collating symbol, equivalence class, or character class. The special characters:

   `.   *   [   \`

   (period, asterisk, left-bracket and backslash, respectively) lose their special meaning within a bracket expression.

   The character sequences:

   `[.   [=   [:`

   (left-bracket followed by a period, equals-sign, or colon) are special inside a bracket expression and are used to delimit collating symbols, equivalence class expressions, and character class expressions. These symbols must be followed by a valid expression and the matching terminating sequence .], =] or :], as described in the following items.

2. A *matching list* expression specifies a list that matches any one of the expressions represented in the list. The first character in the list must not be the circumflex. For example, `[abc]` is an RE that matches any of the characters `a`, `b` or `c`.
3. A *non-matching list* expression begins with a circumflex (^), and specifies a list that matches any character or collating element except for the expressions represented in the list after the leading circumflex. For example, [^abc] is an RE that matches any character or collating element except the characters `a, b`, or `c`. The circumflex will have this special meaning only when it occurs first in the list, immediately following the left-bracket.
4. A *collating symbol* is a collating element enclosed within bracket-period ([..]) delimiters. Multi-character collating elements must be represented as collating symbols when it is necessary to distinguish them from a list of the individual characters that make up the multi-character collating element. For example, if the string `ch` is a collating element in the current collation sequence with the associated collating symbol <ch>, the expression [[.ch.]] will be treated as an RE matching the character sequence `ch`, while [ch] will be treated as an RE matching `c` or `h`. Collating symbols will be recognized only inside bracket expressions. This implies that the RE `[[.ch.]]*c` matches the first to fifth character in the string chchch. If the string is not a collating element in the current collating sequence definition, or if the collating element has no characters associated with it, the symbol will be treated as an invalid expression.
5. An *equivalence class expression* represents the set of collating elements belonging to an equivalence class. Only primary equivalence classes will be recognised. The class is expressed by enclosing any one of the collating elements in the equivalence class within bracket-equal ([==]) delimiters. For example, if a, and belong to the same equivalence class, then [[=a=]b], [[==]b] and [[==]b] will each be equivalent to [ab]. If the collating element does not belong to an equivalence class, the equivalence class expression will be treated as a *collating symbol*.
6. A *character class expression* represents the set of characters belonging to a character class, as defined in the `LC_CTYPE` category in the current locale. All character classes specified in the current locale will be recognized. A character class expression is expressed as a character class name enclosed within bracket-colon ([::]) delimiters.

   The following character class expressions are supported in all locales:

`[:alnum:]`	`[:cntrl:]`	`[:lower:]`	`[:space:]`
`[:alpha:]`	`[:digit:]`	`[:print:]`	`[:upper:]`
`[:blank:]`	`[:graph:]`	`[:punct:]`	`[:xdigit:]`

In addition, character class expressions of the form:

`[:name:]`

are recognized in those locales where the *name* keyword has been given a `charclass` definition in the `LC_CTYPE` category.

7. A *range expression* represents the set of collating elements that fall between two elements in the current collation sequence, inclusively. It is expressed as the starting point and the ending point separated by a hyphen (`-`).

   Range expressions must not be used in portable applications because their behavior is dependent on the collating sequence. Ranges will be treated according to the current collating sequence, and include such characters that fall within the range based on that collating sequence, regardless of character values. This, however, means that the interpretation will differ depending on collating sequence. If, for instance, one collating sequence defines as a variant of `a`, while another defines it as a letter following `z`, then the expression [–z] is valid in the first language and invalid in the second.

   In the following, all examples assume the collation sequence specified for the POSIX locale, unless another collation sequence is specifically defined.

   The starting range point and the ending range point must be a collating element or collating symbol. An equivalence class expression used as a starting or ending point of a range expression produces unspecified results. An equivalence class can be used portably within a bracket expression, but only outside the range. For example, the unspecified expression [[=e=]–f] should be given as [[=e=]e–f]. The ending range point must collate equal to or higher than the starting range point; otherwise, the expression will be treated as invalid. The order used is the order in which the collating elements are specified in the current collation definition. One-to-many mappings (see `locale`(5)) will not be performed. For example, assuming that the character eszet is placed in the collation sequence after `r` and `s`, but before `t`, and that it maps to the sequence `ss` for collation purposes, then the expression [r–s] matches only `r` and `s`, but the expression [s–t] matches `s`, `beta`, or `t`.

   The interpretation of range expressions where the ending range point is also the starting range point of a subsequent range expression (for instance [a–m–o]) is undefined.

   The hyphen character will be treated as itself if it occurs first (after an initial ^, if any) or last in the list, or as an ending range point in a range expression. As examples, the expressions [–ac] and [ac–] are equivalent and match any of the characters `a`, `c`, or `–;` [^–ac] and [^ac–] are equivalent and match any characters except `a`, `c`, or –; the expression [%– –] matches any of the characters between % and – inclusive; the expression [– –@] matches any of the characters between – and @ inclusive; and the expression [a– –@]

is invalid, because the letter a follows the symbol – in the POSIX locale. To use a hyphen as the starting range point, it must either come first in the bracket expression or be specified as a collating symbol, for example: [][.–.]–0], which matches either a right bracket or any character or collating element that collates between hyphen and 0, inclusive.

If a bracket expression must specify both – and], the] must be placed first (after the ^, if any) and the – last within the bracket expression.

Note: Latin-1 characters such as ` or ˆ are not printable in some locales, for example, the ja locale.

BREs Matching Multiple Characters

The following rules can be used to construct BREs matching multiple characters from BREs matching a single character:

1. The concatenation of BREs matches the concatenation of the strings matched by each component of the BRE.
2. A *subexpression* can be defined within a BRE by enclosing it between the character pairs \(and \) . Such a subexpression matches whatever it would have matched without the \(and \), except that anchoring within subexpressions is optional behavior; see BRE Expression Anchoring, below. Subexpressions can be arbitrarily nested.
3. The *back-reference* expression *n* matches the same (possibly empty) string of characters as was matched by a subexpression enclosed between \(and \) preceding the *n*. The character *n* must be a digit from 1 to 9 inclusive, *n*th subexpression (the one that begins with the *n*th \(and ends with the corresponding paired \)). The expression is invalid if less than *n* subexpressions precede the *n*. For example, the expression ^\(.*\)\1$ matches a line consisting of two adjacent appearances of the same string, and the expression \(a\)*\1 fails to match a. The limit of nine back-references to subexpressions in the RE is based on the use of a single digit identifier. This does not imply that only nine subexpressions are allowed in REs. The following is a valid BRE with ten subexpressions:

   ```
   \(\(\(ab\)*c\)*d\)\(ef\)*\(gh\)\{2\}\(ij\)*\(kl\)*\(mn\)*\(op\)*\(qr\)*
   ```
4. When a BRE matching a single character, a subexpression or a back-reference is followed by the special character asterisk (*), together with that asterisk it matches what zero or more consecutive occurrences of the BRE would match. For example, [ab]* and [ab][ab] are equivalent when matching the string ab.
5. When a BRE matching a single character, a subexpression, or a back-reference is followed by an *interval expression* of the format \{*m*\}, \{*m*,\} or \{*m*,*n*\}, together with that interval expression it matches what repeated consecutive occurrences of the BRE would match. The values of

m and *n* will be decimal integers in the range 0 ≤ *m* ≤ *n* ≤ {RE_DUP_MAX}, where *m* specifies the exact or minimum number of occurrences and *n* specifies the maximum number of occurrences. The expression \{*m*\} matches exactly *m* occurrences of the preceding BRE, \{*m*,\} matches at least *m* occurrences and \{*m*,*n*\} matches any number of occurrences between *m* and *n*, inclusive.

For example, in the string `abababccccccd`, the BRE c\{3\} is matched by characters seven to nine, the BRE \(ab\)\{4,\} is not matched at all and the BRE c\{1,3\}d is matched by characters ten to thirteen.

The behavior of multiple adjacent duplication symbols (`*` and intervals) produces undefined results.

BRE Precedence

The order of precedence is as shown in the following table:

BRE Precedence (from high to low)	
collation-related bracket symbols	[= =] [: :] [. .]
escaped characters	\\<*special character*>
bracket expression	[ ]
subexpressions / back-references	\(\) *n*
single-character-BRE duplication	* \{*m*,*n*\}
concatenation	
anchoring	^ $

BRE Expression Anchoring

A BRE can be limited to matching strings that begin or end a line; this is called *anchoring*. The circumflex and dollar sign special characters will be considered BRE anchors in the following contexts:

1. A circumflex (`^`) is an anchor when used as the first character of an entire BRE. The implementation may treat circumflex as an anchor when used as the first character of a subexpression. The circumflex will anchor the expression to the beginning of a string; only sequences starting at the first character of a string will be matched by the BRE. For example, the BRE ^ab matches `ab` in the string `abcdef,` but fails to match in the string `cdefab`. A portable BRE must escape a leading circumflex in a subexpression to match a literal circumflex.
2. A dollar sign (`$`) is an anchor when used as the last character of an entire BRE. The implementation may treat a dollar sign as an anchor when used

as the last character of a subexpression. The dollar sign will anchor the expression to the end of the string being matched; the dollar sign can be said to match the end-of-string following the last character.

