```
**********************************************************
    6739 Tue Dec 11 09:48:08 2018
new/usr/src/man/man1/pmadvise.1
10057 Man page misspellings ouput particuliar overriden
Reviewed by: Gerg¯  Mih^¡ly Doma <domag02@gmail.com>
**********************************************************
```

```
'\" te
.\" Copyright (c) 2008, Sun Microsystems, Inc. All Rights Reserved.
.\" The contents of this file are subject to the terms of the Common Development
.\" You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE or http:
.\" When distributing Covered Code, include this CDDL HEADER in each file and in
.TH PMADVISE 1 "Sep 25, 2008"
.SH NAME
pmadvise \- applies advice about memory to a process
.SH SYNOPSIS
.LP
.nf
\fBpmadvise\fR \fB-o\fR \fIoption\fR[,\fIoption\fR] [\fB-F\fR] [\fB-l\fR] [\fB-v
.fi

.SH DESCRIPTION
.LP
\fBpmadvise\fR applies advice about how memory is used in the specified process
using \fBmadvise\fR(3C).
.sp
.LP
\fBpmadvise\fR allows users to apply advice to a specific sub-range at a
specific instant in time. \fBpmadvise\fR differs from \fBmadv.so.1\fR(1) in
that \fBmadv.so.1\fR(1) applies the advice throughout execution of the target
program to all segments of a specified type.
.SH OPTIONS
.LP
The following options are supported:
.sp
.ne 2
.na
\fB\fB-F\fR\fR
.ad
.RS 6n
Force by grabbing the target process even if another process has control.
.sp
You should exercise caution when using the \fB-F\fR option. See \fBproc\fR(1).
.RE

.sp
.ne 2
.na
\fB\fB-l\fR\fR
.ad
.RS 6n
Show unresolved dynamic linker map names.
.RE

.sp
.ne 2
.na
\fB\fB-o\fR\fR
.ad
.RS 6n
Specify advice to apply in the following form:
.sp
.in +2
.nf
private=\fBadvice\fR
shared=\fBadvice\fR
heap=\fBadvice\fR
```

```
stack=\fBadvice\fR
\fBaddress\fR[:\fBlength\fR]=\fBadvice\fR
.fi
.in -2
.sp

where the \fBadvice\fR can be one of the following:
.sp
.in +2
.nf
normal
random
sequential
willneed
dontneed
free
access_lwp
access_many
access_default
purge
.fi
.in -2
.sp

An \fBaddress\fR and \fBlength\fR can be given to specify a subrange to apply
the advice.The \fBaddress\fR should be hexadecimal and the \fBlength\fR should
be in bytes by default.
.sp
If \fBlength\fR is not specified and the starting address refers to the start
of a segment, the advice is applied to that segment. \fBlength\fR can be
qualified by \fBK\fR, \fBM\fR, \fBG\fR, \fBT\fR, \fBP\fR, or \fBE\fR to specify
kilobytes, megabytes, gigabytes, terabytes, or exabytes respectively as the
unit of measure.
.RE

.sp
.ne 2
.na
\fB\fB-v\fR\fR
.ad
.RS 6n
Print verbose output. Display output as \fBpmap\fR(1) does, showing what advice
is being applied where. This can be useful when the advice is being applied to
a named region (for example, private, shared, and so forth) to get feedback on
exactly where the advice is being applied.
.RE

.sp
.LP
\fBpmadvise\fR tries to process all legal options. If an illegal address range
is specified, an error message is printed and the offending option is skipped.
\fBpmadvise\fR quits without processing any options and prints a usage message
when there is a syntax error.
.sp
.LP
If conflicting advice is given on a region, the order of precedence is from
most specific advice to least, that is, most general. In other words, advice
specified for a particular address range takes precedence over advice for heap
specified for a particuliar address range takes precedence over advice for heap
and stack which in turn takes precedence over advice for private and shared
memory.
.sp
.LP
Moreover, the advice in each of the following groups are mutually exclusive
from the other advice within the same group:
.sp
```

```
.in +2
.nf
MADV_NORMAL, MADV_RANDOM, MADV_SEQUENTIAL
MADV_WILLNEED, MADV_DONTNEED, MADV_FREE, MADV_PURGE
MADV_ACCESS_DEFAULT, MADV_ACCESS_LWP, MADV_ACCESS_MANY
.fi
.in -2
.sp

.SH OPERANDS
.LP
The following operands are supported:
.sp
.ne 2
.na
\fB\fIpid\fR\fR
.ad
.RS 7n
Process ID.
.RE

.SH EXAMPLES
\fBExample 1 \fRApplying Advice to a Segment at Specified Address
.sp
.LP
The following example applies advice to a segment at a specified address:

.sp
.in +2
.nf
% pmap $$
100666: tcsh
00010000     312K r-x--  /usr/bin/tcsh
0006C000      48K rwx--  /usr/bin/tcsh
00078000     536K rwx--    [ heap ]
FF100000     856K r-x--  /lib/libc.so.1
FF1E6000      32K rwx--  /lib/libc.so.1
FF1EE000       8K rwx--  /lib/libc.so.1
FF230000     168K r-x--  /lib/libcurses.so.1
FF26A000      32K rwx--  /lib/libcurses.so.1
FF272000       8K rwx--  /lib/libcurses.so.1
FF280000     576K r-x--  /lib/libnsl.so.1
FF310000      40K rwx--  /lib/libnsl.so.1
FF31A000      24K rwx--  /lib/libnsl.so.1
FF364000       8K rwxs-    [ anon ]
FF370000      48K r-x--  /lib/libsocket.so.1
FF38C000       8K rwx--  /lib/libsocket.so.1
FF3A0000       8K r-x--  /platform/sun4u-us3/lib/libc_psr.so.1
FF3B0000     176K r-x--  /lib/ld.so.1
FF3EC000       8K rwx--  /lib/ld.so.1
FF3EE000       8K rwx--  /lib/ld.so.1
FFBE6000     104K rw---    [ stack ]
%
% pmadvise -o 78000=access_lwp $$

%
.fi
.in -2
.sp

.LP
\fBExample 2 \fRUsing the \fB-v\fR Option
.sp
.LP
The following example displays verbose output from \fBpmadvise\fR:
```

```
.sp
.in +2
.nf

% pmadvise -o heap=access_lwp,stack=access_default -v $$
1720:   -sh
00010000      88K r-x--  /sbin/sh
00036000       8K rwx--  /sbin/sh
00038000      16K rwx--    [ heap ]          <= access_lwp
FF250000      24K r-x--  /lib/libgen.so.1
FF266000       8K rwx--  /lib/libgen.so.1
FF272000       8K rwxs-    [ anon ]
FF280000     840K r-x--  /lib/libc.so.1
FF362000      32K rwx--  /lib/libc.so.1
FF36A000      16K rwx--  /lib/libc.so.1
FF380000       8K r-x--  /platform/sun4u-us3/lib/libc_psr.so.1
FF390000      64K rwx--    [ anon ]
FF3B0000     168K r-x--  /lib/ld.so.1
FF3EA000       8K rwx--  /lib/ld.so.1
FF3EC000       8K rwx--  /lib/ld.so.1
FFBFE000       8K rw---    [ stack ]         <= access_default
.fi
.in -2
.sp

.SH EXIT STATUS
.LP
The following exit values are returned:
.sp
.ne 2
.na
\fB\fB0\fR\fR
.ad
.RS 12n
Successful completion.
.RE

.sp
.ne 2
.na
\fB\fBnon-zero\fR\fR
.ad
.RS 12n
An error occurred.
.RE

.SH FILES
.ne 2
.na
\fB\fB/proc/*\fR\fR
.ad
.RS 19n
Process files
.RE

.sp
.ne 2
.na
\fB\fB/usr/prob/lib/*\fR\fR
.ad
.RS 19n
\fBproc\fR tools support files
.RE

.SH ATTRIBUTES
.LP
```

```
See \fBattributes\fR(5) for descriptions of the following attributes:
.sp

.sp
.TS
box;
c | c
l | l .
ATTRIBUTE TYPE  ATTRIBUTE VALUE
_
Interface Stability     See below.
.TE

.sp
.LP
The command syntax is Evolving. The output formats are Unstable.
.SH SEE ALSO
.LP
\fBmadv.so.1\fR(1), \fBpmap\fR(1), \fBproc\fR(1), \fBmadvise\fR(3C),
\fBattributes\fR(5)
```

```
**********************************************************
   18942 Tue Dec 11 09:48:08 2018
new/usr/src/man/man1m/fsdb_udfs.1m
10057 Man page misspellings ouput particuliar overriden
Reviewed by: Gerg¯  Mih^¡ly Doma <domag02@gmail.com>
**********************************************************
'\" te
.\"  Copyright (c) 1999 Sun Microsystems, Inc. All Rights Reserved.
.\" The contents of this file are subject to the terms of the Common Development
.\" You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE or http:
.\" When distributing Covered Code, include this CDDL HEADER in each file and in
.TH FSDB_UDFS 1M "Nov 26, 2017"
.SH NAME
fsdb_udfs \- udfs file system debugger
.SH SYNOPSIS
.LP
.nf
\fBfsdb \fR [\fB-F\fR] udfs [\fIgeneric_option\fR] [\fB-o \fR\fIspecific_option\
.fi

.SH DESCRIPTION
.LP
The \fBfsdb_udfs\fR command is an interactive tool that can be used to patch up
a damaged \fBudfs\fR file system. \fBfsdb_udfs\fR has conversions to translate
block and i-numbers into their corresponding disk addresses. Mnemonic offsets
to access different parts of an inode are also included. Mnemonic offsets
greatly simplify the process of correcting control block entries or descending
the file system tree.
.sp
.LP
\fBfsdb\fR contains several error-checking routines to verify inode and block
addresses. These can be disabled if necessary by invoking \fBfsdb\fR with the
\fB-o\fR option or by using the \fBo\fR command.
.sp
.LP
\fBfsdb\fR reads one block at a time, and therefore works with raw as well as
block \fBI/O\fR devices. A buffer management routine is used to retain commonly
used blocks of data in order to reduce the number of read system calls. All
assignment operations result in an immediate write-through of the corresponding
block. In order to modify any portion of the disk, \fBfsdb\fR must be invoked
with the \fB-w\fR option.
.sp
.LP
Wherever possible, \fBadb\fR-like syntax has been adopted to promote the use of
\fBfsdb\fR through familiarity.
.SH OPTIONS
.LP
The following options are supported:
.sp
.ne 2
.na
\fB\fB-o\fR \fIspecific_option\fR\fR
.ad
.RS 22n
Specify \fBudfs\fR file system specific options in a comma-separated list with
no intervening spaces. The following specific options are supported:
.sp
.ne 2
.na
\fBo\fR
.ad
.RS 12n
Override some error conditions.
.RE

.sp
```

```
.ne 2
.na
\fBp=\fIstring\fR\fR
.ad
.RS 12n
Set prompt to \fIstring\fR.
.RE

.sp
.ne 2
.na
\fBw\fR
.ad
.RS 12n
Open for write.
.RE

.sp
.ne 2
.na
\fB?\fR
.ad
.RS 12n
Display usage.
.RE

.RE

.SH USAGE
.LP
Numbers are considered hexadecimal by default. The user has control over how
data is to be displayed or accepted. The \fBbase\fR command displays or sets
the input and output base. Once set, all input defaults to this base and all
output displays in this base. The base can be overridden temporarily for input
output displays in this base. The base can be overriden temporarily for input
by preceding hexadecimal numbers by \fB0x\fR, preceding decimal numbers with a
\fB0t\fR, or octal numbers with a \fB0\fR. Hexadecimal numbers beginning with
\fBa\fR-\fBf\fR or \fBA\fR-\fBF\fR must be preceded with a \fB0x\fR to
\fBa\fR-\fBf\fR or \fBA\fR -\fBF\fR must be preceded with a \fB0x\fR to
distinguish them from commands.
.sp
.LP
Disk addressing by \fBfsdb\fR is at the byte level. However, \fBfsdb\fR offers
many commands to convert a desired inode, directory entry, block, and so forth,
to a byte address. After the address has been calculated, \fBfsdb\fR records
the result in the current address (\fBdot\fR).
.sp
.LP
Several global values are maintained by \fBfsdb\fR\fB\fR:
.RS +4
.TP
.ie t \(bu
.el o
Current base (referred to as \fBbase\fR)
.RE
.RS +4
.TP
.ie t \(bu
.el o
Current address (referred to as \fBdot\fR)
.RE
.RS +4
.TP
.ie t \(bu
.el o
Current inode (referred to as \fBinode\fR)
```

```
.RE
.RS +4
.TP
.ie t \(bu
.el o
Current count (referred to as \fBcount\fR)
.RE
.RS +4
.TP
.ie t \(bu
.el o
Current type (referred to as \fBtype\fR)
.RE
.sp
.LP
Most commands use the preset value of \fBdot\fR in their execution. For
example,
.sp
.in +2
.nf
 > 2:inode
.fi
.in -2
.sp

.sp
.LP
first sets the value of dot (\fB\&.\fR) to \fB2\fR, colon (\fB:\fR), signifies
the start of a command, and the \fBinode\fR command sets \fBinode\fR to
\fB2\fR. A count is specified after a comma (\fB,\fR). Once set, count remains
at this value until a new command is encountered that resets the value back to
\fB1\fR (the default).
.sp
.LP
So, if
.sp
.in +2
.nf
> 2000,400/X
.fi
.in -2
.sp

.sp
.LP
is entered, \fB400\fR hex longs are listed from \fB2000\fR, and when completed,
the value of dot is \fB 2000 + 400 * sizeof\fR (long). If a RETURN is then
entered, the output routine uses the current values of \fBdot\fR, \fBcount\fR,
and \fBtype\fR and displays \fB400 \fRmore hex longs. An asterisk (\fB*\fR)
causes the entire block to be displayed. An example showing several commands
and the use of RETURN would be:
.sp
.in +2
.nf
> 2:ino; 0:dir?d
.fi
.in -2
.sp

.sp
.LP
or
.sp
.in +2
.nf
> 2:ino; 0:db:block?d
```

```
.fi
.in -2
.sp

.sp
.LP
The two examples are synonymous for getting to the first directory entry of the
root of the file system. Once there, subsequently entering a RETURN, plus
(\fB+\fR), or minus (\fB-\fR)  advances to subsequent entries. Notice that
.sp
.in +2
.nf
> 2:inode; :ls
.fi
.in -2
.sp

.sp
.LP
or
.sp
.in +2
.nf
> :ls /
.fi
.in -2
.sp

.sp
.LP
is again synonymous.
.SS "Expressions"
.LP
The following symbols are recognized by \fBfsdb\fR:
.sp
.ne 2
.na
\fBRETURN\fR
.ad
.RS 13n
Update the value of dot by the current value of \fItype\fR and \fIdisplay\fR
using the current value of \fIcount\fR.
.RE

.sp
.ne 2
.na
\fB\fI#\fR\fR
.ad
.RS 13n
Update the value of dot by specifying a numeric expression. Specify numeric
expressions using addition, subtraction, multiplication, and division
operators ( \fB+\fR, \fB-\fR, \fB*\fR, and \fB%\fR). Numeric expressions are
evaluated from left to right and can use parentheses. After evaluation, the
value of dot is updated.
.RE

.sp
.ne 2
.na
\fB, \fIcount\fR\fR
.ad
.RS 13n
Update the count indicator. The global value of \fIcount\fR is updated to
\fIcount\fR. The value of \fIcount\fR remains until a new command is run. A
\fIcount\fR specifier of \fB*\fR attempts to show a blocks’s worth of
```

```
information. The default for \fIcount\fR is \fB1\fR.
.RE

.sp
.ne 2
.na
\fB? \fI f\fR\fR
.ad
.RS 13n
Display in structured style with format specifier \fBf\fR. See \fBFormatted
Output\fR.
.RE

.sp
.ne 2
.na
\fB/ \fIf\fR\fR
.ad
.RS 13n
Display in unstructured style with format specifier \fBf\fR. See \fBFormatted
Output\fR.
.RE

.sp
.ne 2
.na
\fB\&.\fR
.ad
.RS 13n
Display the value of dot.
.RE

.sp
.ne 2
.na
\fB+\fIe\fR\fR
.ad
.RS 13n
Increment the value of dot by the expression \fIe\fR. The amount actually
incremented is dependent on the size of type: \fBdot = dot + \fR\fIe \fR\fB*
\fR\fBsizeof\fR (\fItype\fR) The default for \fIe\fR is \fB1\fR.
.RE

.sp
.ne 2
.na
\fB\(mi\fIe\fR\fR
.ad
.RS 13n
Decrement the value of dot by the expression \fIe \fR. See \fB+\fR.