3. A BRE anchored by both `^` and `$` matches only an entire string. For example, the BRE `^abcdef$` matches strings consisting only of `abcdef`.
4. `^` and `$` are not special in subexpressions.

Note: The Solaris implementation does not support anchoring in BRE subexpressions.

EXTENDED REGULAR EXPRESSIONS

The rules specifed for BREs apply to Extended Regular Expressions (EREs) with the following exceptions:

- The characters `|`, `+`, and `?` have special meaning, as defined below.
- The `{` and `}` characters, when used as the duplication operator, are not preceded by backslashes. The constructs `\{` and `\}` simply match the characters `{` and `}`, respectively.
- The back reference operator is not supported.
- Anchoring (`^$`) is supported in subexpressions.

EREs Matching a Single Character

An ERE ordinary character, a special character preceded by a backslash, or a period matches a single character. A bracket expression matches a single character or a single collating element. An *ERE matching a single character* enclosed in parentheses matches the same as the ERE without parentheses would have matched.

ERE Ordinary Characters

An *ordinary character* is an ERE that matches itself. An ordinary character is any character in the supported character set, except for the ERE special characters listed in `ERE Special Characters` below. The interpretation of an ordinary character preceded by a backslash (\) is undefined.

ERE Special Characters

An *ERE special character* has special properties in certain contexts. Outside those contexts, or when preceded by a backslash, such a character is an ERE that matches the special character itself. The extended regular expression special characters and the contexts in which they have their special meaning are:

. [\ (The period, left-bracket, backslash, and left-parenthesis are special except when used in a bracket expression (see `RE Bracket Expression`, above). Outside a bracket expression, a left-parenthesis immediately followed by a right-parenthesis produces undefined results.

) The right-parenthesis is special when matched with a preceding left-parenthesis, both outside a bracket expression.

* + ? { The asterisk, plus-sign, question-mark, and left-brace are special except when used in a bracket expression (see `RE`

`Bracket Expression`, above). Any of the following uses produce undefined results:

- if these characters appear first in an ERE, or immediately following a vertical-line, circumflex or left-parenthesis
- if a left-brace is not part of a valid interval expression.

| The vertical-line is special except when used in a bracket expression (see `RE Bracket Expression`, above). A vertical-line appearing first or last in an ERE, or immediately following a vertical-line or a left-parenthesis, or immediately preceding a right-parenthesis, produces undefined results.

^ The circumflex is special when used:

- as an anchor (see `ERE Expression Anchoring`, below).
- as the first character of a bracket expression (see `RE Bracket Expression`, above).

$ The dollar sign is special when used as an anchor.

Periods in EREs

A period (`.`), when used outside a bracket expression, is an ERE that matches any character in the supported character set except NUL.

ERE Bracket Expression

The rules for ERE Bracket Expressions are the same as for Basic Regular Expressions; see `RE Bracket Expression`, above).

EREs Matching Multiple Characters

The following rules will be used to construct EREs matching multiple characters from EREs matching a single character:

1. A *concatenation of EREs* matches the concatenation of the character sequences matched by each component of the ERE. A concatenation of EREs enclosed in parentheses matches whatever the concatenation without the parentheses matches. For example, both the ERE `cd` and the ERE `(cd)` are matched by the third and fourth character of the string `abcdefabcdef`.
2. When an ERE matching a single character or an ERE enclosed in parentheses is followed by the special character plus-sign (+), together with that plus-sign it matches what one or more consecutive occurrences of the ERE would match. For example, the ERE `b+(bc)` matches the fourth to seventh characters in the string `acabbbcde; [ab]` + and `[ab][ab]*` are equivalent.
3. When an ERE matching a single character or an ERE enclosed in parentheses is followed by the special character asterisk (`*`), together with that asterisk it matches what zero or more consecutive occurrences of the ERE would match. For example, the ERE `b*c` matches the first character in the string

cabbbcde, and the ERE b*cd matches the third to seventh characters in the string cabbbcdebbbbbbcdbc. And, [ab]* and [ab][ab] are equivalent when matching the string ab.

4. When an ERE matching a single character or an ERE enclosed in parentheses is followed by the special character question-mark (?), together with that question-mark it matches what zero or one consecutive occurrences of the ERE would match. For example, the ERE b?c matches the second character in the string acabbbcde.
5. When an ERE matching a single character or an ERE enclosed in parentheses is followed by an *interval expression* of the format {*m*}, {*m*,} or {*m*,*n*}, together with that interval expression it matches what repeated consecutive occurrences of the ERE would match. The values of *m* and *n* will be decimal integers in the range $0 \le m \le n \le$ {RE_DUP_MAX}, where *m* specifies the exact or minimum number of occurrences and *n* specifies the maximum number of occurrences. The expression {*m*} matches exactly *m* occurrences of the preceding ERE, {*m*,} matches at least *m* occurrences and {*m*,*n*} matches any number of occurrences between *m* and *n*, inclusive.

For example, in the string abababcccccd the ERE c{3} is matched by characters seven to nine and the ERE (ab){2,} is matched by characters one to six.

The behavior of multiple adjacent duplication symbols (+, *, ? and intervals) produces undefined results.

ERE Alternation

Two EREs separated by the special character vertical-line (|) match a string that is matched by either. For example, the ERE a((bc)|d) matches the string abc and the string ad. Single characters, or expressions matching single characters, separated by the vertical bar and enclosed in parentheses, will be treated as an ERE matching a single character.

ERE Precedence

The order of precedence will be as shown in the following table:

ERE Precedence (from high to low)	
collation-related bracket symbols	[= =] [: :] [. .]
escaped characters	\\<*special character*>
bracket expression	[ ]
grouping	( )
single-character-ERE duplication	* + ? {*m*,*n*}
concatenation	

anchoring	^ $
alternation	|

For example, the ERE `abba | cde` matches either the string `abba` or the string `cde` (rather than the string `abbade` or `abbcde`, because concatenation has a higher order of precedence than alternation).

ERE Expression Anchoring

An ERE can be limited to matching strings that begin or end a line; this is called *anchoring*. The circumflex and dollar sign special characters are considered ERE anchors when used anywhere outside a bracket expression. This has the following effects:

1. A circumflex (^) outside a bracket expression anchors the expression or subexpression it begins to the beginning of a string; such an expression or subexpression can match only a sequence starting at the first character of a string. For example, the EREs ^ab and (^ab) match ab in the string abcdef, but fail to match in the string cdefab, and the ERE a^b is valid, but can never match because the `a` prevents the expression ^b from matching starting at the first character.
2. A dollar sign `( $ )` outside a bracket expression anchors the expression or subexpression it ends to the end of a string; such an expression or subexpression can match only a sequence ending at the last character of a string. For example, the EREs ef$ and (ef$) match ef in the string abcdef, but fail to match in the string cdefab, and the ERE e$f is valid, but can never match because the `f` prevents the expression e$ from matching ending at the last character.

SEE ALSO

`localedef`(1), `regcomp`(3C), `attributes`(5), `environ`(5), `locale`(5), `regexp`(5)

NAME

regexp, compile, step, advance – simple regular expression compile and match routines

SYNOPSIS

#define INIT *declarations*
#define GETC(void) *getc code*
#define PEEKC(void) *peekc code*
#define UNGETC(void) *ungetc code*
#define RETURN(*ptr*) *return code*
#define ERROR(*val*) *error code*

extern char **loc1*, **loc2*, **locs*;

#include <regexp.h>
char ***`compile`**(char **instring*, char **expbuf*, const char **endfug*, int *eof*);

int **`step`**(const char **string*, const char **expbuf*);

int **`advance`**(const char **string*, const char **expbuf*);

DESCRIPTION

Regular Expressions (REs) provide a mechanism to select specific strings from a set of character strings. The Simple Regular Expressions described below differ from the Internationalized Regular Expressions described on the `regex`(5) manual page in the following ways:

- only Basic Regular Expressions are supported
- the Internationalization features–character class, equivalence class, and multi-character collation–are not supported.

The functions `step()` , `advance()` , and `compile()` are general purpose regular expression matching routines to be used in programs that perform regular expression matching. These functions are defined by the `<regexp.h>` header.

The functions `step()` and `advance()` do pattern matching given a character string and a compiled regular expression as input.

The function `compile()` takes as input a regular expression as defined below and produces a compiled expression that can be used with `step()` or `advance()` .

Basic Regular Expressions

A regular expression specifies a set of character strings. A member of this set of strings is said to be matched by the regular expression. Some characters have special meaning when used in a regular expression; other characters stand for themselves.

The following *one-character RE* s match a *single* character:

1.1 An ordinary character (*not* one of those discussed in 1.2 below) is a one-character RE that matches itself.

1.2 A backslash (\\ ) followed by any special character is a one-character RE that matches the special character itself. The special characters are:

a. `.` , `*` , `[` , and `\\` (period, asterisk, left square bracket, and backslash, respectively), which are always special, *except* when they appear within square brackets (`[]` ; see 1.4 below).

b. ^ (caret or circumflex), which is special at the *beginning* of an *entire* RE (see 4.1 and 4.3 below), or when it immediately follows the left of a pair of square brackets (`[]`) (see 1.4 below).

c. `$` (dollar sign), which is special at the end of an *entire* RE (see 4.2 below).

d. The character used to bound (that is, delimit) an entire RE, which is special for that RE (for example, see how slash (`/`) is used in the `g` command, below.)

1.3 A period (`.`) is a one-character RE that matches any character except new-line.

1.4 A non-empty string of characters enclosed in square brackets (`[]`) is a one-character RE that matches *any one* character in that string. If, however, the first character of the string is a circumflex (^), the one-character RE matches any character *except* new-line and the remaining characters in the string. The ^ has this special meaning *only* if it occurs first in the string. The minus (`-`) may be used to indicate a range of consecutive characters; for example, `[0-9]` is equivalent to `[0123456789]` . The `-` loses this special meaning if it occurs first (after an initial ^, if any) or last in the string. The right square bracket (`]`) does not terminate such a string when it is the first character within it (after an initial ^, if any); for example, `[]a-f]` matches either a right square bracket (`]`) or one of the ASCII letters `a` through `f` inclusive. The four characters listed in 1.2.a above stand for themselves within such a string of characters.

The following rules may be used to construct REs from one-character REs:

2.1 A one-character RE is a RE that matches whatever the one-character RE matches.

2.2 A one-character RE followed by an asterisk (`*`) is a RE that matches `0` or more occurrences of the one-character RE. If there is any choice, the longest leftmost string that permits a match is chosen.

2.3 A one-character RE followed by `\\{` *m* `\\}` , `\\{` *m,* `\\}` , or `\\{` *m,n* `\\}` is a RE that matches a *range* of occurrences of the one-character RE. The values of *m* and *n* must be non-negative integers less than 256; `\\{` *m* `\\}` matches *exactly* *m* occurrences; `\\{` *m,* `\\}` matches *at least m* occurrences; `\\{` *m,n* `\\}` matches *any number* of occurrences *between m* and *n* inclusive. Whenever a choice exists, the RE matches as many occurrences as possible.

2.4 The concatenation of REs is a RE that matches the concatenation of the strings matched by each component of the RE.

2.5 A RE enclosed between the character sequences `\\(` and `\\)` is a RE that matches whatever the unadorned RE matches.

2.6 The expression `\\` *n* matches the same string of characters as was matched by an expression enclosed between `\\(` and `\\)` *earlier* in the same RE. Here *n* is a digit; the sub-expression specified is that beginning with the *n* -th occurrence of `\\(` counting from the left. For example, the expression `^\\(.*\\)\\1$` matches a line consisting of two repeated appearances of the same string.

An RE may be constrained to match words.

3.1 `\\<` constrains a RE to match the beginning of a string or to follow a character that is not a digit, underscore, or letter. The first character matching the RE must be a digit, underscore, or letter.

3.2 `\\>` constrains a RE to match the end of a string or to precede a character that is not a digit, underscore, or letter.

An *entire RE* may be constrained to match only an initial segment or final segment of a line (or both).

4.1 A circumflex (^) at the beginning of an entire RE constrains that RE to match an *initial* segment of a line.

4.2 A dollar sign ($) at the end of an entire RE constrains that RE to match a *final* segment of a line.

4.3 The construction ^*entire RE* $ constrains the entire RE to match the entire line.

The null RE (for example, //) is equivalent to the last RE encountered.

Addressing with REs

Addresses are constructed as follows:

1. The character ". " addresses the current line.
2. The character "$ " addresses the last line of the buffer.
3. A decimal number *n* addresses the *n* -th line of the buffer.
4. '*x* addresses the line marked with the mark name character *x* , which must be an ASCII lower-case letter (a -z). Lines are marked with the k command described below.
5. A RE enclosed by slashes (/) addresses the first line found by searching *forward* from the line *following* the current line toward the end of the buffer and stopping at the first line containing a string matching the RE. If necessary, the search wraps around to the beginning of the buffer and continues up to and including the current line, so that the entire buffer is searched.
6. A RE enclosed in question marks (?) addresses the first line found by searching *backward* from the line *preceding* the current line toward the beginning of the buffer and stopping at the first line containing a string matching the RE. If necessary, the search wraps around to the end of the buffer and continues up to and including the current line.
7. An address followed by a plus sign (+) or a minus sign (-) followed by a decimal number specifies that address plus (respectively minus) the indicated number of lines. A shorthand for .+5 is .5.
8. If an address begins with + or - , the addition or subtraction is taken with respect to the current line; for example, -5 is understood to mean .-5 .
9. If an address ends with + or - , then 1 is added to or subtracted from the address, respectively. As a consequence of this rule and of Rule 8, immediately above, the address - refers to the line preceding the current line. (To maintain compatibility with earlier versions of the editor, the character ^ in addresses is entirely equivalent to - .) Moreover, trailing + and - characters have a cumulative effect, so -- refers to the current line less 2.
10. For convenience, a comma (,) stands for the address pair 1,$, while a semicolon (;) stands for the pair .,$.

Characters With Special Meaning

Characters that have special meaning except when they appear within square brackets ([]) or are preceded by \\ are: . , * , [, \\ . Other special characters, such as $ have special meaning in more restricted contexts.

The character ^ at the beginning of an expression permits a successful match only immediately after a newline, and the character $ at the end of an expression requires a trailing newline.

Two characters have special meaning only when used within square brackets. The character - denotes a range, [*c* - *c*] , unless it is just after the open bracket or before the closing bracket, [- *c*] or [*c* -] in which case it has no special meaning. When used within brackets, the character ^ has the meaning *complement of* if it immediately follows the open bracket (example: [^ *c*]); elsewhere between brackets (example: [*c* ^]) it stands for the ordinary character ^ .

The special meaning of the \\ operator can be escaped only by preceding it with another \\ , for example \\\\ .

Macros

Programs must have the following five macros declared before the `#include <regexp.h>` statement. These macros are used by the `compile( )` routine. The macros `GETC` , `PEEKC` , and `UNGETC` operate on the regular expression given as input to `compile( )` .

`GETC` — This macro returns the value of the next character (byte) in the regular expression pattern. Successive calls to `GETC` should return successive characters of the regular expression.

`PEEKC` — This macro returns the next character (byte) in the regular expression. Immediately successive calls to `PEEKC` should return the same character, which should also be the next character returned by `GETC` .

`UNGETC` — This macro causes the argument `c` to be returned by the next call to `GETC` and `PEEKC` . No more than one character of pushback is ever needed and this character is guaranteed to be the last character read by `GETC` . The return value of the macro `UNGETC(c)` is always ignored.

`RETURN(` *ptr* `)` — This macro is used on normal exit of the `compile( )` routine. The value of the argument *ptr* is a pointer to the character after the last character of the compiled regular expression. This is useful to programs which have memory allocation to manage.

`ERROR(` *val* `)` — This macro is the abnormal return from the `compile( )` routine. The argument *val* is an error number (see ERRORS below for meanings). This call should never return.

compile( )

The syntax of the `compile( )` routine is as follows:

```
compile(instring, expbuf, endbuf, eof)
```

The first parameter, *instring* , is never used explicitly by the `compile( )` routine but is useful for programs that pass down different pointers to input characters. It is sometimes used in the `INIT` declaration (see below). Programs which call functions to input characters or have characters in an external array can pass down a value of `(char *)0` for this parameter.

The next parameter, *expbuf* , is a character pointer. It points to the place where the compiled regular expression will be placed.

The parameter *endbuf* is one more than the highest address where the compiled regular expression may be placed. If the compiled expression cannot fit in `(endbuf-expbuf)` bytes, a call to `ERROR(50)` is made.

The parameter *eof* is the character which marks the end of the regular expression. This character is usually a `/` .

Each program that includes the `<regexp.h>` header file must have a `#define` statement for `INIT` . It is used for dependent declarations and initializations. Most often it is used to set a register variable to point to the beginning of the regular expression so that this register variable can be used in the declarations for `GETC` , `PEEKC` , and `UNGETC` . Otherwise it can be used to declare external variables that might be used by `GETC` , `PEEKC` and `UNGETC` . (See `EXAMPLES` below.)

step(), advance()

The first parameter to the `step( )` and `advance( )` functions is a pointer to a string of characters to be checked for a match. This string should be null terminated.

The second parameter, *expbuf* , is the compiled regular expression which was obtained by a call to the function `compile( )` .

The function `step( )` returns non-zero if some substring of *string* matches the regular expression in *expbuf* and `0` if there is no match. If there is a match, two external character pointers are set as a side effect to the call to `step( )` . The variable `loc1` points to the first character that matched the regular expression; the variable `loc2` points to the character after the last character that matches the regular expression. Thus if the regular expression matches the entire input string, `loc1` will point to the first character of *string* and `loc2` will point to the null at the end of *string* .

The function `advance( )` returns non-zero if the initial substring of *string* matches the regular expression in *expbuf* . If there is a match, an external character pointer, `loc2` , is set as a side effect. The variable `loc2` points to the next character in *string* after the last character that matched.