.RE

.sp
.ne 2
.na
\fB*\fIe\fR\fR
.ad
.RS 13n
Multiply the value of dot by the expression \fIe\fR. Multiplication and
division don’t use \fItype\fR. In the above calculation of dot, consider the
\fBsizeof\fR (\fItype\fR) to be \fB1\fR.
.RE

.sp
.ne 2
.na
```

```
\fB%\fIe\fR\fR
.ad
.RS 13n
Divide the value of dot by the expression \fIe\fR. See \fB*\fR.
.RE

.sp
.ne 2
.na
\fB< \fIname\fR\fR
.ad
.RS 13n
Restore an address saved in register \fIname\fR. \fIname\fR must be a single
letter or digit.
.RE

.sp
.ne 2
.na
\fB> \fIname\fR\fR
.ad
.RS 13n
Save an address in register \fIname\fR. \fIname\fR must be a single letter or
digit.
.RE

.sp
.ne 2
.na
\fB= \fIf\fR\fR
.ad
.RS 13n
Display indicator. If \fIf\fR is a legitimate format specifier (see
\fBFormatted Output\fR), then the value of dot is displayed using format
specifier \fIf\fR. Otherwise, assignment is assumed. See \fB= [s] [e]\fR.
.RE

.sp
.ne 2
.na
\fB= [\fIs\fR] [\fIe\fR]\fR
.ad
.RS 13n
Change the value of dot using an assignment indicator. The address pointed to
by dot has its contents changed to the value of the expression \fIe\fR or to
the \fBASCII\fR representation of the quoted (\fB"\fR) string \fIs\fR. This can
be useful for changing directory names or \fBASCII\fR file information.
.RE

.sp
.ne 2
.na
\fB=+ \fIe\fR\fR
.ad
.RS 13n
Change the value of dot using an incremental assignment. The address pointed to
by dot has its contents incremented by expression \fIe\fR.
.RE

.sp
.ne 2
.na
\fB=- e\fR
.ad
.RS 13n
Change the value of dot using a decremental assignment. Decrement the contents
```

```
of the address pointed to by dot by expression \fIe\fR.
.RE

.SS "Commands"
.LP
A command must be prefixed by a colon (\fB:\fR). Only enough letters of the
command to uniquely distinguish it are needed. Multiple commands can be entered
on one line by  separating them by a SPACE, TAB, or semicolon (\fB;\fR).
.sp
.LP
To view a potentially unmounted disk in a reasonable manner, \fBfsdb\fR
supports the \fBcd\fR, \fBpwd\fR, \fBls\fR, and \fBfind\fR commands. The
functionality of each of these commands basically matches that of its UNIX
counterpart. See \fBcd\fR(1), \fBpwd\fR(1), \fBls\fR(1), and \fBfind\fR(1) for
details. The \fB*\fR, \fB,\fR, \fB?\fR, and \fB-\fR wildcard characters are
also supported.
.sp
.LP
The following commands are supported:
.sp
.ne 2
.na
\fBbase[=\fIb\fR]\fR
.ad
.sp .6
.RS 4n
Display or set the base. All input and output is governed by the current base.
Without the \fB=\fR \fIb\fR, displays the current base. Otherwise, sets the
current base to \fIb\fR. Base is interpreted using the old value of base, so to
ensure correctness use the \fB0\fR, \fB0t\fR, or \fB0x\fR prefix when changing
the base. The default for base is hexadecimal.
.RE

.sp
.ne 2
.na
\fBblock\fR
.ad
.sp .6
.RS 4n
Convert the value of dot to a block address.
.RE

.sp
.ne 2
.na
\fBcd [\fIdir\fR]\fR
.ad
.sp .6
.RS 4n
Change the current directory to directory  \fIdir\fR. The current values of
inode and dot are also updated. If \fBdir\fR is not specified, changes
directories to inode 2, root (\fB/\fR).
.RE

.sp
.ne 2
.na
\fBdirectory\fR
.ad
.sp .6
.RS 4n
If the current inode is a directory, converts the value of dot to a directory
slot offset in that directory, and dot now points to this entry.
.RE
```

```
.sp
.ne 2
.na
\fBfile\fR
.ad
.sp .6
.RS 4n
Set the value of dot as a relative block count from the beginning of the file.
The value of dot is updated to the first byte of  this block.
.RE

.sp
.ne 2
.na
\fBfind \fIdir\fR [\fB-name\fR \fIn\fR] | [\fB-inum\fR\fI i\fR]\fR
.ad
.sp .6
.RS 4n
Find files by name or i-number. Recursively searches directory \fIdir\fR and
below for file names whose i-number matches \fBi\fR or whose name matches
below for file names whose i-number matches\fB i\fR or whose name matches
pattern \fIn\fR. Only one of the two options (\fB-name\fR or \fB-inum\fR) can
be used at one time. The find \fB-print\fR is not necessary or accepted.
.RE

.sp
.ne 2
.na
\fBfill=\fIp\fR\fR
.ad
.sp .6
.RS 4n
Fill an area of disk with pattern \fIp\fR. The area of disk is delimited by dot
and count.
.RE

.sp
.ne 2
.na
\fBinode\fR
.ad
.sp .6
.RS 4n
Convert the value of dot to an inode address. If successful, the current value
of inode is updated as well as the value of dot. As a convenient shorthand, if
\fB:inode\fR appears at the beginning of the line, the value of dot is set to
the current inode and that inode is displayed in inode format.
.RE

.sp
.ne 2
.na
\fBls [ \fB-R\fR ] [\fB-l\fR ] \fIpat1\fR \fIpat2\fR...\fR
.ad
.sp .6
.RS 4n
List directories or files. If no file is specified, the current directory is
assumed. Either or both of the options can be used (but, if used, must be
specified before the filename specifiers). Wild card characters are available
and multiple arguments are acceptable. The long listing shows only the i-number
and the name; use the inode command with \fB?i\fR to get more information.
.RE

.sp
.ne 2
.na
```

```
\fBoverride\fR
.ad
.sp .6
.RS 4n
Toggle the value of override. Some error conditions might be overridden if
override is toggled to \fBon\fR.
.RE

.sp
.ne 2
.na
\fBprompt "\fIp\fR"\fR
.ad
.sp .6
.RS 4n
Change the \fBfsdb\fR prompt to \fIp\fR. \fIp\fR must be enclosed in quotes.
.RE

.sp
.ne 2
.na
\fBpwd\fR
.ad
.sp .6
.RS 4n
Display the current working directory.
.RE

.sp
.ne 2
.na
\fBquit\fR
.ad
.sp .6
.RS 4n
Quit \fBfsdb\fR.
.RE

.sp
.ne 2
.na
\fBtag\fR
.ad
.sp .6
.RS 4n
Convert the value of dot and if this is a valid tag, print the volume structure
according to the tag.
.RE

.sp
.ne 2
.na
\fB!\fR
.ad
.sp .6
.RS 4n
Escape to the shell.
.RE

.SS "Inode Commands"
.LP
In addition to the above commands, several other commands deal with inode
fields and operate directly on the current inode (they still require the colon
(\fB:\fR). They can be used to more easily display or change the particular
fields. The value of dot is only used by the \fB:db\fR and \fB:ib\fR commands.
Upon completion of the command, the value of dot is changed so that it points
```

```
to that particular field. For example,
.sp
.in +2
.nf
> :ln=+1
.fi
.in -2
.sp

.sp
.LP
increments the link count of the current inode and sets the value of dot to the
address of the link count field.
.sp
.LP
The following inode commands are supported:
.sp
.ne 2
.na
\fBat\fR
.ad
.RS 8n
Access time
.RE

.sp
.ne 2
.na
\fBbs\fR
.ad
.RS 8n
Block size
.RE

.sp
.ne 2
.na
\fBct\fR
.ad
.RS 8n
Creation time
.RE

.sp
.ne 2
.na
\fBgid\fR
.ad
.RS 8n
Group id
.RE

.sp
.ne 2
.na
\fBln\fR
.ad
.RS 8n
Link number
.RE

.sp
.ne 2
.na
\fBmt\fR
.ad
```

```
.RS 8n
Modification time
.RE

.sp
.ne 2
.na
\fBmd\fR
.ad
.RS 8n
Mode
.RE

.sp
.ne 2
.na
\fBmaj\fR
.ad
.RS 8n
Major device number
.RE

.sp
.ne 2
.na
\fBmin\fR
.ad
.RS 8n
Minor device number
.RE

.sp
.ne 2
.na
\fBnm\fR
.ad
.RS 8n
This command actually operates on the directory name field. Once      poised at
the desired directory entry (using the \fBdirectory\fR command), this command
allows you to change or display the directory name. For example,
.sp
.in +2
.nf
 > 7:dir:nm="foo"
.fi
.in -2
.sp

gets the \fB7\fRth directory entry of the current inode and changes its name to
\fBfoo\fR. Directory names cannot be made larger than the field allows. If an
attempt is made to make a directory name larger than the field allows,, the
string is truncated to fit and a warning message is displayed.
.RE

.sp
.ne 2
.na
\fBsz\fR
.ad
.RS 8n
File size
.RE

.sp
.ne 2
.na
```

```
\fBuid\fR
.ad
.RS 8n
User \fBID\fR
.RE

.sp
.ne 2
.na
\fBuniq\fR
.ad
.RS 8n
Unique \fBID\fR
.RE

.SS "Formatted Output"
.LP
Formatted output comes in two styles and many format types. The two styles of
formatted output are: structured and unstructured. Structured output is used to
display inodes, directories, and so forth. Unstructured output displays raw
data.
.sp
.LP
Format specifiers are preceded by the slash (\fB/\fR) or question mark
(\fB?\fR) character. \fItype\fR is updated as necessary upon completion.
.sp
.LP
The following format specifiers are preceded by the \fB?\fR character:
.sp
.ne 2
.na
\fBi\fR
.ad
.RS 5n
Display as inodes in the current base.
.RE

.sp
.ne 2
.na
\fBd\fR
.ad
.RS 5n
Display as directories in the current base.
.RE

.sp
.LP
The following format specifiers are preceded by the \fB/\fR character:
.sp
.ne 2
.na
\fBb\fR
.ad
.RS 9n
Display as bytes in the current base.
.RE

.sp
.ne 2
.na
\fBc\fR
.ad
.RS 9n
Display as characters.
.RE
```

```
.sp
.ne 2
.na
\fBo | O\fR
.ad
.RS 9n
Display as octal shorts or longs.
.RE

.sp
.ne 2
.na
\fBd | D\fR
.ad
.RS 9n
Display as decimal shorts or longs.
.RE

.sp
.ne 2
.na
\fBx | X\fR
.ad
.RS 9n
Display as hexadecimal shorts or longs.
.RE

.SH EXAMPLES
.LP
\fBExample 1 \fRUsing fsdb as a calculator for complex arithmetic
.sp
.LP
The following command displays \fB2010\fR in decimal format, and is an example
of using \fBfsdb\fR as a calculator for complex arithmetic.

.sp
.in +2
.nf
> 2000+400%(20+20)=D
.fi
.in -2
.sp

.LP
\fBExample 2 \fRUsing fsdb to display an i-number in inode fomat
\fBExample 2 \fRUsing fsdb to display an i-number in idode fomat
.sp
.LP
The following command displays the i-number \fB386\fR in inode format.\fB386\fR
becomes the current inode.

.sp
.in +2
.nf
> 386:ino?i
.fi
.in -2
.sp

.LP
\fBExample 3 \fRUsing fsdb to change the link count
.sp
.LP
The following command changes the link count for the current inode to \fB4\fR.
```

```
.sp
.in +2
.nf
> :ln=4
.fi
.in -2
.sp

.LP
\fBExample 4 \fRUsing fsdb to increment the link count
.sp
.LP
The following command increments the link count by \fB1\fR.

.sp
.in +2
.nf
> :ln=+1
.fi
.in -2
.sp

.LP
\fBExample 5 \fRUsing fsdb to display the creation time as a hexadecimal long
.sp
.LP
The following command displays the creation time as a hexadecimal long.

.sp
.in +2
.nf
> :ct=X
.fi
.in -2
.sp

.LP
\fBExample 6 \fRUsing fsdb to display the modification time in time format
.sp
.LP
The following command displays the modification time in time format.

.sp
.in +2
.nf
> :mt=t
.fi
.in -2
.sp

.LP
\fBExample 7 \fRUsing fsdb to display in ASCII
.sp
.LP
The following command displays, in \fBASCII\fR, block \fB0\fR of the file
associated with the current inode.

.sp
.in +2
.nf
> 0:file/c
.fi
.in -2
.sp

.LP
```

```
\fBExample 8 \fRUsing fsdb to display the directory enteries for the root inode
.sp
.LP
The following command displays the first block’s directory entries for the root
inode of this file system. This command stops prematurely if the \fBEOF\fR is
reached.

.sp
.in +2
.nf
> 2:ino,*?d
.fi
.in -2
.sp

.LP
\fBExample 9 \fRUsing fsdb to change the current inode
.sp
.LP
 The following command changes the current inode to that associated with the
\fB5\fRth directory entry (numbered from \fB0\fR) of the current inode. The
first logical block of the file is then displayed in \fBASCII\fR.

.sp
.in +2
.nf
> 5:dir:inode; 0:file,*/c
.fi
.in -2
.sp

.LP
\fBExample 10 \fRUsing fsdb to change the i-number
.sp
.LP
 The following command changes the i-number for the \fB7\fRth directory slot in
the root directory to \fB3\fR.

.sp
.in +2
.nf
> 2:inode; 7:dir=3
.fi
.in -2
.sp

.LP
\fBExample 11 \fRUsing fsdb to change the name field
.sp
.LP
The following command changes the \fIname\fR field in the directory slot to
\fBname\fR.

.sp
.in +2
.nf
 > 7:dir:nm="name"
.fi
.in -2
.sp

.LP
\fBExample 12 \fRUsing fsdb to display the a block
.sp
.LP
The following command displays the \fB3\fRrd block of the current inode as
```

```
directory entries.

.LP
\fBExample 13 \fRUsing fsdb to set the contents of address
.sp
.LP
The following command sets the contents of address \fB2050\fR to
\fB0xffffffff\fR. \fB0xffffffff\fR can be truncated, depending on the current
type.

.sp
.in +2
.nf
> 2050=0xffff
.fi
.in -2
.sp

.LP
\fBExample 14 \fRUsing fsdb to place an ASCII string at an address
.sp
.LP
The following command places the \fBASCII\fR string \fBthis is some text\fR at
address \fB1c92434\fR.

.sp
.in +2
.nf
> 1c92434="this is some text"
.fi
.in -2
.sp

.SH SEE ALSO
.LP
\fBclri\fR(1M), \fBfsck_udfs\fR(1M), \fBdir\fR(4),  \fBattributes\fR(5)
```

```
**********************************************************
   22547 Tue Dec 11 09:48:08 2018
new/usr/src/man/man1m/fsdb_ufs.1m
10057 Man page misspellings ouput particuliar overriden
Reviewed by: Gerg¯  Mih^¡ly Doma <domag02@gmail.com>
**********************************************************
'\" te
.\"  Copyright (c) 1988 Regents of the University
.\" of California.  All rights reserved.  Copyright (c) 2003 Sun Microsystems,
.\" Inc.  All Rights Reserved.
.TH FSDB_UFS 1M "Apr 14, 2003"
.SH NAME
fsdb_ufs \- ufs file system debugger
.SH SYNOPSIS
.LP
.nf
\fBfsdb\fR \fB-F\fR ufs [\fIgeneric_options\fR] [\fIspecific_options\fR] \fIspec
.fi

.SH DESCRIPTION
.sp
.LP
The \fBfsdb_ufs\fR command is an interactive tool that can be used to patch up
a damaged \fBUFS\fR file system. It has conversions to translate block and
i-numbers into their corresponding disk addresses. Also included are mnemonic
offsets to access different parts of an inode. These greatly simplify the
process of correcting control block entries or descending the file system tree.
.sp
.LP
\fBfsdb\fR contains several error-checking routines to verify inode and block
addresses. These can be disabled if necessary by invoking \fBfsdb\fR with the
\fB-o\fR option or by the use of the \fBo\fR command.
.sp
.LP
\fBfsdb\fR reads a block at a time and will therefore work with raw as well as
block \fBI/O\fR devices. A buffer management routine is used to retain commonly
used blocks of data in order to reduce the number of read system calls. All
assignment operations result in an immediate write-through of the corresponding
block. Note that in order to modify any portion of the disk, \fBfsdb\fR must be
invoked with the \fBw\fR option.
.sp
.LP
Wherever possible, \fBadb-\fRlike syntax was adopted to promote the use of
\fBfsdb\fR through familiarity.
.SH OPTIONS
.sp
.LP
The following option is supported:
.sp
.ne 2
.na
\fB\fB-o\fR\fR
.ad
.RS 6n
Specify \fBUFS\fR file system specific options. These options can be any
combination of the following separated by commas (with no intervening spaces).
The options available are:
.sp
.ne 2
.na
\fB\fB?\fR\fR
.ad
.RS 14n
Display usage
.RE
```

```
.sp
.ne 2
.na
\fB\fBo\fR\fR
.ad
.RS 14n
Override some error conditions
.RE

.sp
.ne 2
.na
\fB\fBp='string'\fR\fR
.ad
.RS 14n
set prompt to string
.RE

.sp
.ne 2
.na
\fB\fBw\fR\fR
.ad
.RS 14n
open for write
.RE

.RE

.SH USAGE
.sp
.LP
Numbers are considered hexadecimal by default. However, the user has control
over how data is to be displayed or accepted. The \fBbase\fR command will
display or set the input/output base. Once set, all input will default to this
base and all output will be shown in this base. The base can be overridden
temporarily for input by preceding hexadecimal numbers with \&'\fB0x\fR',
preceding decimal numbers with '\fB0t\fR', or octal numbers with '\fB0\fR'.
Hexadecimal numbers beginning with \fBa-f\fR or \fBA-F\fR must be preceded with
\&'\fB0x\fR' to distinguish them from commands.
.sp
.LP
Disk addressing by \fBfsdb\fR is at the byte level. However, \fBfsdb\fR offers
many commands to convert a desired inode, directory entry, block, superblock
and so forth to a byte address. Once the address has been calculated,
\fBfsdb\fR will record the result in dot (\fB\&.\fR).
.sp
.LP
Several global values are maintained by \fBfsdb\fR:
.RS +4
.TP
.ie t \(bu
.el o
the current base (referred to as \fBbase\fR),
.RE
.RS +4
.TP
.ie t \(bu
.el o
the current address (referred to as \fBdot\fR),
.RE
.RS +4
.TP
.ie t \(bu
.el o
the current inode (referred to as \fBinode\fR),
.RE
.RS +4
.TP
.ie t \(bu
.el o
the current count (referred to as \fBcount\fR),
.RE
.RS +4
.TP
.ie t \(bu
.el o
and the current type (referred to as \fBtype\fR).
.RE
.sp
.LP
Most commands use the preset value of \fBdot\fR in their execution. For
example,
.sp
.LP
\fB> 2:inode\fR
.sp
.LP
will first set the value of \fBdot\fR to 2, ':', will alert the start of a
command, and the \fBinode\fR command will set \fBinode\fR to 2. A count is
specified after a ','. Once set, \fBcount\fR will remain at this value until a
new command is encountered which will then reset the value back to 1 (the
default). So, if
.sp
.LP
\fB> 2000,400/X\fR
.sp
.LP
is typed, 400 hex longs are listed from 2000, and when completed, the value of
\fBdot\fR will be \fB2000 + 400 * sizeof (long)\fR. If a  \fBRETURN\fR is then
typed, the output routine will use the current values of \fBdot\fR,
\fBcount\fR, and \fBtype\fR and display 400 more hex longs. A '*' will cause
the entire block to be displayed.
.sp
.LP
End of fragment, block and file are maintained by \fBfsdb\fR. When displaying
data as fragments or blocks, an error message will be displayed when the end of
fragment or block is reached. When displaying data using the \fBdb\fR,
\fBib\fR, \fBdirectory\fR, or \fBfile\fR commands an error message is displayed
if the end of file is reached. This is mainly needed to avoid passing the end
of a directory or file and getting unknown and unwanted results.
.sp
.LP
An example showing several commands and the use of  \fBRETURN\fR would be:
.sp
.in +2
.nf
\fB> 2:ino; 0:dir?d\fR
      or
\fB> 2:ino; 0:db:block?d\fR
.fi
.in -2
.sp

.sp
.LP
The two examples are synonymous for getting to the first directory entry of the
root of the file system. Once there, any subsequent  \fBRETURN\fR (or +, -)
will advance to subsequent entries. Note that
.sp
.in +2
.nf
```

```
\fB> 2:inode; :ls\fR
      or
\fB> :ls /\fR
.fi
.in -2
.sp

.sp
.LP
is again synonymous.
.SS "Expressions"
.sp
.LP
The symbols recognized by \fBfsdb\fR are:
.sp
.ne 2
.na
\fB\fBRETURN\fR\fR
.ad
.RS 13n
update the value of \fBdot\fR by the current value of \fBtype\fR and display
using the current value of \fBcount\fR.
.RE

.sp
.ne 2
.na
\fB\fB#\fR\fR
.ad
.RS 13n
numeric expressions may be composed of +, -, *, and % operators (evaluated left
to right) and may use parentheses. Once evaluated, the value of \fBdot\fR is
updated.
.RE

.sp
.ne 2
.na
\fB\fB,\fR\fI count\fR\fR
.ad
.RS 13n
count indicator. The global value of \fBcount\fR will be updated to
\fBcount\fR. The value of \fBcount\fR will remain until a new command is run. A
count specifier of '*' will attempt to show a \fIblocks's\fR worth of
information. The default for \fBcount\fR is 1.
.RE

.sp
.ne 2
.na
\fB\fB?\fR\fI f\fR\fR
.ad
.RS 13n
display in structured style with format specifier \fIf\fR. See
\fBFormatted Output\fR.
\fBFormatted\fROutput\fB\&.\fR
.RE

.sp
.ne 2
.na
\fB\fB/\fR\fI f\fR\fR
.ad
.RS 13n
display in unstructured style with format specifier \fIf\fR. See
\fBFormatted Output\fR.
```

```
display in unstructured style with format specifier \fIf\fR See
\fBFormatted\fROutput\fB\&.\fR
.RE

.sp
.ne 2
.na
\fB\fB\&.\fR\fR
.ad
.RS 13n
the value of \fBdot\fR.
.RE

.sp
.ne 2
.na
\fB\fB+\fR\fIe\fR\fR
.ad
.RS 13n
increment the value of \fBdot\fR by the expression \fIe.\fR The amount actually
incremented is dependent on the size of \fBtype\fR:
.sp
\fBdot = dot + e * sizeof (type)\fR
.sp
The default for \fIe\fR is \fB1\fR.
.RE

.sp
.ne 2
.na
\fB\fB-\fR\fIe\fR\fR
.ad
.RS 13n
decrement the value of \fBdot\fR by the expression \fIe\fR. See \fB+\fR.
.RE

.sp
.ne 2
.na
\fB\fB*\fR\fIe\fR\fR
.ad
.RS 13n
multiply the value of \fBdot\fR by the expression \fIe.\fR Multiplication and
division don't use \fBtype\fR. In the above calculation of \fBdot\fR, consider
the \fBsizeof(type)\fR to be \fB1\fR.
.RE

.sp
.ne 2
.na
\fB\fB%\fR\fIe\fR\fR
.ad
.RS 13n
divide the value of \fBdot\fR by the expression \fIe\fR. See \fB*\fR.
.RE

.sp
.ne 2
.na
\fB\fB<\fR\fI name\fR\fR
.ad
.RS 13n
restore an address saved in register \fIname\fR. \fIname\fR must be a single
letter or digit.
.RE
```

```
.sp
.ne 2
.na
\fB\fB>\fR\fI name\fR\fR
.ad
.RS 13n
save an address in register \fIname\fR. \fIname\fR must be a single letter or
digit.
.RE

.sp
.ne 2
.na
\fB\fB=\fR\fI f\fR\fR
.ad
.RS 13n
display indicator. If \fIf\fR is a legitimate format specifier, then the value
display indicator. If \fIf\fR is a legitimate format specifier. then the value
of \fBdot\fR is displayed using the format specifier \fIf\fR. See
\fBFormatted Output\fR. Otherwise, assignment is assumed. See \fB=\fR.
\fBFormatted\fROutput. Otherwise, assignment is assumed See  \fB=\fR.
.RE

.sp
.ne 2
.na
\fB\fB= [\fR\fIs\fR\fB] [\fR\fIe\fR\fB]\fR\fR
.ad
.RS 13n
assignment indicator. The address pointed to by \fBdot\fR has its contents
changed to the value of the expression \fIe\fR or to the \fBASCII\fR
representation of the quoted (") string \fIs\fR. This may be useful for
changing directory names or \fBASCII\fR file information.
.RE

.sp
.ne 2
.na
\fB\fB=+\fR\fI e\fR\fR
.ad
.RS 13n
incremental assignment. The address pointed to by \fBdot\fR has its contents
incremented by expression \fIe\fR.
.RE

.sp
.ne 2
.na
\fB\fB=-\fR\fI e\fR\fR
.ad
.RS 13n
decremental assignment. The address pointed to by \fBdot\fR has its contents
decremented by expression \fIe\fR.
.RE

.SS "Commands"
.sp
.LP
A command must be prefixed by a ':' character. Only enough letters of the
command to uniquely distinguish it are needed. Multiple commands may be entered
on one line by separating them by a  \fBSPACE,\fR \fBTAB\fR or ';'.
.sp
.LP
In order to view a potentially unmounted disk in a reasonable manner,
\fBfsdb\fR offers the \fBcd\fR, \fBpwd\fR, \fBls\fR and \fBfind\fR commands.
The functionality of these commands substantially matches those of its UNIX
```

```
counterparts. See individual commands for details. The '*', '?', and '[-]' wild
card characters are available.
.sp
.ne 2
.na
\fB\fBbase=b\fR\fR
.ad
.sp .6
.RS 4n
display or set base. As stated above, all input and output is governed by the
current \fBbase\fR. If the  \fB=b\fR is omitted, the current \fBbase\fR is
displayed. Otherwise, the current \fBbase\fR is set to \fIb.\fR Note that this
is interpreted using the old value of \fBbase\fR, so to ensure correctness use
the '0', '0t', or '0x' prefix when changing the \fBbase\fR. The default for
\fBbase\fR is hexadecimal.
.RE

.sp
.ne 2
.na
\fB\fBblock\fR\fR
.ad
.sp .6
.RS 4n
convert the value of \fBdot\fR to a block address.
.RE

.sp
.ne 2
.na
\fB\fBcd \fR\fIdir\fR\fR
.ad
.sp .6
.RS 4n
change the current directory to directory \fIdir\fR. The current values of
\fBinode\fR and \fBdot\fR are also updated. If no \fIdir\fR is specified, then
change directories to inode \fB2\fR ("/").
.RE

.sp
.ne 2
.na
\fB\fBcg\fR\fR
.ad
.sp .6
.RS 4n
convert the value of \fBdot\fR to a cylinder group.
.RE

.sp
.ne 2
.na
\fB\fBdirectory\fR\fR
.ad
.sp .6
.RS 4n
If the current \fBinode\fR is a directory, then the value of \fBdot\fR is
converted to a directory slot offset in that directory and \fBdot\fR now points
to this entry.
.RE

.sp
.ne 2
.na
\fB\fBfile\fR\fR
.ad
```

```
.sp .6
.RS 4n
the value of \fBdot\fR is taken as a relative block count from the beginning of
the file. The value of \fBdot\fR is updated to the first byte of this block.
.RE

.sp
.ne 2
.na
\fB\fBfind\fR \fIdir\fR [ \fB-name\fR \fIn\fR] [\fB-inum\fR \fIi\fR]\fR
.ad
.sp .6
.RS 4n
find files by name or i-number. \fBfind\fR recursively searches directory
\fBdir\fR and below for filenames whose i-number matches \fIi\fR or whose name
matches pattern \fIn\fR. Note that only one of the two options (-name or -inum)
may be used at one time. Also, the -print is not needed or accepted.
.RE

.sp
.ne 2
.na
\fB\fBfill\fR\fI=p\fR\fR
.ad
.sp .6
.RS 4n
fill an area of disk with pattern \fIp\fR. The area of disk is delimited by
\fBdot\fR and \fBcount\fR.
.RE

.sp
.ne 2
.na
\fB\fBfragment\fR\fR
.ad
.sp .6
.RS 4n
convert the value of \fIdot\fR to a fragment address. The only difference
between the \fBfragment\fR command and the \fBblock\fR command is the amount
that is able to be displayed.
.RE

.sp
.ne 2
.na
\fB\fBinode\fR\fR
.ad
.sp .6
.RS 4n
convert the value of \fIdot\fR to an inode address. If successful, the current
value of \fBinode\fR will be updated as well as the value of \fIdot\fR. As a
convenient shorthand, if ':inode' appears at the beginning of the line, the
value of \fIdot\fR is set to the current \fBinode\fR and that inode is
displayed in inode format.
.RE

.sp
.ne 2
.na
\fB\fBlog_chk\fR\fR
.ad
.sp .6
.RS 4n
run through the valid log entries without printing any information and verify
the layout.
.RE
```

```
.sp
.ne 2
.na
\fB\fBlog_delta\fR\fR
.ad
.sp .6
.RS 4n
count the number of deltas into the log, using the value of dot as an offset
into the log. No checking is done to make sure that offset is within the
head/tail offsets.
.RE

.sp
.ne 2
.na
\fB\fBlog_head\fR\fR
.ad
.sp .6
.RS 4n
display the header information about the file system logging. This shows the
block allocation for the log and the data structures on the disk.
.RE

.sp
.ne 2
.na
\fB\fBlog_otodb\fR\fR
.ad
.sp .6
.RS 4n
return the physical disk block number, using the value of dot as an offset into
the log.
.RE

.sp
.ne 2
.na
\fB\fBlog_show\fR\fR
.ad
.sp .6
.RS 4n
display all deltas between  the beginning of the log (BOL) and the end of the
log (EOL).
.RE

.sp
.ne 2
.na
\fB\fBls\fR\fR
.ad
.sp .6
.RS 4n
[ \fB-R\fR ] [ \fB-l\fR ] \fIpat1 pat2\fR\|.\|.\|. list directories or files.
If no file is specified, the current directory is assumed. Either or both of
the options may be used (but, if used, \fImust\fR be specified before the
filename specifiers). Also, as stated above, wild card characters are available
and multiple arguments may be given. The long listing shows only the i-number
and the name; use the \fBinode\fR command with '?i' to get more information.
.RE

.sp
.ne 2
.na
\fB\fBoverride\fR\fR
.ad
```

```
.sp .6
.RS 4n
toggle the value of override. Some error conditions may be overridden if
toggle the value of override. Some error conditions may be overriden if
override is toggled on.
.RE

.sp
.ne 2
.na
\fB\fBprompt\fR\fI p\fR\fR
.ad
.sp .6
.RS 4n
change the \fBfsdb\fR prompt to \fIp\fR. \fIp\fR must be surrounded by (")s.
.RE

.sp
.ne 2
.na
\fB\fBpwd\fR\fR
.ad
.sp .6
.RS 4n
display the current working directory.
.RE

.sp
.ne 2
.na
\fB\fBquit\fR\fR
.ad
.sp .6
.RS 4n
quit \fBfsdb\fR.
.RE

.sp
.ne 2
.na
\fB\fBsb\fR\fR
.ad
.sp .6
.RS 4n
the value of \fIdot\fR is taken as a cylinder group number and then converted
to the address of the superblock in that cylinder group. As a shorthand, ':sb'
at the beginning of a line will set the value of \fIdot\fR to \fIthe\fR
superblock and display it in superblock format.
.RE

.sp
.ne 2
.na
\fB\fBshadow\fR\fR
.ad
.sp .6
.RS 4n
if the current inode is a shadow inode, then the value of \fIdot\fR is set to
the beginning of the shadow inode data.
.RE

.sp
.ne 2
.na
\fB\fB!\fR\fR
.ad
```

```
.sp .6
.RS 4n
escape to shell
.RE

.SS "Inode Commands"
.sp
.LP
In addition to the above commands, there are several commands that deal with
inode fields and operate directly on the current \fBinode\fR (they still
require the ':'). They may be used to more easily display or change the
particular fields. The value of \fIdot\fR is only used by the '\fB:db\fR'
and '\fB:ib\fR' commands. Upon completion of the command, the value of \fIdot\fR
changed to point to that particular field. For example,
.sp
.LP
\fB> :ln=+1\fR
.sp
.LP
would increment the link count of the current \fBinode\fR and set the value of
\fIdot\fR to the address of the link count field.
.sp
.ne 2
.na
\fB\fBat\fR\fR
.ad
.RS 7n
access time.
.RE

.sp
.ne 2
.na
\fB\fBbs\fR\fR
.ad
.RS 7n
block size.