When `advance( )` encounters a * or `\\{ \\}` sequence in the regular expression, it will advance its pointer to the string to be matched as far as possible and will recursively call itself trying to match the rest of the string to the rest of the regular expression. As long as there is no match, `advance( )` will back up along the string until it finds a match or reaches the point in the string that initially matched the * or `\\{ \\}` . It is sometimes desirable to stop this backing up before the initial point in the string is reached. If the external character pointer `locs` is equal to the point in the string at sometime during the backing up process, `advance( )` will break out of the loop that backs up and will return zero.

The external variables `circf` , `sed` , and `nbra` are reserved.

EXAMPLES

EXAMPLE 1 The following is an example of how the regular expression macros and calls might be defined by an application program:

```
#define INIT         register char *sp = instring;
#define GETC       (*sp++)
#define PEEKC      (*sp)
#define UNGETC(c)    (--sp)
#define RETURN(*c)    return;
#define ERROR(c)     regerr
#include <regexp.h>
 . . .
      (void) compile(*argv, expbuf, &expbuf[ESIZE],'\\0');
 . . .
      if (step(linebuf, expbuf))
                        succeed;
```

DIAGNOSTICS

The function `compile( )` uses the macro `RETURN` on success and the macro `ERROR` on failure (see above). The functions `step( )` and `advance( )` return non-zero on a successful match and zero if there is no match. Errors are:

11	range endpoint too large.
16	bad number.
25	`\\` *digit* out of range.
36	illegal or missing delimiter.
41	no remembered search string.
42	`\\( \\)` imbalance.
43	too many `\\(` .
44	more than 2 numbers given in `\\{ \\}` .
45	`}` expected after `\\` .

46	first number exceeds second in `\\{ \\}` .
49	`[ ]` imbalance.
50	regular expression overflow.

SEE ALSO

`regex`(5)

NAME

SEAM – overview of Sun Enterprise Authentication Mechanism

DESCRIPTION

SEAM (Sun Enterprise Authentication Mechanism) authenticates clients in a network environment, allowing for secure transactions. (A client may be a user or a network service) SEAM validates the identity of a client and the authenticity of transferred data. SEAM is a *single-sign-on* system, meaning that a user needs to provice a password only at the beginning of a session. SEAM is based on the Kerberos™ system developed at MIT, and is compatible with Kerberos V5 systems over heterogeneous networks.

SEAM works by granting clients *tickets*, which uniquely identify a client, and which have a finite lifetime. A client possessing a ticket is automatically validated for network services for which it is entitled; for example, a user with a valid SEAM ticket may rlogin into another machine running SEAM without having to identify itself. Because each client has a unique ticket, its identity is guaranteed.

To obtain tickets, a client must first initialize the SEAM session, either by using the `kinit`(1) command or a PAM module. (See `pam_krb5`(5)). `kinit` prompts for a password, and then communicates with a *Key Distribution Center* (KDC). The KDC returns a *Ticket-Granting Ticket* (TGT) and prompts for a confirmation password. If the client confirms the password, it can use the Ticket-Granting Ticket to obtain tickets for specific network services. Because tickets are granted transparently, the user need not worry about their management. Current tickets may be viewed by using the `klist`(1) command.

Tickets are valid according to the system *policy* set up at installation time. For example, tickets have a default lifetime for which they are valid. A policy may further dictate that privileged tickets, such as those belonging to root, have very short lifetimes. Policies may allow some defaults to be overruled; for example, a client may request a ticket with a lifetime greater or less than the default.

Tickets can be renewed using `kinit`. Tickets are also *forwardable*, allowing you to use a ticket granted on one machine on a different host. Tickets can be destroyed by using `kdestroy`(1). It is a good idea to include a call to `kdestroy` in your `.logout` file.

Under SEAM, a client is referred to as a *principal*. A principal takes the following form:

```
primary/instance@REALM
```

primary — A user, a host, or a service.

instance — A qualification of the primary. If the primary is a host — indicated by the keyword `host`— then

	the instance is the fully-qualified domain name of that host. If the primary is a user or service, then the instance is optional. Some instances, such as `admin` or `root`, are privileged.
realm	The Kerberos equivalent of a domain; in fact, in most cases the realm is directly mapped to a DNS domain name. SEAM realms are given in upper-case only. For examples of principal names, see the EXAMPLES.

By taking advantage of the General Security Services API (GSS-API), SEAM offers, besides user authentication, two other types of security service: *integrity*, which authenticates the validity of transmitted data, and *privacy*, which encrypts transmitted data. Developers can take advantage of the GSS-API through the use of the RPCSEC_GSS API interface (see `rpcsec_gss`(3NSL)).

EXAMPLES

EXAMPLE 1 Examples of valid principal names

The following are examples of valid principal names:

```
joe
joe/admin
joe@ENG.ACME.COM
joe/admin@ENG.ACME.COM
rlogin/bigmachine.eng.acme.com@ENG.ACME.COM
host/bigmachine.eng.acme.com@ENG.ACME.COM
```

The first four cases are *user principals*. In the first two cases, it is assumed that the user `joe` is in the same realm as the client, so no realm is specified. Note that `joe`and `joe/admin` are different principals, even if the same user uses them; `joe/admin` has different privileges from `joe`. The fifth case is a *service principal*, while the final case is a *host principal*. The word `host` is required for host principals. With host principals, the instance is the fully qualified hostname. Note that the words `admin` and `host` are reserved keywords.

SEE ALSO

`kdestroy`(1), `kinit`(1), `klist`(1), `kpasswd`(1), `krb5.conf`(5)

Sun Enterprise Authentication Mechanism Guide

NOTES

If you enter your username and `kinit` responds with this message:

```
Principal unknown (kerberos)
```

you haven't been registered as a SEAM user. See your system administrator or the *Sun Enterprise Authentication Mechanism Guide*.

NAME

sgml, solbook – Standard Generalized Markup Language

DESCRIPTION

Standard Generalized Markup Language (SGML) is the ISO standard 8879:1986 that describes a syntax for marking up documents with tags that describe the purpose of the text rather than the appearance on the page. This form of markup facilitates document interchange between different platforms and applications. SGML allows the management of information as data objects rather than text on a page.

In an SGML document the main structural components are called `elements` . The organization and structure of a document and the meaning of elements are described in the Document Type Definition (DTD). Elements are the *tags* that identify the content. Element names may be descriptive of the content for ease of use. For example `<para>` for paragraphs. Elements can have `attributes` which are used to modify or refine the properties or characteristics of the element. Within the DTD a valid context for each element is defined and a framework is provided for the types of elements that constitute a compliant document.

Another component of the DTD is `entities` . Entities are a collection of characters that can be referenced as a unit. Entities are similar to constants in a programming language such as C. They can be defined and referenced. An entity can represent one character or symbol which does not appear on a standard keyboard, a word or group of words, or an entire separate sgml marked-up file. Entities allow reuse of standard text.

There is no single standard DTD , but the de facto standard for the computer industry is the DocBook DTD , developed and maintained by the Davenport Group. Within Sun, the SolBook DTD , which is a proper subset of DocBook DTD , is used when writing reference manual pages. The SolBook DTD contains a number of tags that are designed for the unique needs of the reference pages.

SolBook Elements

Elements are defined with a hierarchical structure that gives a structure to the document. The following is a description of some of the elements from the SolBook DTD which are used for reference pages.

DOCTYPE

The first line in an SGML file that identifies the location of the DTD that is used to define the document. The `<!DOCTYPE` string is what the SGML -aware `man`(1) command uses to identify that a file is formatted in SGML rather than `nroff`(1) .

RefEntry

The top layer element that contains a reference page is `<refentry>` . All of the text and other tags must be contained within this tag.

RefMeta

The next tag in a reference page is `<refmeta>` , which is a container for several other tags. They are:

<refentrytitle> This is the title of the reference page. It is equivalent to the name of the reference page's file name, without the section number extension.

<manvolnum> This is the section number that the reference page resides in. The contents may be a text entity reference.

<refmiscinfo> There are one or more `<refmiscinfo>` tags which contain *meta* information. Meta information is information about the reference page. The `<refmiscinfo>` tag has the `class` attribute. There are four classes that are routinely used.

date	This is the date that the file was last modified. By consensus this date is changed only when the technical information on the page changes and not simply for an editorial change.
sectdesc	This is the section title of the reference page; for example `User Commands` . The value of this attribute may be a text entity reference.
software	This is the name of the software product that the topic discussed on the reference page belongs to. For example UNIX commands are part of the SunOS x.x release. The value of this attribute may be a text entity reference.
arch	This is the architectural platform limitation of the subject discussed on the reference page. If there are no limitations the value used is `generic` . Other values are `sparc` and `IA` .
copyright	This attribute contains the Sun Microsystems copyright. Any other copyrights that may pertain to the individual reference page file should be entered as separate `<refmiscinfo>` entries. The value of this attribute may be a text entity reference.

RefNameDiv This tag contains the equivalent information to the `.TH` macro line in an `nroff`(1) reference page. `<refnamediv>` contains three tags. These tags contain the text that is before and after the ‘-’ (dash) on the `NAME` line.