.RE

.sp
.ne 2
.na
\fB\fBct\fR\fR
.ad
.RS 7n
creation time.
.RE

.sp
.ne 2
.na
\fB\fBdb\fR\fR
.ad
.RS 7n
use the current value of \fIdot\fR as a direct block index, where direct blocks
number from 0 - 11. In order to display the block itself, you need to 'pipe'
this result into the \fBblock\fR or \fBfragment\fR command. For example,
.sp
.in +2
.nf
\fB     > 1:db:block,20/X\fR
.fi
.in -2
.sp

would get the contents of data block field 1 from the inode and convert it to a
```

```
block address. 20 longs are then displayed in hexadecimal. See
\fBFormatted Output\fR.
\fBFormatted\fROutput\fB\&.\fR
.RE

.sp
.ne 2
.na
\fB\fBgid\fR\fR
.ad
.RS 7n
group id.
.RE

.sp
.ne 2
.na
\fB\fBib\fR\fR
.ad
.RS 7n
use the current value of \fIdot\fR as an indirect block index where indirect
blocks number from 0 - 2. This will only get the indirect block itself (the
block containing the pointers to the actual blocks). Use the \fBfile\fR command
and start at block 12 to get to the actual blocks.
.RE

.sp
.ne 2
.na
\fB\fBln\fR\fR
.ad
.RS 7n
link count.
.RE

.sp
.ne 2
.na
\fB\fBmt\fR\fR
.ad
.RS 7n
modification time.
.RE

.sp
.ne 2
.na
\fB\fBmd\fR\fR
.ad
.RS 7n
mode.
.RE

.sp
.ne 2
.na
\fB\fBmaj\fR\fR
.ad
.RS 7n
major device number.
.RE

.sp
.ne 2
.na
\fB\fBmin\fR\fR
```

```
.ad
.RS 7n
minor device number.
.RE

.sp
.ne 2
.na
\fB\fBnm\fR\fR
.ad
.RS 7n
although listed here, this command actually operates on the directory name
field. Once poised at the desired directory entry (using the \fIdirectory\fR
command), this command will allow you to change or display the directory name.
For example,
.sp
\fB> 7:dir:nm="foo"\fR
.sp
will get the \fB7\fRth directory entry of the current \fBinode\fR and change
its name to foo. Note that names cannot be made larger than the field is set up
for. If an attempt is made, the string is truncated to fit and a warning
message to this effect is displayed.
.RE

.sp
.ne 2
.na
\fB\fBsi\fR\fR
.ad
.RS 7n
shadow inode.
.RE

.sp
.ne 2
.na
\fB\fBsz\fR\fR
.ad
.RS 7n
file size.
.RE

.sp
.ne 2
.na
\fB\fBuid\fR\fR
.ad
.RS 7n
user id.
.RE

.SS "Formatted Output"
.sp
.LP
There are two styles and many format types. The two styles are structured and
unstructured. Structured output is used to display inodes, directories,
superblocks and the like. Unstructured displays raw data. The following shows
the different ways of displaying:
.sp
.ne 2
.na
\fB\fB?\fR\fR
.ad
.RS 5n
.sp
.ne 2
```

```
.na
\fB\fBc\fR\fR
.ad
.RS 5n
display as cylinder groups
.RE

.sp
.ne 2
.na
\fB\fBi\fR\fR
.ad
.RS 5n
display as inodes
.RE

.sp
.ne 2
.na
\fB\fBd\fR\fR
.ad
.RS 5n
display as directories
.RE

.sp
.ne 2
.na
\fB\fBs\fR\fR
.ad
.RS 5n
display as superblocks
.RE

.sp
.ne 2
.na
\fB\fBS\fR\fR
.ad
.RS 5n
display as shadow inode data
.RE

.RE

.sp
.ne 2
.na
\fB\fB/\fR\fR
.ad
.RS 5n
.sp
.ne 2
.na
\fB\fBb\fR\fR
.ad
.RS 7n
display as bytes
.RE

.sp
.ne 2
.na
\fB\fBc\fR\fR
.ad
.RS 7n
```

```
display as characters
.RE

.sp
.ne 2
.na
\fB\fBo O\fR\fR
.ad
.RS 7n
display as octal shorts or longs
.RE

.sp
.ne 2
.na
\fB\fBd D\fR\fR
.ad
.RS 7n
display as decimal shorts or longs
.RE

.sp
.ne 2
.na
\fB\fBx X\fR\fR
.ad
.RS 7n
display as hexadecimal shorts or longs
.RE

The format specifier immediately follows the '/' or '?' character. The values
displayed by '/b' and all '?' formats are displayed in the current \fBbase\fR.
Also, \fBtype\fR is appropriately updated upon completion.
.RE

.SH EXAMPLES
.LP
\fBExample 1 \fRDisplaying in Decimal
.sp
.LP
The following command displays \fB2010\fR in decimal (use of \fBfsdb\fR as a
calculator for complex arithmetic):

.sp
.in +2
.nf
> 2000+400%(20+20)=D
.fi
.in -2
.sp

.LP
\fBExample 2 \fRDisplaying an i-number in Inode Format
.sp
.LP
The following command displays i-number \fB386\fR in an inode format. This now
becomes the current \fBinode\fR:

.sp
.in +2
.nf
> 386:ino?i
.fi
.in -2
.sp
```

```
.LP
\fBExample 3 \fRChanging the Link Count
.sp
.LP
The following command changes the link count for the current \fBinode\fR to
\fB4\fR:

.sp
.in +2
.nf
> :ln=4
.fi
.in -2
.sp

.LP
\fBExample 4 \fRIncrementing the Link Count
.sp
.LP
The following command increments the link count by \fB1\fR:

.sp
.in +2
.nf
> :ln=+1
.fi
.in -2
.sp

.LP
\fBExample 5 \fRDisplaying the Creation Time
.sp
.LP
The following command displays the creation time as a hexadecimal long:

.sp
.in +2
.nf
> :ct=X
.fi
.in -2
.sp

.LP
\fBExample 6 \fRDisplaying the Modification Time
.sp
.LP
The following command displays the modification time in time format:

.sp
.in +2
.nf
> :mt=t
.fi
.in -2
.sp

.LP
\fBExample 7 \fRDisplaying in ASCII
.sp
.LP
The following command displays in \fBASCII,\fR block zero of the file
associated with the current \fBinode\fR:

.sp
.in +2
```

```
.nf
> 0:file/c
.fi
.in -2
.sp

.LP
\fBExample 8 \fRDisplaying the First Block’s Worth of Directorty Entries
.sp
.LP
The following command displays the first block’s worth of directory entries for
the root inode of this file system. It will stop prematurely if the \fBEOF\fR
is reached:

.sp
.in +2
.nf
> 2:ino,*?d
.fi
.in -2
.sp

.LP
\fBExample 9 \fRDisplaying Changes to the Current Inode
.sp
.LP
The following command displays changes the current inode to that associated
with the \fB5\fRth directory entry (numbered from zero) of the current
\fBinode\fR. The first logical block of the file is then displayed in
\fBASCII\fR:

.sp
.in +2
.nf
> 5:dir:inode; 0:file,*/c
.fi
.in -2
.sp

.LP
\fBExample 10 \fRDisplaying the Superblock
.sp
.LP
The following command displays the superblock of this file system:

.sp
.in +2
.nf
> :sb
.fi
.in -2
.sp

.LP
\fBExample 11 \fRDisplaying the Cylinder Group
.sp
.LP
The following command displays cylinder group information and summary for
cylinder group \fB1\fR:

.sp
.in +2
.nf
> 1:cg?c
.fi
.in -2
```

```
.sp

.LP
\fBExample 12 \fRChanging the i-number
.sp
.LP
The following command changes the i-number for the seventh directory slot in
the root directory to \fB3\fR:

.sp
.in +2
.nf
> 2:inode; 7:dir=3
.fi
.in -2
.sp

.LP
\fBExample 13 \fRDisplaying as Directory Entries
.sp
.LP
The following command displays the third block of the current \fBinode\fR as
directory entries:

.sp
.in +2
.nf
> 2:db:block,*?d
.fi
.in -2
.sp

.LP
\fBExample 14 \fRChanging the Name Field
.sp
.LP
The following command changes the name field in the directory slot to
\fIname\fR:

.sp
.in +2
.nf
> 7:dir:nm="name"
.fi
.in -2
.sp

.LP
\fBExample 15 \fRGetting and Filling Elements
.sp
.LP
The following command gets fragment \fB3c3\fR and fill \fB20\fR \fBtype\fR
elements with \fB0x20\fR:

.sp
.in +2
.nf
> 3c3:fragment,20:fill=0x20
.fi
.in -2
.sp

.LP
\fBExample 16 \fRSetting the Contents of an Address
.sp
.LP
```

```
The following command sets the contents of address \fB2050\fR to
\fB0xffffffff\fR. \fB0xffffffff\fR may be truncated depending on the current
\fBtype\fR:

.sp
.in +2
.nf
> 2050=0xffff
.fi
.in -2
.sp

.LP
\fBExample 17 \fRPlacing ASCII
.sp
.LP
The following command places the \fBASCII\fR for the string at \fB1c92434\fR:

.sp
.in +2
.nf
> 1c92434="this is some text"
.fi
.in -2
.sp

.LP
\fBExample 18 \fRDisplaying Shadow Inode Data
.sp
.LP
The following command displays all of the shadow inode data in the shadow inode
associated with the root inode of this file system:

.sp
.in +2
.nf
> 2:ino:si:ino;0:shadow,*?S
.fi
.in -2
.sp

.SH SEE ALSO
.sp
.LP
\fBclri\fR(1M), \fBfsck_ufs\fR(1M), \fBdir_ufs\fR(4), \fBattributes\fR(5),
\fBufs\fR(7FS)
.SH WARNINGS
.sp
.LP
Since \fBfsdb\fR reads the disk raw, extreme caution is advised in determining
its availability of \fBfsdb\fR on the system. Suggested permissions are 600 and
owned by bin.
.SH NOTES
.sp
.LP
The old command line syntax for clearing i-nodes using the ufs-specific
\fB\&'-z i-number'\fR option is still supported by the new debugger, though it
is obsolete and will be removed in a future release. Use of this flag will
result in correct operation, but an error message will be printed warning of
the impending obsolesence of this option to the command. The equivalent
functionality is available using the more flexible \fBclri\fR(1M) command.
```

```
**********************************************************
   19056 Tue Dec 11 09:48:08 2018
new/usr/src/man/man1m/ikeadm.1m
10057 Man page misspellings ouput particuliar overriden
Reviewed by: Gerg¯  Mih^¡ly Doma <domag02@gmail.com>
**********************************************************
'\" te
.\" Copyright (c) 2009, Sun Microsystems, Inc. All Rights Reserved.
.\" The contents of this file are subject to the terms of the Common Development
.\" You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE or http:
.\" When distributing Covered Code, include this CDDL HEADER in each file and in
.TH IKEADM 1M "Jan 27, 2009"
.SH NAME
ikeadm \- manipulate Internet Key Exchange (IKE) parameters and state
.SH SYNOPSIS
.LP
.nf
\fBikeadm\fR [\fB-np\fR]
.fi

.LP
.nf
\fBikeadm\fR [\fB-np\fR] get [debug | priv | stats | defaults]
.fi

.LP
.nf
\fBikeadm\fR [\fB-np\fR] set [debug | priv] [level] [file]
.fi

.LP
.nf
\fBikeadm\fR [\fB-np\fR] [get | del] [p1 | rule | preshared] [id]
.fi

.LP
.nf
\fBikeadm\fR [\fB-np\fR] add [rule | preshared] { \fIdescription\fR }
.fi

.LP
.nf
ikeadm [\fB-np\fR] token [login | logout] \fIPKCS#11_Token_Object\fR
.fi

.LP
.nf
\fBikeadm\fR [\fB-np\fR] [read | write] [rule | preshared | certcache] \fIfile\f
.fi

.LP
.nf
\fBikeadm\fR [\fB-np\fR] [dump | pls | rule | preshared]
.fi

.LP
.nf
\fBikeadm\fR [\fB-np\fR] flush [p1 | certcache]
.fi

.LP
.nf
\fBikeadm\fR help
     [get | set | add | del | read | write | dump | flush | token]
.fi
```

```
.SH DESCRIPTION
.sp
.LP
The \fBikeadm\fR utility retrieves information from and manipulates the
configuration of the Internet Key Exchange (\fBIKE\fR) protocol daemon,
\fBin.iked\fR(1M).
.sp
.LP
\fBikeadm\fR supports a set of operations, which may be performed on one or
more of the supported object types. When invoked without arguments,
\fBikeadm\fR enters interactive mode which prints a prompt to the standard
output and accepts commands from the standard input until the end-of-file is
reached.
.sp
.LP
Because \fBikeadm\fR manipulates sensitive keying information, you must be
superuser to use this command. Additionally, some of the commands available
require that the daemon be running in a privileged mode, which is established
when the daemon is started.
.sp
.LP
For details on how to use this command securely see .
.SH OPTIONS
.sp
.LP
The following options are supported:
.sp
.ne 2
.na
\fB\fB-n\fR\fR
.ad
.sp .6
.RS 4n
Prevent attempts to print host and network names symbolically when reporting
actions. This is useful, for example, when all name servers are down or are
otherwise unreachable.
.RE

.sp
.ne 2
.na
\fB\fB-p\fR\fR
.ad
.sp .6
.RS 4n
Paranoid. Do not print any keying material, even if saving Security
Associations. Instead of an actual hexadecimal digit, print an \fBX\fR when
this flag is turned on.
.RE

.SH USAGE
.SS "Commands"
.sp
.LP
The following commands are supported:
.sp
.ne 2
.na
\fB\fBadd\fR\fR
.ad
.sp .6
.RS 4n
Add the specified object. This option can be used to add a new policy rule or a
new preshared key to the current (running) in.iked configuration. When adding a
new preshared key, the command cannot be invoked from the command line, as it
will contain keying material. The rule or key being added is specified using
```

```
appropriate id-value pairs as described in the \fBID FORMATS\fR section.
.RE

.sp
.ne 2
.na
\fB\fBdel\fR\fR
.ad
.sp .6
.RS 4n
Delete a specific object or objects from \fBin.iked\fR’s current configuration.
This operation is available for \fBIKE\fR (Phase 1) \fBSA\fRs, policy rules,
and preshared keys. The object to be deleted is specified as described in the
\fBId Formats\fR.
.RE

.sp
.ne 2
.na
\fB\fBdump\fR\fR
.ad
.sp .6
.RS 4n
Display all objects of the specified type known to \fBin.iked\fR. This option
can be used to display all Phase 1 \fBSA\fRs, policy rules, preshared keys, or
the certificate cache. A large amount of output may be generated by this
command.
.RE

.sp
.ne 2
.na
\fB\fBflush\fR\fR
.ad
.sp .6
.RS 4n
Remove all \fBIKE\fR (Phase 1) \fBSA\fRs or cached certificates from
\fBin.iked\fR.
.sp
Note that flushing the \fBcertcache\fR will also (as a side-effect) update IKE
with any new certificates added or removed.
.RE

.sp
.ne 2
.na
\fB\fBget\fR\fR
.ad
.sp .6
.RS 4n
Lookup and display the specified object. May be used to view the current debug
or privilege level, global statistics and default values for the daemon, or a
specific \fBIKE\fR (Phase 1) \fBSA\fR, policy rule, or preshared key. The
latter three object types require that identifying information be passed in;
the appropriate specification for each object type is described below.
.RE

.sp
.ne 2
.na
\fB\fBhelp\fR\fR
.ad
.sp .6
.RS 4n
Print a brief summary of commands, or, when followed by a command, prints
information about that command.
```

```
.RE

.sp
.ne 2
.na
\fB\fBread\fR\fR
.ad
.sp .6
.RS 4n
Update the current \fBin.iked\fR configuration by reading the policy rules or
preshared keys from either the default location or from the file specified.
.RE

.sp
.ne 2
.na
\fB\fBset\fR\fR
.ad
.sp .6
.RS 4n
Adjust the current debug or privilege level. If the debug level is being
modified, an output file may optionally be specified; the output file
\fBmust\fR be specified if the daemon is running in the background and is not
currently printing to a file. When changing the privilege level, adjustments
may only be made to lower the access level; it cannot be increased using
ikeadm.
.RE

.sp
.ne 2
.na
\fB\fBwrite\fR\fR
.ad
.sp .6
.RS 4n
Write the current \fBin.iked\fR policy rule set or preshared key set to the
specified file. A destination file must be specified. This command should not
be used to overwrite the existing configuration files.
.RE

.sp
.ne 2
.na
\fB\fBtoken\fR\fR
.ad
.sp .6
.RS 4n
Log into a PKCS#11 token object and grant access to keying material or log out
and invalidate access to keying material.
.sp
\fBtoken\fR can be run as a normal user with the following authorizations:
.RS +4
.TP
.ie t \(bu
.el o
\fBtoken\fR login: \fBsolaris.network.ipsec.ike.token.login\fR
.RE
.RS +4
.TP
.ie t \(bu
.el o
\fBtoken\fR logout: \fBsolaris.network.ipsec.ike.token.logout\fR
.RE
.RE

.SS "Object Types"
```

```
.sp
.ne 2
.na
\fBdebug\fR
.ad
.sp .6
.RS 4n
Specifies the daemon’s debug level. This determines the amount and type of
output provided by the daemon about its operations. The debug level is actually
a bitmask, with individual bits enabling different types of information.
.sp

.sp
.TS
c c c
l l l .
Description     Flag    Nickname
_
Certificate management  0x0001  cert
Key management  0x0002  key
Operational     0x0004  op
Phase 1 SA creation     0x0008  phase1
Phase 2 SA creation     0x0010  phase2
PF_KEY interface        0x0020  pfkey
Policy management       0x0040  policy
Proposal construction   0x0080  prop
Door interface  0x0100  door
Config file processing  0x0200  config
All debug flags 0x3ff   all
.TE

When specifying the debug level, either a number (decimal or hexadecimal) or a
string of nicknames may be given. For example, \fB88\fR, \fB0x58\fR, and
\fBphase1\fR+\fBphase2\fR+\fBpolicy\fR are all equivalent, and will turn on
debug for \fBphase 1\fR \fBsa\fR creation, \fBphase 2 sa\fR creation, and
policy management. A string of nicknames may also be used to remove certain
types of information; \fBall-op\fR has the effect of turning on all debug
\fBexcept\fR for operational messages; it is equivalent to the numbers
\fB1019\fR or \fB0x3fb\fR.
.RE

.sp
.ne 2
.na
\fBpriv\fR
.ad
.sp .6
.RS 4n
Specifies the daemon’s access privilege level. The possible values are:
.sp
.in +2
.nf
Description                  Level   Nickname
Base level                   0       base
Access to preshared key info 1       modkeys
Access to keying material    2       keymat
.fi
.in -2
.sp

By default, \fBin.iked\fR is started at the base level. A command-line option
can be used to start the daemon at a higher level. \fBikeadm\fR can be used to
lower the level, but it cannot be used to raise the level.
.sp
Either the numerical level or the nickname may be used to specify the target
privilege level.
```

```
.sp
In order to get, add, delete, dump, read, or write preshared keys, the
privilege level must at least give access to preshared key information.
However, when viewing preshared keys (either using the get or dump command),
the key itself will only be available if the privilege level gives access to
keying material. This is also the case when viewing Phase 1 \fBSA\fRs.
.RE

.sp
.ne 2
.na
\fBstats\fR
.ad
.sp .6
.RS 4n
Global statistics from the daemon, covering both successful and failed Phase 1
\fBSA\fR creation.
.sp
Reported statistics include:
.RS +4
.TP
.ie t \(bu
.el o
Count of current P1 \fBSA\fRs which the local entity initiated
.RE
.RS +4
.TP
.ie t \(bu
.el o
Count of current P1 \fBSA\fRs where the local entity was the responder
.RE
.RS +4
.TP
.ie t \(bu
.el o
Count of all P1 \fBSA\fRs which the local entity initiated since boot
.RE
.RS +4
.TP
.ie t \(bu
.el o
Count of all P1 \fBSA\fRs where the local entity was the responder since boot
.RE
.RS +4
.TP
.ie t \(bu
.el o
Count of all attempted \fBP1\fR \fBSA\fRs since boot, where the local entity
was the initiator; includes failed attempts
.RE
.RS +4
.TP
.ie t \(bu
.el o
Count of all attempted P1 \fBSA\fRs since boot, where the local entity was the
responder; includes failed attempts
.RE
.RS +4
.TP
.ie t \(bu
.el o
Count of all failed attempts to initiate a \fBP1\fR \fBSA\fR, where the failure
occurred because the peer did not respond
.RE
.RS +4
.TP
```

```
.ie t \(bu
.el o
Count of all failed attempts to initiate a P1 \fBSA\fR, where the peer
responded
.RE
.RS +4
.TP
.ie t \(bu
.el o
Count of all failed \fBP1\fR \fBSA\fRs where the peer was the initiator
.RE
.RS +4
.TP
.ie t \(bu
.el o
Whether a PKCS#11 library is in use, and if applicable, the PKCS#11 library
that is loaded. See .
.RE
.RE

.sp
.ne 2
.na
\fBdefaults\fR
.ad
.sp .6
.RS 4n
Display default values used by the \fBin.iked\fR daemon. Some values can be
overridden in the daemon configuration file (see \fBike.config\fR(4)); for these
overriden in the daemon configuration file (see \fBike.config\fR(4)); for these
values, the token name is displayed in the \fBget defaults\fR output. The
output will reflect where a configuration token has changed the default.
.sp
Default values might be ignored in the event a peer system makes a valid
alternative proposal or they can be overridden by per-rule values established in
alternative proposal or they can be overriden by per-rule values established in
\fBike.config\fR. In such instances, a \fBget defaults\fR command continues to
display the default values, not the values used to override the defaults.
.RE

.sp
.ne 2
.na
\fBp1\fR
.ad
.sp .6
.RS 4n
An \fBIKE\fR Phase 1 \fBSA\fR. A \fBp1\fR object is identified by an \fBIP\fR
address pair or a cookie pair; identification formats are described below.
.RE

.sp
.ne 2
.na
\fBrule\fR
.ad
.sp .6
.RS 4n
An \fBIKE\fR policy rule, defining the acceptable security characteristics for
Phase 1 \fBSA\fRs between specified local and remote identities. A rule is
identified by its label; identification formats are described below.
.RE

.sp
.ne 2
.na
```

```
\fBpreshared\fR
.ad
.sp .6
.RS 4n
A preshared key, including the local and remote identification and applicable
\fBIKE\fR mode. A preshared key is identified by an \fBIP\fR address pair or an
identity pair; identification formats are described below.
.RE

.SS "Id Formats"
.sp
.LP
Commands like \fBadd\fR, \fBdel\fR, and \fBget\fR require that additional
information be specified on the command line. In the case of the delete and get
commands, all that is required is to minimally identify a given object; for the
add command, the full object must be specified.
.sp
.LP
Minimal identification is accomplished in most cases by a pair of values. For
\fBIP\fR addresses, the local addr and then the remote addr are specified,
either in dot-notation for IPv4 addresses, colon-separated hexadecimal format
for IPv6 addresses, or a host name present in the host name database. If a host
name is given that expands to more than one address, the requested operation
will be performed multiple times, once for each possible combination of
addresses.
.sp
.LP
Identity pairs are made up of a local type-value pair, followed by the remote
type-value pair. Valid types are:
.sp
.ne 2
.na
\fBprefix\fR
.ad
.sp .6
.RS 4n
An address prefix.
.RE

.sp
.ne 2
.na
\fBfqdn\fR
.ad
.sp .6
.RS 4n
A fully-qualified domain name.
.RE

.sp
.ne 2
.na
\fBdomain\fR
.ad
.sp .6
.RS 4n
Domain name, synonym for fqdn.
.RE

.sp
.ne 2
.na
\fBuser_fqdn\fR
.ad
.sp .6
.RS 4n
```

```
User identity of the form \fIuser\fR@fqdn.
.RE

.sp
.ne 2
.na
\fBmailbox\fR
.ad
.sp .6
.RS 4n
Synonym for \fBuser_fqdn\fR.
.RE

.sp
.LP
A cookie pair is made up of the two cookies assigned to a Phase 1 Security
Association (\fBSA\fR) when it is created; first is the initiator’s, followed
by the responder’s. A cookie is a 64-bit number.
.sp
.LP
Finally, a label (which is used to identify a policy rule) is a character
string assigned to the rule when it is created.
.sp
.LP
Formatting a rule or preshared key for the add command follows the format rules
for the in.iked configuration files. Both are made up of a series of id-value
pairs, contained in curly braces (\fB{\fR and \fB}\fR). See \fBike.config\fR(4)
and \fBike.preshared\fR(4) for details on the formatting of rules and preshared
keys.
.SH SECURITY
.sp
.LP
The \fBikeadm\fR command allows a privileged user to enter cryptographic keying
information. If an adversary gains access to such information, the security of
IPsec traffic is compromised. The following issues should be taken into account
when using the \fBikeadm\fR command.
.RS +4
.TP
.ie t \(bu
.el o
Is the \fBTTY\fR going over a network (interactive mode)?
.sp
If it is, then the security of the keying material is the security of the
network path for this \fBTTY\fR’s traffic. Using \fBikeadm\fR over a clear-text
telnet or rlogin session is risky. Even local windows may be vulnerable to
attacks where a concealed program that reads window events is present.
.RE
.RS +4
.TP
.ie t \(bu
.el o
Is the file accessed over the network or readable to the world (read/write
commands)?
.sp
A network-mounted file can be sniffed by an adversary as it is being read. A
world-readable file with keying material in it is also risky.
.RE
.sp
.LP
If your source address is a host that can be looked up over the network, and
your naming system itself is compromised, then any names used will no longer be
trustworthy.
.sp
.LP
Security weaknesses often lie in misapplication of tools, not the tools
themselves. It is recommended that administrators are cautious when using the
```

```
\fBikeadm\fR command. The safest mode of operation is probably on a console, or
other hard-connected \fBTTY\fR.
.sp
.LP
For additional information regarding this subject, see the afterward by Matt
Blaze in Bruce Schneier’s \fIApplied Cryptography: Protocols, Algorithms, and
Source Code in C.\fR
.SH EXAMPLES
.LP
\fBExample 1 \fREmptying out all Phase 1 Security Associations
.sp
.LP
The following command empties out all Phase 1 Security Associations:

.sp
.in +2
.nf
example# \fBikeadm flush p1\fR
.fi
.in -2
.sp

.LP
\fBExample 2 \fRDisplaying all Phase 1 Security Associations
.sp
.LP
The following command displays all Phase 1 Security Associations:

.sp
.in +2
.nf
example# \fBikeadm dump p1\fR
.fi
.in -2
.sp

.LP
\fBExample 3 \fRDeleting a Specific Phase 1 Security Association
.sp
.LP
The following command deletes the specified Phase 1 Security Associations:

.sp
.in +2
.nf
example# \fBikeadm del p1 local_ip remote_ip\fR
.fi
.in -2
.sp

.LP
\fBExample 4 \fRAdding a Rule From a File
.sp
.LP
The following command adds a rule from a file:

.sp
.in +2
.nf
example# \fBikeadm add rule rule_file\fR
.fi
.in -2
.sp

.LP
\fBExample 5 \fRAdding a Preshared Key
```

```
.sp
.LP
The following command adds a preshared key:

.sp
.in +2
.nf
example# \fBikeadm\fR
     ikeadm> \fBadd preshared { localidtype ip localid local_ip
             remoteidtype ip remoteid remote_ip ike_mode main
             key 1234567890abcdef1234567890abcdef }\fR
.fi
.in -2
.sp

.LP
\fBExample 6 \fRSaving All Preshared Keys to a File
.sp
.LP
The following command saves all preshared keys to a file:

.sp
.in +2
.nf
example# \fBikeadm write preshared target_file\fR
.fi
.in -2
.sp

.LP
\fBExample 7 \fRViewing a Particular Rule
.sp
.LP
The following command views a particular rule:

.sp
.in +2
.nf
example# \fBikeadm get rule rule_label\fR
.fi
.in -2
.sp

.LP
\fBExample 8 \fRReading in New Rules from \fBike.config\fR
.sp
.LP
The following command reads in new rules from the ike.config file:

.sp
.in +2
.nf
example# \fBikeadm read rules\fR
.fi
.in -2
.sp

.LP
\fBExample 9 \fRLowering the Privilege Level
.sp
.LP
The following command lowers the privilege level:

.sp
.in +2
.nf
```

```
example# \fBikeadm set priv base\fR
.fi
.in -2
.sp

.LP
\fBExample 10 \fRViewing the Debug Level
.sp
.LP
The following command shows the current debug level

.sp
.in +2
.nf
example# \fBikeadm get debug\fR
.fi
.in -2
.sp

.LP
\fBExample 11 \fRUsing stats to Verify Hardware Accelerator
.sp
.LP
The following example shows how stats may include an optional line at the end
to indicate if IKE is using a PKCS#11 library to accelerate public-key
operations, if applicable.

.sp
.in +2
.nf
example# \fBikeadm get stats\fR
Phase 1 SA counts:
Current:  initiator:     0    responder:      0
Total:    initiator:    21   responder:      27
Attempted:initiator:    21   responder:      27
Failed:   initiator:     0   responder:       0
                 initiator fails include 0 time-out(s)
PKCS#11 library linked in from /opt/SUNWconn/lib/libpkcs11.so
example#
.fi
.in -2
.sp

.LP
\fBExample 12 \fRDisplaying the Certificate Cache
.sp
.LP
The following command shows the certificate cache and the status of associated
private keys, if applicable:

.sp
.in +2
.nf
example# \fBikeadm dump certcache\fR
.fi
.in -2
.sp

.LP
\fBExample 13 \fRLogging into a PKCS#11 Token
.sp
.LP
The following command shows logging into a PKCS#11 token object and unlocking
private keys:

.sp
```

```
.in +2
.nf
example# \fBikeadm token login "Sun Metaslot"\fR
Enter PIN for PKCS#11 token:
ikeadm: PKCS#11 operation successful
.fi
.in -2
.sp

.SH EXIT STATUS
.sp
.LP
The following exit values are returned:
.sp
.ne 2
.na
\fB\fB0\fR\fR
.ad
.RS 12n
Successful completion.
.RE

.sp
.ne 2
.na
\fB\fBnon-zero\fR\fR
.ad
.RS 12n
An error occurred. Writes an appropriate error message to standard error.
.RE

.SH ATTRIBUTES
.sp
.LP
See \fBattributes\fR(5) for descriptions of the following attributes:
.sp

.sp
.TS
box;
c | c
l | l .
ATTRIBUTE TYPE  ATTRIBUTE VALUE
_
Interface Stability     Not an Interface
.TE

.SH SEE ALSO
.sp
.LP
\fBin.iked\fR(1M), \fBike.config\fR(4), \fBike.preshared\fR(4),
\fBattributes\fR(5), \fBipsec\fR(7P)
.sp
.LP
Schneier, Bruce, \fIApplied Cryptography: Protocols, Algorithms, and Source
Code in C\fR, Second Edition, John Wiley & Sons, New York, NY, 1996.
.SH NOTES
.sp
.LP
As \fBin.iked\fR can run only in the global zone and exclusive-IP zones, this
command is not useful in shared-IP zones.
```

```
**********************************************************
    4539 Tue Dec 11 09:48:08 2018
new/usr/src/man/man1m/mount_tmpfs.1m
10057 Man page misspellings ouput particuliar overriden
Reviewed by: Gerg¯  Mih^¡ly Doma <domag02@gmail.com>
**********************************************************
'\" te
.\"  Copyright (c) 2003, Sun Microsystems, Inc.  All Rights Reserved
.\"  Copyright 2015 Joyent, Inc.
.\" The contents of this file are subject to the terms of the Common Development
.\" You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE or http:
.\" When distributing Covered Code, include this CDDL HEADER in each file and in
.TH MOUNT_TMPFS 1M "Mar 18, 2015"
.SH NAME
mount_tmpfs \- mount tmpfs file systems
.SH SYNOPSIS
.LP
.nf
\fBmount\fR [\fB-F\fR tmpfs] [\fB-o\fR \fIspecific_options\fR] [\fB-O\fR] \fIspe
.fi

.SH DESCRIPTION
.LP
\fBtmpfs\fR is a memory based file system which uses kernel resources relating
to the \fBVM\fR system and page cache as a file system.
.sp
.LP
\fBmount\fR attaches a \fBtmpfs\fR file system to the file system hierarchy at
the pathname location \fImount_point\fR, which must already exist. If
\fImount_point\fR has any contents prior to the \fBmount\fR operation, these
remain hidden until the file system is once again unmounted. The attributes
(mode, owner, and group) of the root of the \fBtmpfs\fR filesystem are
inherited from the underlying \fImount_point\fR, provided that those attributes
are determinable. If not, the root's attributes are set to their default
values. The mode may also be overridden by the \fBmode\fR mount option, which
values. The mode may also be overriden by the \fBmode\fR mount option, which
takes precedence if set.