<refname> These are the names of the topics that are discussed in the file. There may be more than one `<refname>` for a page. The first `<refname>` must match the name of the file

and the `<refentrytitle>` . If there are more than one `<refname>` tags, each is separated by a ',' (comma). The comma is generated by the publisher of sgml files, so it should not be typed. This is referred to as *auto-generated* text.

<refpurpose> The text after the dash on the NAME line is contained in this tag. This is a short summary of what the object or objects described on the reference page do or are used for. The dash is also auto-generated and should not be typed in.

<refdiscriptor> In some cases the `<refentrytitle>` is a general topic descriptor of a group of related objects that are discussed on the same page. In this case the first tag after the `<refnamediv>` is a `<refdiscriptor>` . The `<refname>` tags follow. Only one `<refdiscriptor>` is allowed, and it should match the `<refentrytitle>` .

RefSynopsisDiv

The SYNOPSIS line of the reference page is contained by this tag. There is a `<title>` that usually contains an entity reference. The text is the word SYNOPSIS. There are several tags within `<refsynopsisdiv>` that are designed specifically for the type of synopsis that is used in the different reference page sections. The three types are:

<cmdsynopsis> Used for commands and utilities pages.

<funcsynopsis> Used for programming interface pages.

<synopsis> Used for pages that do not fall into the other two categories.

RefSect1

This tag is equivalent to the .SH nroff macro. It contains a `<title>` element that is the title of the reference page section. Section names are the standard names such as DESCRIPTION, OPTIONS, PARAMETERS, SEE ALSO, and others. The contents of the `<title>` may be a text entity reference.

RefSect2

This tag is equivalent to the .SS nroff macro. It contains a `<title>` element that contains the text of the sub-section heading. `<refsect2>` tags may also be used within a `<refsynopsisdiv>` as a sub-section heading for the SYNOPSIS section.

Block Elements

There are a number of block elements that are used for grouping text. This is a list of some of these elements.

<para> This tag is used to contain a paragraph of text.

<variablelist> This tag is used to create two column lists. For example descriptions for command options, where the first column lists the option and the second column describes the option.

<orderedlist> An list of items in a specific order.

<itemizedlist>	A list of items that are marked with a character such as a bullet or a dash.
<literallayout>	Formatted program output as produced by a program or command. This tag is a container for lines set off from the main text in which line breaks, tabs, and leading white space are significant.
<programlisting>	A segment of program code. Line breaks and leading white space are significant.
<table>	This tag contains the layout and content for tabular formatting of information. `<table>` has a required `<title>` .
<informaltable>	This tag is the same as the `<table>` tag except the `<title>` is not required.
<example>	This tag contains examples of source code or usage of commands. It contains a required `<title>` .
<informalexample>	This tag is the same as the `<example>` tag except the `<title>` is not required.

Inline Elements

The inline elements are used for tagging text.

<command>	An executable program or the entry a user makes to execute a command.
<function>	A subroutine in a program or external library.
<literal>	Contains any literal string.
<parameter>	An argument passed to a computer program by a function or routine.
<inlineequation>	An untitled mathematical equation occurring in-line.
<link>	A hypertext link to text within a book, in the case of the reference manual it is used to cross reference to another reference page.
<olink>	A hypertext link used to create cross references to books other than the reference manual.
<xref>	A cross reference to another part of the same reference page.

SEE ALSO

man(1), nroff(1), man(5)

NAME

smartcard – overview of smartcard features on Solaris

DESCRIPTION

The smartcard framework provides a mechanism to abstract the details of interacting with smart cards and smart cardreaders (called card terminals). The framework is based on the OpenCard Framework V1.1 (OCF) with Sun extensions to allow OCF to operate in a multi-user environment. The core OCF software protocol stack is implemented as a system service daemon. This implementation allows smartcards and card terminals to be shared cooperatively among many different clients on the system while providing access control to the smart card and card terminal resources on a per-UID basis.

An event dispatcher is provided to inform clients of events occuring on the card and at the card terminal, such as card insertion and card removal.

A high-level authentication mechanism is provided to allow clients to perform smartcard-based authentications without requiring knowledge of specific card or reader authentication features.

A set of applet administration tools is provided for JavaCards that support downloading Java applets (although applet build tools are not provided).

Administration of the smartcard framework is provided with the `smartcard`(1M) command line administration utility and the smartcardguiadmin(1) GUI administration tool.

Support for several card terminals is provided:

- Sun External Smart Card Reader I (see `ocf_escr1`(7D))
- Sun Internal Smart Card Reader I (see `ocf_iscr1`(7D))
- Dallas iButton Serial Reader (see `ocf_iButton`(7D))

Support for several smart cards is provided:

- Schlumberger Cyberflex Access JavaCard
- Schlumberger MicroPayflex
- Dallas Semiconductor Java iButton JavaCard

Each of the supported cards has a complete set of OCF card services that implement the necessary functionality for authentication and secure storage of data. For the two supported JavaCards, an authentication and secure data storage applet is provided that can be loaded into these cards with the supplied applet administration tools. See `smartcard`(1M).

A PAM smart card module is provided to allow PAM clients to use smartcard-based authentication. See `pam_smartcard`(5)

CDE is able to use the PAM smart card module for `dtlogin` and `dtsession` authentication. CDE also uses the smartcard framework event dispatcher to listen for events on the card terminal and provide corresponding visual feedback to the user.

SEE ALSO

ocfserv(1M), smartcard(1M), pam_start(3PAM), pam_smartcard(5), ocf_escri(7D), ocf_iButton(7D), ocf_iscr1(7D), scmi2c(7D)

NAME

standards, ANSI, C, ISO, POSIX, POSIX.1, POSIX.2, SUS, SUSv2, SVID, SVID3, XNS, XNS4, XNS5, XPG, XPG3, XPG4, XPG4v2 – standards and specifications supported by Solaris

DESCRIPTION

Solaris 7 supports IEEE Std 1003.1 and IEEE Std 1003.2, commonly known as POSIX.1 and POSIX.2, respectively. The following table lists each version of these standards with a brief description and the SunOS or Solaris release that first conformed to it.

POSIX Standard	Description	Release
POSIX.1-1988	system interfaces and headers	SunOS 4.1
POSIX.1-1990	POSIX.1-1988 update	Solaris 2.0
POSIX.1b-1993	realtime extensions	Solaris 2.4
POSIX.1c-1996	threads extensions	Solaris 2.6
POSIX.2-1992	shell and utilities	Solaris 2.5
POSIX.2a-1992	interactive shell and utilities	Solaris 2.5

Solaris 7 also supports the X/Open Common Applications Environment (CAE) Portability Guide Issue 3 (XPG3) and Issue 4 (XPG4), Single UNIX Specification (SUS, also known as XPG4v2), and Single UNIX Specification, Version 2 (SUSv2). Both XPG4 and SUS include Networking Services Issue 4 (XNS4). SUSv2 includes Networking Services Issue 5 (XNS5).

Solaris 7 also supports two application programming environments, ILP32 (32-bit) and LP64 (64-bit).

The following table lists each X/Open specification with a brief description and the SunOS or Solaris release that first conformed to it.

X/Open CAE Specification	Description	Release
XPG3	superset of POSIX.1-1988 containing utilities from SVID3	SunOS 4.1
XPG4	superset of POSIX.1-1990, POSIX.2-1992, and POSIX.2a-1992 containing extensions to POSIX standards from XPG3	Solaris 2.4
SUS (XPG4v2)	superset of XPG4 containing historical BSD interfaces widely used by common application packages	Solaris 2.6

X/Open CAE Specification	Description	Release
XNS4	sockets and XTI interfaces	Solaris 2.6
SUSv2	superset of SUS extended to support POSIX.1b-1993, POSIX.1c-1996, and ISO/IEC 9899 (C Standard) Amendment 1	Solaris 7
XNS5	superset and LP64-clean derivative of XNS4.	Solaris 7

The XNS4 specification is safe for use only in ILP32 (32-bit) environments and should not be used for LP64 (64-bit) application environments. Use XNS5, which has LP64-clean interfaces that are portable across ILP32 and LP64 environments.

Solaris 7 has been branded to conform to The Open Group's UNIX 98 Product Standard.

Solaris releases 2.0 through 7 also support the interfaces specified by the System V Interface Definition, Third Edition, Volumes 1 through 4 (SVID3). Note, however, that since the developers of this specification (UNIX Systems Laboratories) are no longer in business and since this specification defers to POSIX and X/Open CAE specifications, there is some disagreement about what is currently required for conformance to this specification.

When `Sun WorkShop Compiler(TM) C 4.2` is installed, Solaris releases 2.0 through 7 support the ANSI X3.159-1989 Programming Language - C and ISO/IEC 9899:1990 Programming Language - C (C) interfaces.

When `Sun WorkShop Compiler(TM) C 5.0` is installed, Solaris 7 also supports ISO/IEC 9899 Amendment 1: C Integrity.