.sp
.LP
The \fIspecial\fR argument is usually specified as \fBswap\fR but is in fact
disregarded and assumed to be the virtual memory resources within the system.
.SH OPTIONS
.ne 2
.na
\fB\fB-o\fR \fIspecific_options\fR\fR
.ad
.RS 23n
Specify \fBtmpfs\fR file system specific options in a comma-separated list with
no intervening spaces. If invalid options are specified, a warning  message is
printed and the invalid options are ignored. The following options are
available:
.sp
.ne 2
.na
\fB\fBremount\fR\fR
.ad
.sp .6
.RS 19n
Remounts a file system with a new size. A size not explicitly
set with \fBremount\fR reverts to no limit.
.RE

.sp
.ne 2
.na
\fBmode=\fIoctalmode\fR\fR
```

```
.ad
.RS 19n
The \fImode\fR argument controls the permissions of the \fBtmpfs\fR mount
point.  The argument must be an octal number, of the form passed to
\fBchmod\fR(1).  Only the access mode, setuid, setgid, and sticky bits (a mask
of \fB07777\fR) may be set.  If this option is not provided then the default
mode behaviour, as described above, applies.
.RE

.sp
.sp
.ne 2
.na
\fBsize=\fIsz\fR\fR
.ad
.RS 19n
The \fIsz\fR argument controls the size of this particular \fBtmpfs\fR file
system. If the argument is has a 'k' suffix, the number will be interpreted as
a number of kilobytes. An 'm' suffix will be interpreted as a number of
megabytes. A 'g' suffix will be interpreted as a number of gigabytes. A '%'
suffix will be interpreted as a percentage of the swap space available to the
zone. No suffix is interpreted as bytes. In all cases, the actual size of
the file system is the number of bytes specified, rounded up to the physical
pagesize of the system.
.RE

.sp
.ne 2
.na
\fB\fBxattr\fR | \fBnoxattr\fR\fR
.ad
.RS 19n
Allow or disallow the creation and manipulation of extended attributes. The
default is \fBxattr\fR. See \fBfsattr\fR(5) for a description of extended
attributes.
.RE

.RE

.sp
.ne 2
.na
\fB\fB-O\fR\fR
.ad
.RS 23n
Overlay  mount. Allow the file system to be mounted over an existing mount
point, making the underlying file system inaccessible. If a mount is attempted
on a pre-existing mount point without setting this flag, the mount will fail,
producing the error: \f(CWdevice busy\fR.
.RE

.SH FILES
.ne 2
.na
\fB\fB/etc/mnttab\fR\fR
.ad
.RS 15n
Table of mounted file systems
.RE

.SH SEE ALSO
.LP
\fBmount\fR(1M), \fBmkdir\fR(2), \fBmount\fR(2), \fBopen\fR(2),
\fBumount\fR(2), \fBmnttab\fR(4), \fBattributes\fR(5), \fBfsattr\fR(5),
\fBtmpfs\fR(7FS)
.SH NOTES
```

```
.LP
If the directory on which a file system is to be mounted is a symbolic link,
the file system is mounted on the directory to which the symbolic link refers,
rather than on top of the symbolic link itself.
```

****
   13174 Tue Dec 11 09:48:08 2018
new/usr/src/man/man1m/nfsstat.1m
10057 Man page misspellings ouput particuliar overriden
Reviewed by: Gerg¯ Mih^¡ly Doma <domag02@gmail.com>
****

```
'\" te
.\"  Copyright (c) 2007, Sun Microsystems, Inc. All Rights Reserved
.\" The contents of this file are subject to the terms of the Common Development
.\"  See the License for the specific language governing permissions and limitat
.\" the fields enclosed by brackets "[]" replaced with your own identifying info
.TH NFSSTAT 1M "Jun 16, 2009"
.SH NAME
nfsstat \- NFS statistics
.SH SYNOPSIS
.LP
.nf
\fBnfsstat\fR [\fB-cnrsza\fR] [\fB-T\fR u | d ] [\fB-v\fR \fIversion\fR] [\fIint
.fi

.LP
.nf
\fBnfsstat\fR \fB-m\fR [\fIpathname\fR]...
.fi

.SH DESCRIPTION
.sp
.LP
\fBnfsstat\fR displays statistical information about the \fBNFS\fR and
\fBRPC\fR (Remote Procedure Call), interfaces to the kernel. It can also be
used to reinitialize this information. If no options are given the default is
as follows:
.sp
.LP
\fBnfsstat\fR \fB-csnra\fR
.sp
.LP
The default displays everything, but reinitializes nothing.
.SH OPTIONS
.sp
.ne 2
.na
\fB\fB-a\fR\fR
.ad
.sp .6
.RS 4n
Display \fBNFS_ACL\fR information.
.RE

.sp
.ne 2
.na
\fB\fB-c\fR\fR
.ad
.sp .6
.RS 4n
Display client information. Only the client side \fBNFS\fR, \fBRPC\fR, and
\fBNFS_ACL\fR information is printed. Can be combined with the \fB-n\fR,
\fB-r\fR, and \fB-a\fR options to print client side \fBNFS\fR, \fBRPC\fR, and
\fBNFS_ACL\fR information only.
.RE

.sp
.ne 2
.na
\fB\fB-m\fR [\fIpathname...\fR]\fR
```

```
.ad
.sp .6
.RS 4n
Display statistics for each \fBNFS\fR mounted file system. If \fIpathname\fR is
not specified, displays statistics for all NFS mounted file systems. If
\fIpathname\fR is specified, displays statistics for the NFS mounted file
systems indicated by \fIpathname\fR.
.sp
This includes the server name and address, mount flags, current read and write
sizes, the retransmission count, the attribute cache timeout values, failover
information, and the timers used for dynamic retransmission. The dynamic
retransmission timers are displayed only where dynamic retransmission is in
use. By default, \fBNFS\fR mounts over the \fBTCP\fR protocols and \fBNFS\fR
Version 3 mounts over either \fBTCP\fR or \fBUDP\fR do not use dynamic
retransmission.
.sp
If you specify the \fB-m\fR option, this is the only option that \fBnfsstat\fR
uses. If you specify other options with \fB-m\fR, you receive an error message
alerting that the \fB-m\fR flag cannot be combined with other options.
.RE

.sp
.ne 2
.na
\fB\fB-n\fR\fR
.ad
.sp .6
.RS 4n
Display \fBNFS\fR information. \fBNFS\fR information for both the client and
server side are printed. Can be combined with the \fB-c\fR and \fB-s\fR options
to print client or server \fBNFS\fR information only.
.RE

.sp
.ne 2
.na
\fB\fB-r\fR\fR
.ad
.sp .6
.RS 4n
Display \fBRPC\fR information.
.RE

.sp
.ne 2
.na
\fB\fB-s\fR\fR
.ad
.sp .6
.RS 4n
Display server information.
.RE

.sp
.ne 2
.na
\fB\fB-T\fR \fBu\fR | \fBd\fR\fR
.ad
.sp .6
.RS 4n
Display a time stamp.
.sp
Specify \fBu\fR for a printed representation of the internal representation of
time. See \fBtime\fR(2). Specify \fBd\fR for standard date format. See
\fBdate\fR(1).
.RE
```

```
.sp
.ne 2
.na
\fB\fB-v\fR \fIversion\fR\fR
.ad
.sp .6
.RS 4n
Specify which NFS version for which to print statistics. When followed by the
optional \fIversion\fR argument, (\fB2\fR|\fB3\fR|\fB4\fR), specifies
statistics for that version. By default, prints statistics for all versions.
.RE

.sp
.ne 2
.na
\fB\fB-z\fR\fR
.ad
.sp .6
.RS 4n
Zero (reinitialize) statistics. This option is for use by the super user only,
and can be combined with any of the above options to zero particular sets of
statistics after printing them.
.RE

.SH OPERANDS
.sp
.LP
The following operands are supported:
.sp
.ne 2
.na
\fB\fIcount\fR\fR
.ad
.sp .6
.RS 4n
Display only count reports
.RE

.sp
.ne 2
.na
\fB\fIinterval\fR\fR
.ad
.sp .6
.RS 4n
Report once each interval seconds.
.RE

.sp
.ne 2
.na
\fB\fIpathname\fR\fR
.ad
.sp .6
.RS 4n
Specify the pathname of a file in an NFS mounted file system for which
statistics are to be displayed.
.RE

.SH DISPLAYS
.sp
.LP
The server \fBRPC\fR display includes the following fields:
.sp
.ne 2
```

```
.na
\fB\fBbadcalls\fR\fR
.ad
.sp .6
.RS 4n
The total number of calls rejected by the \fBRPC\fR layer (the sum of
\fBbadlen\fR and \fBxdrcall\fR as defined below).
.RE

.sp
.ne 2
.na
\fB\fBbadlen\fR\fR
.ad
.sp .6
.RS 4n
The number of \fBRPC\fR calls with a length shorter than a minimum-sized
\fBRPC\fR call.
.RE

.sp
.ne 2
.na
\fB\fBcalls\fR\fR
.ad
.sp .6
.RS 4n
The total number of \fBRPC\fR calls received.
.RE

.sp
.ne 2
.na
\fB\fBdupchecks\fR\fR
.ad
.sp .6
.RS 4n
The number of \fBRPC\fR calls that looked up in the duplicate request cache.
.RE

.sp
.ne 2
.na
\fB\fBdupreqs\fR\fR
.ad
.sp .6
.RS 4n
The number of \fBRPC\fR calls that were found to be duplicates.
.RE

.sp
.ne 2
.na
\fB\fBnullrecv\fR\fR
.ad
.sp .6
.RS 4n
The number of times an \fBRPC\fR call was not available when it was thought to
be received.
.RE

.sp
.ne 2
.na
\fB\fBxdrcall\fR\fR
.ad
```

```
.sp .6
.RS 4n
The number of \fBRPC\fR calls whose header could not be \fBXDR\fR decoded.
.RE

.sp
.LP
The server \fBNFS\fR display shows the number of \fBNFS\fR calls received
(\fBcalls\fR) and rejected (\fBbadcalls\fR), and the counts and percentages for
the various calls that were made.
.sp
.LP
The server \fBNFS_ACL\fR display shows the counts and percentages for the
various calls that were made.
.sp
.LP
The client \fBRPC\fR display includes the following fields:
.sp
.ne 2
.na
\fB\fBcalls\fR\fR
.ad
.sp .6
.RS 4n
The total number of \fBRPC\fR calls made.
.RE

.sp
.ne 2
.na
\fB\fBbadcalls\fR\fR
.ad
.sp .6
.RS 4n
The total number of calls rejected by the \fBRPC\fR layer.
.RE

.sp
.ne 2
.na
\fB\fBbadverfs\fR\fR
.ad
.sp .6
.RS 4n
The number of times the call failed due to a bad verifier in the response.
.RE

.sp
.ne 2
.na
\fB\fBbadxids\fR\fR
.ad
.sp .6
.RS 4n
The number of times a reply from a server was received which did not correspond
to any outstanding call.
.RE

.sp
.ne 2
.na
\fB\fBcantconn\fR\fR
.ad
.sp .6
.RS 4n
The number of times the call failed due to a failure to make a connection to
```

```
the server.
.RE

.sp
.ne 2
.na
\fB\fBcantsend\fR\fR
.ad
.sp .6
.RS 4n
The number of times a client was unable to send an \fBRPC\fR request over a
connectionless transport when it tried to do so.
.RE

.sp
.ne 2
.na
\fB\fBinterrupts\fR\fR
.ad
.sp .6
.RS 4n
The number of times the call was interrupted by a signal before completing.
.RE

.sp
.ne 2
.na
\fB\fBnewcreds\fR\fR
.ad
.sp .6
.RS 4n
The number of times authentication information had to be refreshed.
.RE

.sp
.ne 2
.na
\fB\fBnomem\fR\fR
.ad
.sp .6
.RS 4n
The number of times the call failed due to a failure to allocate memory.
.RE

.sp
.ne 2
.na
\fB\fBretrans\fR\fR
.ad
.sp .6
.RS 4n
The number of times a call had to be retransmitted due to a timeout while
waiting for a reply from the server. Applicable only to \fBRPC\fR over
connection-less transports.
.RE

.sp
.ne 2
.na
\fB\fBtimeouts\fR\fR
.ad
.sp .6
.RS 4n
The number of times a call timed out while waiting for a reply from the server.
.RE
```

```
.sp
.ne 2
.na
\fB\fBtimers\fR\fR
.ad
.sp .6
.RS 4n
The number of times the calculated time-out value was greater than or equal to
the minimum specified time-out value for a call.
.RE

.sp
.LP
The client \fBNFS\fR display shows the number of calls sent and rejected, as
well as the number of times a \fBCLIENT\fR handle was received (\fBclgets\fR),
the number of times the \fBCLIENT\fR handle cache had no unused entries
(\fBcltoomany\fR), as well as a count of the various calls and their respective
percentages.
.sp
.LP
The client \fBNFS_ACL\fR display shows the counts and percentages for the
various calls that were made.
.sp
.LP
The \fB-m\fR option includes information about mount flags set by mount
options, mount flags internal to the system, and other mount information. See
\fBmount_nfs\fR(1M).
.sp
.LP
The following mount flags are set by mount options:
.sp
.ne 2
.na
\fB\fBgrpid\fR\fR
.ad
.sp .6
.RS 4n
System V group id inheritance.
.RE

.sp
.ne 2
.na
\fB\fBhard\fR\fR
.ad
.sp .6
.RS 4n
Hard mount.
.RE

.sp
.ne 2
.na
\fB\fBintr\fR\fR
.ad
.sp .6
.RS 4n
Interrupts allowed on hard mount.
.RE

.sp
.ne 2
.na
\fB\fBllock\fR\fR
.ad
.sp .6
```

```
.RS 4n
Local locking being used (no lock manager).
.RE

.sp
.ne 2
.na
\fB\fBnoac\fR\fR
.ad
.sp .6
.RS 4n
Client is not caching attributes.
.RE

.sp
.ne 2
.na
\fB\fBnointr\fR\fR
.ad
.sp .6
.RS 4n
No interrupts allowed on hard mount.
.RE

.sp
.ne 2
.na
\fB\fBnocto\fR\fR
.ad
.sp .6
.RS 4n
No close-to-open consistency.
.RE

.sp
.ne 2
.na
\fB\fBretrans\fR\fR
.ad
.sp .6
.RS 4n
\fBNFS\fR retransmissions.
.RE

.sp
.ne 2
.na
\fB\fBrpctimesync\fR\fR
.ad
.sp .6
.RS 4n
\fBRPC\fR time sync.
.RE

.sp
.ne 2
.na
\fB\fBrsize\fR\fR
.ad
.sp .6
.RS 4n
Read buffer size in bytes.
.RE

.sp
.ne 2
```

```
.na
\fB\fBsec\fR\fR
.ad
.sp .6
.RS 4n
\fBsec\fR has one of the following values:
.sp
.ne 2
.na
\fB\fBdh\fR\fR
.ad
.sp .6
.RS 4n
\fBdes\fR-style authentication (encrypted timestamps).
.RE

.sp
.ne 2
.na
\fB\fBkrb5\fR\fR
.ad
.sp .6
.RS 4n
\fBkerberos v5\fR-style authentication.
.RE

.sp
.ne 2
.na
\fB\fBkrb5i\fR\fR
.ad
.sp .6
.RS 4n
\fBkerberos v5\fR-style authentication with integrity.
.RE

.sp
.ne 2
.na
\fB\fBkrb5p\fR\fR
.ad
.sp .6
.RS 4n
\fBkerberos v5\fR-style authentication with privacy.
.RE

.sp
.ne 2
.na
\fB\fBnone\fR\fR
.ad
.sp .6
.RS 4n
No authentication.
.RE

.sp
.ne 2
.na
\fB\fBshort\fR\fR
.ad
.sp .6
.RS 4n
Short hand UNIX-style authentication.
.RE
```

```
.sp
.ne 2
.na
\fB\fBsys\fR\fR
.ad
.sp .6
.RS 4n
UNIX-style authentication (UID, GID).
.RE

.RE

.sp
.ne 2
.na
\fB\fBsoft\fR\fR
.ad
.sp .6
.RS 4n
Soft mount.
.RE

.sp
.ne 2
.na
\fB\fBtimeo\fR\fR
.ad
.sp .6
.RS 4n
Initial \fBNFS\fR timeout, in tenths of a second.
.RE

.sp
.ne 2
.na
\fB\fBwsize\fR\fR
.ad
.sp .6
.RS 4n
Write buffer size in bytes.
.RE

.sp
.LP
The following mount flags are internal to the system:
.sp
.ne 2
.na
\fB\fBacl\fR\fR
.ad
.sp .6
.RS 4n
Server supports \fBNFS_ACL\fR.
.RE

.sp
.ne 2
.na
\fB\fBdown\fR\fR
.ad
.sp .6
.RS 4n
Server is down.
.RE

.sp
```

```
.ne 2
.na
\fB\fBdynamic\fR\fR
.ad
.sp .6
.RS 4n
Dynamic transfer size adjustment.
.RE

.sp
.ne 2
.na
\fB\fBlink\fR\fR
.ad
.sp .6
.RS 4n
Server supports links.
.RE

.sp
.ne 2
.na
\fB\fBmirrormount\fR\fR
.ad
.sp .6
.RS 4n
Mounted automatically by means of the \fBmirrormount\fR mechanism.
.RE

.sp
.ne 2
.na
\fB\fBprinted\fR\fR
.ad
.sp .6
.RS 4n
"Not responding" message printed.
.RE

.sp
.ne 2
.na
\fB\fBreaddir\fR\fR
.ad
.sp .6
.RS 4n
Use \fBreaddir\fR instead of \fBreaddirplus\fR.
.RE

.sp
.ne 2
.na
\fB\fBsymlink\fR\fR
.ad
.sp .6
.RS 4n
Server supports symbolic links.
.RE

.sp
.LP
The following flags relate to additional mount information:
.sp
.ne 2
.na
\fB\fBproto\fR\fR
```

```
.ad
.sp .6
.RS 4n
Protocol.
.RE

.sp
.ne 2
.na
\fB\fBvers\fR\fR
.ad
.sp .6
.RS 4n
\fBNFS\fR version.
.RE

.sp
.LP
The \fB-m\fR option also provides attribute cache timeout values. The following
fields in \fB-m\fR output provide timeout values for attribute cache:
fields in \fB-m\fR ouput provide timeout values for attribute cache:
.sp
.ne 2
.na
\fB\fBacdirmax\fR\fR
.ad
.sp .6
.RS 4n
Maximum seconds to hold cached directory attributes.
.RE

.sp
.ne 2
.na
\fB\fBacdirmin\fR\fR
.ad
.sp .6
.RS 4n
Minimum seconds to hold cached directory attributes.
.RE

.sp
.ne 2
.na
\fB\fBacregmax\fR\fR
.ad
.sp .6
.RS 4n
Maximum seconds to hold cached file attributes.
.RE

.sp
.ne 2
.na
\fB\fBacregmin\fR\fR
.ad
.sp .6
.RS 4n
Minimum seconds to hold cached file attributes.
.RE

.sp
.LP
The following fields in \fB-m\fR output provide failover information:
.sp
.ne 2
```

```
.na
\fB\fBcurrserver\fR\fR
.ad
.sp .6
.RS 4n
Which server is currently providing \fBNFS\fR service. See the \fI\fR for
additional details.
.RE

.sp
.ne 2
.na
\fB\fBfailover\fR\fR
.ad
.sp .6
.RS 4n
How many times a new server has been selected.
.RE

.sp
.ne 2
.na
\fB\fBnoresponse\fR\fR
.ad
.sp .6
.RS 4n
How many times servers have failed to respond.
.RE

.sp
.ne 2
.na
\fB\fBremap\fR\fR
.ad
.sp .6
.RS 4n
How many times files have been re-evaluated to the new server.
.RE

.sp
.LP
The fields in \fB-m\fR output shown below provide information on dynamic
retransmissions. These items are displayed only where dynamic retransmission is
in use.
.sp
.ne 2
.na
\fB\fBcur\fR\fR
.ad
.sp .6
.RS 4n
Current backed-off retransmission value, in milliseconds.
.RE

.sp
.ne 2
.na
\fB\fBdev\fR\fR
.ad
.sp .6
.RS 4n
Estimated deviation, in milliseconds.
.RE

.sp
.ne 2
```

```
.na
\fB\fBsrtt\fR\fR
.ad
.sp .6
.RS 4n
The value for the smoothed round-trip time, in milliseconds.
.RE

.SH EXIT STATUS
.sp
.LP
The following exit values are returned:
.sp
.ne 2
.na
\fB\fB0\fR\fR
.ad
.sp .6
.RS 4n
Successful completion.
.RE

.sp
.ne 2
.na
\fB\fB>0\fR\fR
.ad
.sp .6
.RS 4n
An error occurred.
.RE

.SH SEE ALSO
.sp
.LP
\fBmount_nfs\fR(1M), \fBattributes\fR(5)
.sp
.LP
\fI\fR
.sp
.LP
\fI\fR
```

```
***********************************************************
   13599 Tue Dec 11 09:48:09 2018
new/usr/src/man/man1m/smbadm.1m
10057 Man page misspellings ouput particuliar overriden
Reviewed by: Gerg¯  Mih^¡ly Doma <domag02@gmail.com>
***********************************************************
'\" te
.\" Copyright 2014 Nexenta Systems, Inc.  All rights reserved.
.\" Copyright (c) 2009, Sun Microsystems, Inc. All Rights Reserved.
.\" The contents of this file are subject to the terms of the Common Development
.\" You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE or http:
.\" When distributing Covered Code, include this CDDL HEADER in each file and in
.TH SMBADM 1M "April 9, 2016"
.SH NAME
smbadm \- configure and manage CIFS local groups and users, and manage domain
membership
.SH SYNOPSIS
.LP
.nf
\fBsmbadm add-member\fR -m \fImember\fR [[-m \fImember\fR] \&.\|.\|.] \fIgroup\f
.fi

.LP
.nf
\fBsmbadm create\fR [-d \fIdescription\fR] \fIgroup\fR
.fi

.LP
.nf
\fBsmbadm delete\fR \fIgroup\fR
.fi

.LP
.nf
\fBsmbadm disable-user\fR \fIusername\fR
.fi

.LP
.nf
\fBsmbadm enable-user\fR \fIusername\fR
.fi

.LP
.nf
\fBsmbadm get\fR [[-p \fIproperty\fR] \&.\|.\|.] \fIgroup\fR
.fi

.LP
.nf
\fBsmbadm join\fR [-y] -u \fIusername\fR \fIdomain\fR
.fi

.LP
.nf
\fBsmbadm join\fR [-y] -w \fIworkgroup\fR
.fi

.LP
.nf
\fBsmbadm list\fR
.fi

.LP
.nf
\fBsmbadm lookup\fR \fIaccount-name\fR [\fIaccount-name\fR [\&.\|.\|.]]
.fi
```

```
.LP
.nf
\fBsmbadm remove-member\fR -m \fImember\fR [[-m \fImember\fR] \&.\|.\|.] \fIgrou
.fi

.LP
.nf
\fBsmbadm rename\fR \fIgroup\fR \fInew-group\fR
.fi

.LP
.nf
\fBsmbadm set\fR -p \fIproperty\fR=\fIvalue\fR [[-p \fIproperty\fR=\fIvalue\fR]
.fi

.LP
.nf
\fBsmbadm show\fR [-m] [-p] [\fIgroup\fR]
.fi

.SH DESCRIPTION
.LP
The \fBsmbadm\fR command is used to configure \fBCIFS\fR local groups and to
manage domain membership. You can also use the \fBsmbadm\fR command to enable
or disable SMB password generation for individual local users.
.sp
.LP
\fBCIFS\fR local groups can be used when Windows accounts must be members of
some local groups and when Windows style privileges must be granted. Solaris
local groups cannot provide these functions.
.sp
.LP
There are two types of local groups: user defined and built-in. Built-in local
groups are predefined local groups to support common administration tasks.
.sp
.LP
In order to provide proper identity mapping between \fBCIFS\fR local groups and
Solaris groups, a \fBCIFS\fR local group must have a corresponding Solaris
group. This requirement has two consequences: first, the group name must
conform to the intersection of the Windows and Solaris group name rules. Thus,
a \fBCIFS\fR local group name can be up to eight (8) characters long and
contain only lowercase characters and numbers. Second, a Solaris local group
has to be created before a \fBCIFS\fR local group can be created.
.sp
.LP
Built-in groups are standard Windows groups and are predefined by the
\fBCIFS\fR service. The built-in groups cannot be added, removed, or renamed,
and these groups do not follow the \fBCIFS\fR local group naming conventions.
.sp
.LP
When the \fBCIFS\fR server is started, the following built-in groups are
available:
.sp
.ne 2
.na
\fBAdministrators\fR
.ad
.sp .6
.RS 4n
Group members can administer the system.
.RE

.sp
.ne 2
.na
```

```
\fBBackup Operators\fR
.ad
.sp .6
.RS 4n
Group members can bypass file access controls to back up and restore files.
.RE

.sp
.ne 2
.na
\fBPower Users\fR
.ad
.sp .6
.RS 4n
Group members can share directories.
.RE

.sp
.LP
Solaris local users must have an SMB password for authentication and to gain
access to CIFS resources. This password is created by using the \fBpasswd\fR(1)
command when the \fBpam_smb_password\fR module is added to the system’s PAM
configuration. See the \fBpam_smb_passwd\fR(5) man page.
.sp
.LP
The \fBdisable-user\fR and \fBenable-user\fR subcommands control SMB
password-generation for a specified local user. When disabled, the user is
prevented from connecting to the Solaris CIFS service. By default, SMB
password-generation is enabled for all local users.
.sp
.LP
To reenable a disabled user, you must use the \fBenable-user\fR subcommand and
then reset the user’s password by using the \fBpasswd\fR command. The
\fBpam_smb_passwd.so.1\fR module must be added to the system’s PAM
configuration to generate an SMB password.
.SS "Escaping Backslash Character"
.LP
For the \fBadd-member\fR, \fBremove-member\fR, and \fBjoin\fR (with \fB-u\fR)
subcommands, the backslash character (\fB\e\fR) is a valid separator between
member or user names and domain names. The backslash character is a shell
special character and must be quoted. For example, you might escape the
backslash character with another backslash character:
\fIdomain\fR\fB\e\e\fR\fIusername\fR. For more information about handling shell
special characters, see the man page for your shell.
.SH OPERANDS
.LP
The \fBsmbadm\fR command uses the following operands:
.sp
.ne 2
.na
\fB\fIdomain\fR\fR
.ad
.sp .6
.RS 4n
Specifies the name of an existing Windows domain to join.
.RE

.sp
.ne 2
.na
\fB\fIgroup\fR\fR
.ad
.sp .6
.RS 4n
Specifies the name of the \fBCIFS\fR local group.
.RE
```

```

.sp
.ne 2
.na
\fB\fIusername\fR\fR
.ad
.sp .6
.RS 4n
Specifies the name of a Solaris local user.
.RE

.SH SUBCOMMANDS
.LP
The \fBsmbadm\fR command includes these subcommands:
.sp
.ne 2
.na
\fB\fBadd-member\fR -m \fImember\fR [[-m \fImember\fR] \&.\|.\|.]
\fIgroup\fR\fR
.ad
.sp .6
.RS 4n
Adds the specified member to the specified \fBCIFS\fR local group. The \fB-m\fR
\fImember\fR option specifies the name of a \fBCIFS\fR local group member. The
member name must include an existing user name and an optional domain name.
.sp
Specify the member name in either of the following formats:
.sp
.in +2
.nf
[\fIdomain\fR\e]\fIusername\fR
[\fIdomain\fR/]\fIusername\fR
.fi
.in -2
.sp

For example, a valid member name might be \fBsales\eterry\fR or
\fBsales/terry\fR, where \fBsales\fR is the Windows domain name and \fBterry\fR
is the name of a user in the \fBsales\fR domain.
.RE

.sp
.ne 2
.na
\fB\fBcreate\fR [\fB-d\fR \fIdescription\fR] \fIgroup\fR\fR
.ad
.sp .6
.RS 4n
Creates a \fBCIFS\fR local group with the specified name. You can optionally
specify a description of the group by using the \fB-d\fR option.
.RE

.sp
.ne 2
.na
\fB\fBdelete\fR \fIgroup\fR\fR
.ad
.sp .6
.RS 4n
Deletes the specified \fBCIFS\fR local group. The built-in groups cannot be
deleted.
.RE

.sp
.ne 2
.na
```

```
\fB\fBdisable\fR \fIusername\fR\fR
.ad
.sp .6
.RS 4n
Disables SMB password-generation capabilities for the specified local user. A
disabled local user is prevented from accessing the system by means of the CIFS
service. When a local user account is disabled, you cannot use the \fBpasswd\fR
command to modify the user’s SMB password until the user account is reenabled.
.RE

.sp
.ne 2
.na
\fB\fBenable\fR \fIusername\fR\fR
.ad
.sp .6
.RS 4n
Enables SMB password-generation capabilities for the specified local user.
After the password-generation capabilities are reenabled, you must use the
\fBpasswd\fR command to generate the SMB password for the local user before he
can connect to the CIFS service.
.sp
The \fBpasswd\fR command manages both the Solaris password and SMB password for
this user if the \fBpam_smb_passwd\fR module has been added to the system’s PAM
configuration.
.RE

.sp
.ne 2
.na
\fB\fBget\fR [[\fB-p\fR \fIproperty\fR=\fIvalue\fR] \&.\|.\|.] \fIgroup\fR\fR
.ad
.sp .6
.RS 4n
Retrieves property values for the specified group. If no property is specified,
all property values are shown.
.RE

.sp
.ne 2
.na
\fB\fBjoin\fR \fB[-y] -u\fR \fIusername\fR \fIdomain\fR\fR
.ad
.sp .6
.RS 4n
Joins a Windows domain or a workgroup.
.sp
The default mode for the \fBCIFS\fR service is workgroup mode, which uses the
default workgroup name, \fBWORKGROUP\fR.
.sp
An authenticated user account is required to join a domain, so you must specify
the Windows administrative user name with the \fB-u\fR option. If the password
is not specified on the command line, the user is prompted for it. This user
should be the domain administrator or any user who has administrative
privileges for the target domain.
.sp
\fIusername\fR and \fIdomain\fR can be entered in any of the following formats:
.sp
.in +2
.nf
\fIusername\fR[+\fIpassword\fR] \fIdomain\fR
\fIdomain\fR\e\fIusername\fR[+\fIpassword\fR]
\fIdomain\fR/\fIusername\fR[+\fIpassword\fR]
\fIusername\fR@\fIdomain\fR
.fi
.in -2
```

```
.sp

\&...where \fIdomain\fR can be the NetBIOS or DNS domain name.
.sp
If a machine trust account for the system already exists on a domain
controller, any authenticated user account can be used when joining the domain.
However, if the machine trust account does \fBnot\fR already exist, an account
that has administrative privileges on the domain is required to join the
domain.
Specifying \fB-y\fR will bypass the smb service restart prompt.
.RE

.sp
.ne 2
.na
\fB\fBjoin\fR \fB[-y] -w\fR \fIworkgroup\fR\fR
.ad
.sp .6
.RS 4n
Joins a Windows domain or a workgroup.
.sp
The \fB-w\fR \fIworkgroup\fR option specifies the name of the workgroup to join
when using the \fBjoin\fR subcommand.
Specifying \fB-y\fR will bypass the smb service restart prompt.
.RE

.sp
.ne 2
.na
\fB\fBlist\fR\fR
.ad
.sp .6
.RS 4n
Shows information about the current workgroup or domain. The information
typically includes the workgroup name or the primary domain name. When in
domain mode, the information includes domain controller names and trusted
domain names.
.sp
Each entry in the output is identified by one of the following tags:
Each entry in the ouput is identified by one of the following tags:
.sp
.ne 2
.na
\fB\fB- [*] -\fR\fR
.ad
.RS 11n
Primary domain
.RE

.sp
.ne 2
.na
\fB\fB- [.] -\fR\fR
.ad
.RS 11n
Local domain
.RE

.sp
.ne 2
.na
\fB\fB- [-] -\fR\fR
.ad
.RS 11n
Other domains
.RE
```

```
.sp
.ne 2
.na
\fB\fB- [+] -\fR\fR
.ad
.RS 11n
Selected domain controller
.RE

.RE

.sp
.ne 2
.na
\fB\fBlookup\fR\fR \fIaccount-name\fR [\fIaccount-name\fR [\&.\|.\|.]]

.ad
.sp .6
.RS 4n
Lookup the SID for the given \fIaccount-name\fR, or lookup the
\fIaccount-name\fR for the given SID.  This subcommand is
primarily for diagnostic use, to confirm whether the server
can lookup domain accounts and/or SIDs.