Utilities

If the behavior required by POSIX.2, POSIX.2a, XPG4, SUS, or SUSv2 conflicts with historical Solaris utility behavior, the original Solaris version of the utility is unchanged; a new version that is standard-conforming has been provided in `/usr/xpg4/bin` . For applications wishing to take advantage of POSIX.2, POSIX.2a, XPG4, SUS, or SUSv2 features, the `PATH` (`sh` or `ksh`) or `path` (`csh`) environment variables should be set with `/usr/xpg4/bin` preceding any other directories in which utilities specified by those specifications are found, such as `/bin` , `/usr/bin` , `/usr/ucb` , and `/usr/ccs/bin` .

Feature Test Macros

Feature test macros are used by applications to indicate additional sets of features that are desired beyond those specified by the C standard. If an application uses only those interfaces and headers defined by a particular standard (such as POSIX or X/Open CAE), then it need only define the appropriate feature test macro specified by that standard. If the application is using interfaces and headers not defined by that standard, then in addition to defining the appropriate standard feature test macro, it must also define `__EXTENSIONS__` . Defining `__EXTENSIONS__` provides the application with access to all interfaces and headers not in conflict with the specified standard. The application must define `__EXTENSIONS__` either at compile time or within the application.

ANSI/ISO C

No feature test macros need to be defined to indicate that an application is a conforming C application.

POSIX

Applications that are intended to be conforming POSIX.1 applications must define the feature test macros specified by the standard before including any headers. For the standards listed below, applications must define the feature test macros listed. Application writers must check the corresponding standards for other macros that can be queried to determine if desired options are supported by the implementation.

POSIX Standard	Feature Test Macros
POSIX.1-1990	`_POSIX_SOURCE`
POSIX.1-1990 and	`_POSIX_SOURCE` and
POSIX.2-1992 C-Language	`_POSIX_C_SOURCE=2`
Bindings Option	
POSIX.1b-1993	`_POSIX_C_SOURCE=199309L`
POSIX.1c-1996	`_POSIX_C_SOURCE=199506L`

SVID3

The SVID3 specification does not specify any feature test macros to indicate that an application is written to meet SVID3 requirements. The SVID3 specification was written before the C standard was completed.

X/Open CAE

To build or compile an application that conforms to one of the X/Open CAE specifications, use the following guidelines. Applications need not set the POSIX feature test macros if they require both CAE and POSIX functionality.

XPG3 The application must define `_XOPEN_SOURCE` with a value other than 500 (preferably 1).

XPG4	The application must define _XOPEN_SOURCE with a value other than 500 (preferably 1) and set _XOPEN_VERSION=4 .
SUS (XPG4v2)	The application must define _XOPEN_SOURCE with a value other than 500 (preferably 1) and set _XOPEN_SOURCE_EXTENDED=1 .
SUSv2	The application must define _XOPEN_SOURCE=500 .

Compilation

A POSIX.2-, XPG4-, SUS-, or SUSv2-conforming implementation must include an ANSI X3.159-1989 (ANSI C Language) standard-conforming compilation system and the cc and c89 utilities. Solaris 7 was tested with the cc and c89 utilities and the compilation system provided by Sun WorkShop Compiler(TM) C 5.0 in the SPARC and IA environments. When cc is used to link applications, /usr/ccs/lib/values-xpg4.o must be specified on any link/load command line, but the preferred way to build applications is described below.

An XNS4- or XNS5-conforming application must include –l XNS on any link/load command line.

If the compiler suppports the redefine_extname pragma feature (the Sun WorkShop Compiler(TM) C 4.2 and Sun WorkShop Compiler(TM) C 5.0 compiler defines the macro __PRAGMA_REDEFINE_EXTNAME to indicate that it supports this feature), then the standard headers use #pragma redefine_extname directives to properly map function names onto library entry point names. This mapping provides full support for ISO C, POSIX, and X/Open namespace reservations. The Sun WorkShop Compiler(TM) C 5.0 compiler was used for all branding and certification tests for Solaris 7.

If this pragma feature is not supported by the compiler, the headers use the #define directive to map internal function names onto appropriate library entry point names. In this instance, applications should avoid using the explicit 64-bit file offset symbols listed on the lf64(5) manual page, since these names are used by the implementation to name the alternative entry points.

When using Sun WorkShop Compiler(TM) C 5.0 , applications conforming to the specifications listed above should be compiled using the utilities and flags indicated in the following table:

Specification	Compiler/Flags	Feature Test Macros
ANSI/ISO C	c89	none
SVID3	cc –Xt	none
POSIX.1-1990	c89	_POSIX_SOURCE
POSIX.1-1990 and	c89	_POSIX_SOURCE and
POSIX.2-1992		POSIX_C_SOURCE=2

Specification	Compiler/Flags	Feature Test Macros
C-Language Bindings Option		
POSIX.1b-1993	c89	_POSIX_C_SOURCE=199309L
POSIX.1c-1996	c89	_POSIX_C_SOURCE=199506L
CAE XPG3	cc –Xa	_XOPEN_SOURCE
CAE XPG4	c89	_XOPEN_SOURCE and _XOPEN_VERSION=4
SUS (CAE XPG4v2) (includes XNS4)	c89	_XOPEN_SOURCE and _XOPEN_SOURCE_EXTENDED=1
SUSv2(includes XNS5)	c89	_XOPEN_SOURCE=500

For platforms supporting the LP64 (64-bit) programming environment where the SC5.0 Compilers have been installed, SUSv2-conforming LP64 applications using XNS5 library calls should be built with command lines of the form:

```
c89 $(getconf XBS5_LP64_OFF64_CFLAGS) -D_XOPEN_SOURCE=500 \\
    $(getconf XBS5_LP64_OFF64_LDFLAGS) foo.c -o foo \\
    $(getconf XBS5_LP64_OFF64_LIBS) -lxnet
```

SEE ALSO

sysconf(3C) , environ(5) , lf64(5)

NAME

sticky – mark files for special treatment

DESCRIPTION

The *sticky bit* (file mode bit `01000`, see `chmod`(2)) is used to indicate special treatment of certain files and directories. A directory for which the sticky bit is set restricts deletion of files it contains. A file in a sticky directory may only be removed or renamed by a user who has write permission on the directory, and either owns the file, owns the directory, or is the super-user. This is useful for directories such as `/tmp`, which must be publicly writable, but should deny users permission to arbitrarily delete or rename the files of others.

If the sticky bit is set on a regular file and no execute bits are set, the system's page cache will not be used to hold the file's data. This bit is normally set on swap files of diskless clients so that accesses to these files do not flush more valuable data from the system's cache. Moreover, by default such files are treated as swap files, whose inode modification times may not necessarily be correctly recorded on permanent storage.

Any user may create a sticky directory. See `chmod` for details about modifying file modes.

FILES

`/tmp`

SEE ALSO

`chmod`(1), `chmod`(2), `chown`(2), `mkdir`(2)

BUGS

`mkdir`(2) will not create a directory with the sticky bit set.

NAME

term – conventional names for terminals

DESCRIPTION

Terminal names are maintained as part of the shell environment in the environment variable `TERM.` See `sh`(1), `profile`(4), and `environ`(5). These names are used by certain commands (for example, `tabs`, `tput`, and `vi`) and certain functions (for example, see `curses`(3CURSES)).

Files under `/usr/share/lib/terminfo` are used to name terminals and describe their capabilities. These files are in the format described in `terminfo`(4). Entries in `terminfo` source files consist of a number of comma-separated fields. To print a description of a terminal *term*, use the command `infocmp -I` *term*. See `infocmp`(1M). White space after each comma is ignored. The first line of each terminal description in the `terminfo` database gives the names by which `terminfo` knows the terminal, separated by bar (`|`) characters. The first name given is the most common abbreviation for the terminal (this is the one to use to set the environment variable `TERMINFO` in `$HOME/.profile` ; see `profile`(4)), the last name given should be a long name fully identifying the terminal, and all others are understood as synonyms for the terminal name. All names but the last should contain no blanks and must be unique in the first 14 characters; the last name may contain blanks for readability.

Terminal names (except for the last, verbose entry) should be chosen using the following conventions. The particular piece of hardware making up the terminal should have a root name chosen, for example, for the AT&T 4425 terminal, `att4425`. This name should not contain hyphens, except that synonyms may be chosen that do not conflict with other names. Up to 8 characters, chosen from the set `a` through `z` and `0` through `9`, make up a basic terminal name. Names should generally be based on original vendors rather than local distributors. A terminal acquired from one vendor should not have more than one distinct basic name. Terminal sub-models, operational modes that the hardware can be in, or user preferences should be indicated by appending a hyphen and an indicator of the mode. Thus, an AT&T 4425 terminal in 132 column mode is `att4425–w`. The following suffixes should be used where possible:

Suffix	Meaning	Example
–w	Wide mode (more than 80 columns)	att4425–w
–am	With auto. margins (usually default)	vt100–am
–nam	Without automatic margins	vt100–nam
–*n*	Number of lines on the screen	aaa–60

–na	No arrow keys (leave them in local)	c100–na
–np	Number of pages of memory	c100–4p
–rv	Reverse video	att4415–rv

To avoid conflicts with the naming conventions used in describing the different modes of a terminal (for example, `–w`), it is recommended that a terminal's root name not contain hyphens. Further, it is good practice to make all terminal names used in the `terminfo`(4) database unique. Terminal entries that are present only for inclusion in other entries via the `use=` facilities should have a '+' in their name, as in `4415+nl`.

Here are some of the known terminal names: (For a complete list, enter the command `ls -C /usr/share/lib/terminfo/?`).

2621,hp2621	Hewlett-Packard 2621 series
2631	Hewlett-Packard 2631 line printer
2631–c	Hewlett-Packard 2631 line printer, compressed mode
2631–e	Hewlett-Packard 2631 line printer, expanded mode
2640,hp2640	Hewlett-Packard 2640 series
2645,hp2645	Hewlett-Packard 2645 series
3270	IBM Model 3270
33,tty33	AT&T Teletype Model 33 KSR
35,tty35	AT&T Teletype Model 35 KSR
37,tty37	AT&T Teletype Model 37 KSR
4000a	Trendata 4000a
4014,tek4014	TEKTRONIX 4014
40,tty40	AT&T Teletype Dataspeed 40/2
43,tty43	AT&T Teletype Model 43 KSR
4410,5410	AT&T 4410/5410 in 80-column mode, version 2
4410–nfk,5410–nfk	AT&T 4410/5410 without function keys, version 1
4410–nsl,5410–nsl	AT&T 4410/5410 without pln defined
4410–w,5410–w	AT&T 4410/5410 in 132-column mode
4410v1,5410v1	AT&T 4410/5410 in 80-column mode, version 1

4410v1–w,5410v1–w	AT&T 4410 / 5410 in 132-column mode, version 1
4415,5420	AT&T 4415 / 5420 in 80-column mode
4415–nl,5420–nl	AT&T 4415 / 5420 without changing labels
4415–rv,5420–rv	AT&T 4415 / 5420 80 columns in reverse video
4415–rv–nl,5420–rv–nl	AT&T 4415 / 5420 reverse video without changing labels
4415–w,5420–w	AT&T 4415 / 5420 in 132-column mode
4415–w–nl,5420–w–nl	AT&T 4415 / 5420 in 132-column mode without changing labels
4415–w–rv,5420–w–rv	AT&T 4415 / 5420 132 columns in reverse video
4418,5418	AT&T 5418 in 80-column mode
4418–w,5418–w	AT&T 5418 in 132-column mode
4420	AT&T Teletype Model 4420
4424	AT&T Teletype Model 4424
4424-2	AT&T Teletype Model 4424 in display function group ii
4425,5425	AT&T 4425 / 5425
4425–fk,5425–fk	AT&T 4425 / 5425 without function keys
4425–nl,5425–nl	AT&T 4425 / 5425 without changing labels in 80-column mode
4425–w,5425–w	AT&T 4425 / 5425 in 132-column mode
4425–w–fk,5425–w–fk	AT&T 4425 / 5425 without function keys in 132-column mode
4425–nl–w,5425–nl–w	AT&T 4425 / 5425 without changing labels in 132-column mode
4426	AT&T Teletype Model 4426S
450	DASI 450 (same as Diablo 1620)
450–12	DASI 450 in 12-pitch mode
500,att500	AT&T-IS 500 terminal
510,510a	AT&T 510 / 510a in 80-column mode
513bct,att513	AT&T 513 bct terminal
5320	AT&T 5320 hardcopy terminal
5420_2	AT&T 5420 model 2 in 80-column mode

5420_2−w	AT&T 5420 model 2 in 132-column mode
5620,dmd	AT&T 5620 terminal 88 columns
5620−24,dmd−24	AT&T Teletype Model DMD 5620 in a 24x80 layer
5620−34,dmd−34	AT&T Teletype Model DMD 5620 in a 34x80 layer
610,610bct	AT&T 610 bct terminal in 80-column mode
610−w,610bct−w	AT&T 610 bct terminal in 132-column mode
630,630MTG	AT&T 630 Multi-Tasking Graphics terminal
7300,pc7300,unix_pc	AT&T UNIX PC Model 7300
735,ti	Texas Instruments TI735 and TI725
745	Texas Instruments TI745
dumb	generic name for terminals that lack reverse line-feed and other special escape sequences
hp	Hewlett-Packard (same as 2645)
lp	generic name for a line printer
pt505	AT&T Personal Terminal 505 (22 lines)
pt505−24	AT&T Personal Terminal 505 (24-line mode)
sync	generic name for synchronous Teletype Model 4540-compatible terminals

Commands whose behavior depends on the type of terminal should accept arguments of the form −T*term* where *term* is one of the names given above; if no such argument is present, such commands should obtain the terminal type from the environment variable `TERM`, which, in turn, should contain *term*.

FILES

`/usr/share/lib/terminfo/?/*` compiled terminal description database

SEE ALSO

`sh`(1), `stty`(1), `tabs`(1), `tput`(1), `vi`(1), `infocmp`(1M), `curses`(3CURSES), `profile`(4), `terminfo`(4), `environ`(5)

NAME

vgrindefs – vgrind's language definition data base

SYNOPSIS

`/usr/lib/vgrindefs`

DESCRIPTION

`vgrindefs` contains all language definitions for `vgrind`(1). Capabilities in `vgrindefs` are of two types: Boolean capabilities which indicate that the language has some particular feature and string capabilities which give a regular expression or keyword list. Entries may continue onto multiple lines by giving a \ as the last character of a line. Lines starting with # are comments.

Capabilities

The following table names and describes each capability.

Name	Type	Description
`ab`	`str`	Regular expression for the start of an alternate form comment
`ae`	`str`	Regular expression for the end of an alternate form comment
`bb`	`str`	Regular expression for the start of a block
`be`	`str`	Regular expression for the end of a lexical block
`cb`	`str`	Regular expression for the start of a comment
`ce`	`str`	Regular expression for the end of a comment
`id`	`str`	String giving characters other than letters and digits that may legally occur in identifiers (default '_')
`kw`	`str`	A list of keywords separated by spaces
`lb`	`str`	Regular expression for the start of a character constant
`le`	`str`	Regular expression for the end of a character constant
`oc`	`bool`	Present means upper and lower case are equivalent
`pb`	`str`	Regular expression for start of a procedure
`pl`	`bool`	Procedure definitions are constrained to the lexical level matched by the 'px' capability
`px`	`str`	A match for this regular expression indicates that procedure definitions may occur at the next lexical level. Useful for lisp-like languages in which procedure definitions occur as subexpressions of defuns.
`sb`	`str`	Regular expression for the start of a string
`se`	`str`	Regular expression for the end of a string
`tc`	`str`	Use the named entry as a continuation of this one
`tl`	`bool`	Present means procedures are only defined at the top lexical level

Regular Expressions

`vgrindefs` uses regular expressions similar to those of `ex`(1) and `lex`(1). The characters '^', '$', ':', and '\' are reserved characters and must be 'quoted' with a preceding \ if they are to be included as normal characters. The metasymbols and their meanings are:

`$`	The end of a line
`^`	The beginning of a line
`\d`	A delimiter (space, tab, newline, start of line)
`\a`	Matches any string of symbols (like '.*' in lex)
`\p`	Matches any identifier. In a procedure definition (the 'pb' capability) the string that matches this symbol is used as the procedure name.
`()`	Grouping
`|`	Alternation
`?`	Last item is optional
`\e`	Preceding any string means that the string will not match an input string if the input string is preceded by an escape character (\). This is typically used for languages (like C) that can include the string delimiter in a string by escaping it.

Unlike other regular expressions in the system, these match words and not characters. Hence something like '(tramp|steamer)flies?' would match 'tramp', 'steamer', 'trampflies', or 'steamerflies'. Contrary to some forms of regular expressions, `vgrindef` alternation binds very tightly. Grouping parentheses are likely to be necessary in expressions involving alternation.

Keyword List

The keyword list is just a list of keywords in the language separated by spaces. If the 'oc' boolean is specified, indicating that upper and lower case are equivalent, then all the keywords should be specified in lower case.

EXAMPLES

EXAMPLE 1 A sample program.

The following entry, which describes the C language, is typical of a language entry.

```
C|c|the C programming language:\
     :pb=^\d?*?\d?\p\d?(\a?\)(\d|{):bb={:be=}:cb=/*:ce=*/:sb=":se=\e":\
     :le=\e':tl:\
     :kw=asm auto break case char continue default do double else enum\
     extern float for fortran goto if int long register return short\
     sizeof static struct switch typedef union unsigned void while #define\
     #else #endif #if #ifdef #ifndef #include #undef # define endif\
     ifdef ifndef include undef defined:
```

Note that the first field is just the language name (and any variants of it). Thus the C language could be specified to `vgrind`(1) as ‘c’ or ‘C’.

FILES

`/usr/lib/vgrindefs` file containing vgrind descriptions

SEE ALSO

`ex`(1), `lex`(1), `troff`(1), `vgrind`(1)

Index

A

ansi – standards and specifications supported by Solaris 298
architecture – characteristics of commands, utilities, and device drivers 16
ascii — ASCII character set 14
attributes – characteristics of commands, utilities, and device drivers 16
  Architecture 16
  Availability 16
  Interface Stability 17
  MT-Level 20
authentication, account, session, and password management PAM modules for Kerberos V5 — pam_krb5 250
availability – characteristics of commands, utilities, and device drivers 16

C

c – standards and specifications supported by Solaris 298
C – standards and specifications supported by Solaris 298
character set description file — charmap 24
characteristics of commands, utilities, and device drivers
  – architecture 16
  – attributes 16
  – availability 16
  – CSI 16
  – MT-Level 16
  – stability 16
charmap — character set description file 24
  Decimal Constants 26
  Declarations 24
  Format 25
  Ranges of Symbolic Names 26
  Symbolic Names 24
code set conversion tables — iconv_1250 92, 98, 106, 112, 115, 121, 127, 133, 141, 145, 153, 160, 164, 169
  — iconv_1250 92
  — iconv_1251 98
  — iconv_646 112
  — iconv_852 115
  — iconv_8859-1 121
  — iconv_8859-2 127
  — iconv_8859-5 133
  — iconv_dhn 141
  — iconv_koi8-r 145
  — iconv_mac_cyr 153
  — iconv_maz 160
  — iconv_pc_cyr 164
compilation environment, transitional — lfcompile64 188
CSI – characteristics of commands, utilities, and device drivers 16

D

document production
man — macros to format manual pages 220
mansun — macros to format manual pages 225
me — macros to format technical papers 229
mm — macros to format articles, theses and books 234
ms — macros to format articles, theses and books 241

E

environ — user environment 27
environment variables
HOME 27
LANG 27
LC_COLLATE 27
LC_CTYPE 27
LC_MESSAGES 27
LC_MONETARY 27
LC_NUMERIC 27
LC_TIME 27
MSGVERB 27
NETPATH 27
PATH 27
SEV_LEVEL 27
TERM 27
TZ 27
eqnchar — special character definitions for eqn 35
extensions — localedef extensions description file 36

F

file format notation — formats
formats 86
file name pattern matching — fnmatch 56
filesystem — file system organization 37
Root File System 38
/usr File System 45
fnmatch — file name pattern matching 56
fns — overview of FNS 60
Composite Names 60
FNS and Naming Systems 61
Why FNS? 60
XFN 60
FNS
overview — fns 60
overview of FNS References — fns_references 79
overview over DNS implementation — fns_dns 62
overview over files implementation — fns_files 65
overview over NIS (YP) implementation — fns_nis 73
overview over NIS+ implementation — fns_nis+ 71
overview over X.500 implementation — fns_x500 83
fns_dns — overview of FNS over DNS implementation 62
fns_files — overview of FNS over files implementation 65
FNS Policies and /etc Files 65
fns_initial_context — overview of the FNS Initial Context 67
fns_nis — overview of FNS over NIS (YP) implementation 73
Federating NIS with DNS or X.500 73
FNS Policies and NIS 73
NIS Security 73
fns_nis+ — overview of FNS over NIS+ implementation 71
FNS Policies and NIS+ 71
fns_policies — overview of the FNS Policies 75
fns_references — overview of FNS References 79
Address Types 80
Reference Types 79
fns_x500 — overview of FNS over X.500 implementation 83
formats — file format notation 86

I

iconv — code set conversion tables 106
iconv_1250 — code set conversion tables for MS 1250 (Windows Latin 2) 92

iconv_1251 — code set conversion tables for MS 1251 (Windows Cyrillic) 98
iconv_646 — code set conversion tables for ISO 646 112
iconv_852 — code set conversion tables for MS 852 (MS-DOS Latin 2) 115
iconv_8859-1 — code set conversion tables for ISO 8859-1 (Latin 1) 121
iconv_8859-2 — code set conversion tables for ISO 8859-2 (Latin 2) 127
iconv_8859-5 — code set conversion tables for ISO 8859-5 (Cyrillic) 133
iconv_dhn — code set conversion tables for DHN (Dom Handlowy Nauki) 141
iconv_koi8-r — code set conversion tables for KOI8-R 145
iconv_mac_cyr — code set conversion tables for Macintosh Cyrillic 153, 160
iconv_pc_cyr — code set conversion tables for Alternative PC Cyrillic 164
iconv_unicode — code set conversion tables for Unicode 169
internationalized basic and extended regular expression matching — regex 271
isalist — the native instruction sets known to Solaris 174
ISO – standards and specifications supported by Solaris 298

L

large file status of utilities — largefile 176
largefile — large file status of utilities 176
  Large file aware utilities 176
  Large file safe utilities 178
lf64 — transitional interfaces for 64-bit file offsets 179
  Data Types 179
  System Interfaces 181
lfcompile — large file compilation environment
  Access to Additional Large File Interfaces 185
lfcompile64 — transitional compilation environment 188
  Access to Additional Large File Interfaces 188
locale — subset of a user's environment that depends on language and cultural conventions 190
  collating-element keyword 200
  collating-symbol keyword 200
  Collation Order 202
  LC_COLLATE 199
  LC_CTYPE 193
  LC_MESSAGES 218
  LC_MONETARY 205
  LC_NUMERIC 210
  LC_TIME 211
  LC_TIME C-language Access 214
  LC_TIME General Information 217
  Locale Definition 190
  order_end keyword 205
  order_start keyword 201
localedef extensions description file — extensions 36

M

macros
  to format articles, theses and books — mm 234, 241
  to format Manual pages — man 220, 225
  to format technical papers — me 229
man — macros to format manual pages 220
mansun — macros to format manual pages 225
manual pages
  macros to format manual pages — man 220
  Sun macros to format manual pages — mansun 225
mark files for special treatment — sticky 303
me — macros to format technical papers 229
mm — macros to format articles, theses and books 234
ms — macros to format articles, theses and books 241
MT-Level – characteristics of commands, utilities, and device drivers 16

N

native instruction sets known to Solaris — isalist 174
NFS and sticky bits — sticky 303
nfssec — overview of NFS security modes 247

O

overview of FNS — fns 60
overview of FNS over DNS implementation — fns_dns 62
overview of FNS over files implementation — fns_files 65
overview of FNS over NIS (YP) implementation — fns_nis 73
overview of FNS over NIS+ implementation — fns_nis+ 71
overview of FNS over X.500 implementation — fns_x500 83
overview of FNS References — fns_references 79
overview of NFS security modes — nfssec 247
overview of the FNS Initial Context — fns_initial_context 67
overview of the FNS Policies — fns_policies 75

P

pam_dial_auth — authentication management for dialups 249
pam_krb5 — authentication, account, session and password management for Kerberos V5 250
pam_rhosts_auth — authentication management using ruserok() 257
pam_sample — sample module for PAM 260
pam_unix — authentication, account, session and password management for UNIX 264
POSIX – standards and specifications supported by Solaris 298
POSIX.1 – standards and specifications supported by Solaris 298
posix.2 – standards and specifications supported by Solaris 298
profiling utilities
  profile within a function — prof 267

R

rbac — role-based access control 268
regex — internationalized basic and extended regular expression matching 271
regular expression compile and match routines
  – advance 281
  – compile 281
  – regexp 281
  – step 281
role-based access control — rbac 268

S

ftp — authentication system 289
sgml – Standard Generalized Markup Language 291
  RefEntry 291
  RefMeta 291
  RefNameDiv 292
  RefSect1 293
  RefSect2 293
  RefSynopsisDiv 293
shell environment
  conventional names for terminals — term 304
solbook – Standard Generalized Markup Language 291
special character definitions for eqn — eqnchar 35
stability – characteristics of commands, utilities, and device drivers 16
Standard Generalized Markup Language
  – sgml 291
  – solbook 291
standards – standards and specifications supported by Solaris 298
standards and specifications supported by Solaris
  – c 298
  – C 298
  – ansi 298
  – ISO 298
  – posix 298

– posix.1 298
– posix.2 298
– standards 298
– sus 298
– susv2 298
– svid 298
– SVID3 298
– XNS 298
– XNS4 298
– XNS5 298
– XPG 298
– XPG3 298
– xpg4 298
– xpg4v2 298
sticky — mark files for special treatment 303
subset of a user's environment that depends on language and cultural conventions — locale 190
sus – standards and specifications supported by Solaris 298
susv2 – standards and specifications supported by Solaris 298
svid – standards and specifications supported by Solaris 298
svid3 – standards and specifications supported by Solaris 298

T

term — conventional names for terminals 304
terminals
conventional names — term 304
transitional compilation environment — lfcompile64 188
transitional interfaces for 64-bit file offsets — lf64 179

U

unicode
code set conversion tables — iconv_unicode 169
user environment
— environ 27

V

vgrindefs — vgrind language definitions 308

X

XNS – standards and specifications supported by Solaris 298
XNS4 – standards and specifications supported by Solaris 298
XNS5 – standards and specifications supported by Solaris 298
xpg – standards and specifications supported by Solaris 298
xpg3 – standards and specifications supported by Solaris 298
xpg4 – standards and specifications supported by Solaris 298
xpg4v2 – standards and specifications supported by Solaris 298