.RE

.sp
.ne 2
.na
\fB\fBremove-member\fR -m \fImember\fR [[-m \fImember\fR] \&.\|.\|.]
\fIgroup\fR\fR
.ad
.sp .6
.RS 4n
Removes the specified member from the specified \fBCIFS\fR local group. The
\fB-m\fR \fImember\fR option specifies the name of a \fBCIFS\fR local group
member. The member name must include an existing user name and an optional
domain name.
.sp
Specify the member name in either of the following formats:
.sp
.in +2
.nf
[\fIdomain\fR\e]\fIusername\fR
[\fIdomain\fR/]\fIusername\fR
.fi
.in -2
.sp

For example, a valid member name might be \fBsales\eterry\fR or
\fBsales/terry\fR, where \fBsales\fR is the Windows domain name and \fBterry\fR
is the name of a user in the \fBsales\fR domain.
.RE

.sp
.ne 2
.na
\fB\fBrename\fR \fIgroup\fR \fInew-group\fR\fR
.ad
.sp .6
.RS 4n
Renames the specified \fBCIFS\fR local group. The group must already exist. The
built-in groups cannot be renamed.
.RE

.sp
```

```
.ne 2
.na
\fB\fBset\fR \fB-p\fR \fIproperty\fR=\fIvalue\fR [[\fB-p\fR
\fIproperty\fR=\fIvalue\fR] \&.\|.\|.] \fIgroup\fR\fR
.ad
.sp .6
.RS 4n
Sets configuration properties for a \fBCIFS\fR local group. The description and
the privileges for the built-in groups cannot be changed.
.sp
The \fB-p\fR \fIproperty\fR\fB=\fR\fIvalue\fR option specifies the list of
properties to be set on the specified group.
.sp
The group-related properties are as follows:
.sp
.ne 2
.na
\fB\fBbackup=[on|off]\fR\fR
.ad
.sp .6
.RS 4n
Specifies whether members of the \fBCIFS\fR local group can bypass file access
controls to back up file system objects.
.RE

.sp
.ne 2
.na
\fB\fBdescription=\fR\fIdescription-text\fR\fR
.ad
.sp .6
.RS 4n
Specifies a text description for the \fBCIFS\fR local group.
.RE

.sp
.ne 2
.na
\fB\fBrestore=[on|off]\fR\fR
.ad
.sp .6
.RS 4n
Specifies whether members of the \fBCIFS\fR local group can bypass file access
controls to restore file system objects.
.RE

.sp
.ne 2
.na
\fB\fBtake-ownership=[on|off]\fR\fR
.ad
.sp .6
.RS 4n
Specifies whether members of the \fBCIFS\fR local group can take ownership of
file system objects.
.RE

.RE

.sp
.ne 2
.na
\fB\fBshow\fR [\fB-m\fR] [\fB-p\fR] [\fIgroup\fR]\fR
.ad
.sp .6
.RS 4n
```

```
Shows information about the specified \fBCIFS\fR local group or groups. If no
group is specified, information is shown for all groups. If the \fB-m\fR option
is specified, the group members are also shown. If the \fB-p\fR option is
specified, the group privileges are also shown.
.RE

.SH EXIT STATUS
.LP
The following exit values are returned:
.sp
.ne 2
.na
\fB0\fR
.ad
.RS 13n
Successful completion.
.RE

.sp
.ne 2
.na
\fB>0\fR
.ad
.RS 13n
An error occurred.
.RE

.SH ATTRIBUTES
.LP
See the \fBattributes\fR(5) man page for descriptions of the following
attributes:
.sp

.sp
.TS
box;
c | c
l | l .
ATTRIBUTE TYPE  ATTRIBUTE VALUE
_
Utility Name and Options        Uncommitted
_
Utility Output Format   Not-An-Interface
_
\fBsmbadm join\fR       Obsolete
.TE

.SH SEE ALSO
.LP
\fBpasswd\fR(1), \fBgroupadd\fR(1M), \fBidmap\fR(1M), \fBidmapd\fR(1M),
\fBkclient\fR(1M), \fBshare\fR(1M), \fBsharectl\fR(1M), \fBsharemgr\fR(1M),
\fBsmbd\fR(1M), \fBsmbstat\fR(1M), \fBsmb\fR(4), \fBsmbautohome\fR(4),
\fBattributes\fR(5), \fBpam_smb_passwd\fR(5), \fBsmf\fR(5)
```

```
**********************************************************
    4185 Tue Dec 11 09:48:09 2018
new/usr/src/man/man1m/uucleanup.1m
10057 Man page misspellings ouput particuliar overriden
Reviewed by: Gerg¯  Mih^¡ly Doma <domag02@gmail.com>
**********************************************************
'\" te
.\" Copyright (c) 1996, Sun Microsystems, Inc. All Rights Reserved.
.\" Copyright 1989 AT&T
.\" The contents of this file are subject to the terms of the Common Development
.\" You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE or http:
.\" When distributing Covered Code, include this CDDL HEADER in each file and in
.TH UUCLEANUP 1M "May 19, 1993"
.SH NAME
uucleanup \- uucp spool directory clean-up
.SH SYNOPSIS
.LP
.nf
\fB/usr/lib/uucp/uucleanup\fR [\fB-C\fR\fItime\fR] [\fB-D\fR\fItime\fR] [\fB-m\f
     [\fB-o\fR\fItime\fR] [\fB-s\fR\fIsystem\fR] [\fB-W\fR\fItime\fR] [\fB-x\fR\
.fi

.SH DESCRIPTION
.sp
.LP
\fBuucleanup\fR will scan the spool directories for old files and take
appropriate action to remove them in a useful way:
.RS +4
.TP
.ie t \(bu
.el o
Inform the requester of send/receive requests for systems that can not be
reached.
.RE
.RS +4
.TP
.ie t \(bu
.el o
Return undeliverable mail to the sender.
.RE
.RS +4
.TP
.ie t \(bu
.el o
Deliver \fBrnews\fR files addressed to the local system.
.RE
.RS +4
.TP
.ie t \(bu
.el o
Remove all other files.
.RE
.sp
.LP
In addition, there is a provision to warn users of requests that have been
waiting for a given number of days (default 1 day). Note: \fBuucleanup\fR will
process as if all option \fBtimes\fR were specified to the default values
unless \fBtime\fR is specifically set.
.sp
.LP
This program is typically started by the shell \fBuudemon.cleanup\fR, which
should be started by \fBcron\fR(1M).
.SH OPTIONS
.sp
.ne 2
.na
```

```
\fB\fB-C\fR\fBtime\fR\fR
.ad
.RS 17n
Remove any \fBC.\fR files greater or equal to \fBtime\fR days old and send
appropriate information to the requester (default 7 days).
.RE

.sp
.ne 2
.na
\fB\fB-D\fR\fBtime\fR\fR
.ad
.RS 17n
Remove any \fBD.\fR files greater or equal to \fBtime\fR days old, make an
attempt to deliver mail messages, and execute  \fBrnews\fR when appropriate
(default 7 days).
.RE

.sp
.ne 2
.na
\fB\fB-m\fR\fIstring\fR\fR
.ad
.RS 17n
Include \fIstring\fR in the warning message generated by the \fB-W\fR option.
The default line is "See your local administrator to locate the problem".
.RE

.sp
.ne 2
.na
\fB\fB-o\fR\fBtime\fR\fR
.ad
.RS 17n
Delete other files whose age is more than \fBtime\fR days (default 2 days).
.RE

.sp
.ne 2
.na
\fB\fB-s\fR\fIsystem\fR\fR
.ad
.RS 17n
Execute for \fIsystem\fR spool directory only.
.RE

.sp
.ne 2
.na
\fB\fB-W\fR\fBtime\fR\fR
.ad
.RS 17n
Any \fBC.\fR files equal to \fBtime\fR days old will cause a mail message to be
sent to the requester warning about the delay in contacting the remote. The
message includes the \fIJOBID\fR, and in the case of mail, the mail message.
The administrator may include a message line telling whom to call to check the
problem (\fB-m\fR option) (default 1 day).
.RE

.sp
.ne 2
.na
\fB\fB-x\fR\fIdebug-level\fR\fR
.ad
.RS 17n
Produce debugging output on standard output. \fIdebug-level\fR is a single digit
```

```
*Produce debugging output on standard ouput. \fIdebug-level\fR is a single digit*
between 0 and 9; higher numbers give more detailed debugging information. (This
option may not be available on all systems.)
.RE

.sp
.ne 2
.na
\fB\fB-X\fR\fBtime\fR\fR
.ad
.RS 17n
Any \fBX.\fR files greater or equal to \fBtime\fR days old will be removed. The
\fBD.\fR files are probably not present (if they were, the \fBX.\fR could get
executed). But if there are \fBD.\fR files, they will be taken care of by D.
processing (default 2 days).
.RE

.SH FILES
*.sp*
.ne 2
.na
\fB\fB/usr/lib/uucp\fR\fR
.ad
.RS 19n
directory with commands used by \fBuucleanup\fR internally
.RE

.sp
.ne 2
.na
\fB\fB/var/spool/uucp\fR\fR
.ad
.RS 19n
spool directory
.RE

.SH SEE ALSO
*.sp*
.LP
\fBuucp\fR(1C), \fBuux\fR(1C), \fBcron\fR(1M), \fBattributes\fR(5)
```

```
**********************************************************
    6144 Tue Dec 11 09:48:09 2018
new/usr/src/man/man3lgrp/lgrp_affinity_get.3lgrp
10057 Man page misspellings ouput particuliar overriden
Reviewed by: Gerg¯  Mih^¡ly Doma <domag02@gmail.com>
**********************************************************
'\" te
.\" Copyright (c) 2003, Sun Microsystems, Inc. All Rights Reserved.
.\" The contents of this file are subject to the terms of the Common Development
.\" You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE or http:
.\" When distributing Covered Code, include this CDDL HEADER in each file and in
.TH LGRP_AFFINITY_GET 3LGRP "Apr 16, 2003"
.SH NAME
lgrp_affinity_get, lgrp_affinity_set \- get of set lgroup affinity
.SH SYNOPSIS
.LP
.nf
cc [ \fIflag \&.\|.\|.\fR ] \fIfile\fR\&.\|.\|. \fB-llgrp\fR [ \fIlibrary \&.\|.
#include <sys/lgrp_user.h>

\fBlgrp_affinity_t\fR \fBlgrp_affinity_get\fR(\fBidtype_t\fR \fIidtype\fR, \fBid
     \fBlgrp_id_t\fR \fIlgrp\fR);
.fi

.LP
.nf
\fBint\fR \fBlgrp_affinity_set\fR(\fBidtype_t\fR \fIidtype\fR, \fBid_t\fR \fIid\
     \fBlgrp_affinity_t\fR \fIaffinity\fR);
.fi

.SH DESCRIPTION
.sp
.LP
The \fBlgrp_affinity_get()\fR function returns the affinity that the LWP or set
of LWPs specified by the \fIidtype\fR and \fIid\fR arguments have for the given
lgroup.
.sp
.LP
The \fBlgrp_affinity_set()\fR function sets the affinity that the LWP or set of
LWPs specified by \fIidtype\fR and \fIid\fR have for the given lgroup.  The
lgroup affinity can be set to \fBLGRP_AFF_STRONG\fR, \fBLGRP_AFF_WEAK\fR, or
\fBLGRP_AFF_NONE\fR.
.sp
.LP
If the \fIidtype\fR is \fBP_PID\fR, the affinity is retrieved for one of the
LWPs in the process or set for all the LWPs of the process with process ID
(PID) \fIid\fR. The affinity is retrieved or set for the LWP of the current
process with LWP ID \fIid\fR if \fIidtype\fR is \fBP_LWPID\fR.  If \fIid\fR is
\fBP_MYID\fR, then the current LWP or process is specified.
.sp
.LP
The operating system uses the lgroup affinities as advice on where to run a
thread and allocate its memory and factors this advice in with other
constraints.  Processor binding and processor sets can restrict which lgroups a
thread can run on, but do not change the lgroup affinities.
.sp
.LP
Each thread can have an affinity for an lgroup in the system such that the
thread will tend to be scheduled to run on that lgroup and allocate memory from
there whenever possible.  If the thread has affinity for more than one lgroup,
the operating system will try to run the thread and allocate its memory on the
lgroup for which it has the strongest affinity, then the next strongest, and so
on up through some small, system-dependent number of these lgroup affinities.
When multiple lgroups have the same affinity, the order of preference among
them is unspecified and up to the operating system to choose.  The lgroup with
the strongest affinity that the thread can run on is known as its "home lgroup"
```

```
(see \fBlgrp_home\fR(3LGRP)) and is usually the operating system's first choice
of where to run the thread and allocate its memory.
.sp
.LP
There are different levels of affinity that can be specified by a thread for a
particular lgroup.  The levels of affinity are the following from strongest to
 There are different levels of affinity that can be specified by a thread for a
particuliar lgroup.  The levels of affinity are the following from strongest to
weakest:
.sp
.in +2
.nf
LGRP_AFF_STRONG         /* strong affinity */
LGRP_AFF_WEAK           /* weak affinity */
LGRP_AFF_NONE           /* no affinity */
.fi
.in -2

.sp
.LP
The \fBLGRP_AFF_STRONG\fR affinity serves as a hint to the operating system
that the calling thread has a strong affinity for the given lgroup.  If this is
the thread's home lgroup, the operating system will avoid rehoming it to
another lgroup if possible.  However, dynamic reconfiguration, processor
offlining, processor binding, and processor set binding and manipulation are
examples of events that can cause the operating system to change the thread's
home lgroup for which it has a strong affinity.
.sp
.LP
The \fBLGRP_AFF_WEAK\fR affinity is a hint to the operating system that the
calling thread has a weak affinity for the given lgroup.  If a thread has a
weak affinity for its home lgroup, the operating system interpets this to mean
that thread does not mind whether it is rehomed, unlike \fBLGRP_AFF_STRONG\fR.
Load balancing, dynamic reconfiguration, processor binding, or processor set
binding and manipulation are examples of events that can cause the operating
system to change a thread's home lgroup for which it has a weak affinity.
.sp
.LP
The \fBLGRP_AFF_NONE\fR affinity signifies no affinity and can be used to
remove a thread's affinity for a particular lgroup.  Initially, each thread
remove a thread's affinity for a particuliar lgroup.  Initially, each thread
has no affinity to any lgroup.  If a thread has no lgroup affinities set, the
operating system chooses a home lgroup for the thread with no affinity set.
.SH RETURN VALUES
.sp
.LP
Upon successful completion, \fBlgrp_affinity_get()\fR returns the affinity for
the given lgroup.
.sp
.LP
Upon successful completion, \fBlgrp_affinity_set()\fR return 0.
.sp
.LP
Otherwise, both functions return \(mi1 and set \fBerrno\fR to indicate the
error.
.SH ERRORS
.sp
.LP
The \fBlgrp_affinity_get()\fR and \fBlgrp_affinity_set()\fR functions will fail
if:
.sp
.ne 2
.na
\fB\fBEINVAL\fR\fR
.ad
.RS 10n
```

```
The specified lgroup, affinity, or ID type is not valid.
.RE

.sp
.ne 2
.na
\fB\fBEPERM\fR\fR
.ad
.RS 10n
The effective user of the calling process does not have appropriate privileges,
and its real or effective user ID does not match the real or effective user ID
of one of the LWPs.
.RE

.sp
.ne 2
.na
\fB\fBESRCH\fR\fR
.ad
.RS 10n
The specified lgroup or LWP(s) was not found.
.RE

.SH ATTRIBUTES
.sp
.LP
See \fBattributes\fR(5) for descriptions of the following attributes:
.sp

.sp
.TS
box;
c | c
l | l .
ATTRIBUTE TYPE  ATTRIBUTE VALUE
_
Interface Stability     Evolving
_
MT-Level        MT-Safe
.TE

.SH SEE ALSO
.sp
.LP
\fBlgrp_home\fR(3LGRP), \fBliblgrp\fR(3LIB), \fBattributes\fR(5)
```

```
**********************************************************
   26503 Tue Dec 11 09:48:09 2018
new/usr/src/man/man3perl/Lgrp.3perl
10057 Man page misspellings ouput particuliar overriden
Reviewed by: Gerg¯  Mih^¡ly Doma <domag02@gmail.com>
**********************************************************
'\" te
.\" Copyright (c) 2008, Sun Microsystems, Inc. All Rights Reserved.
.\" The contents of this file are subject to the terms of the Common Development
.\" You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE or http:
.\" When distributing Covered Code, include this CDDL HEADER in each file and in
.TH LGRP 3PERL "April 9, 2016"
.SH NAME
Lgrp \- Perl interface to Solaris liblgrp library
.SH SYNOPSIS
.LP
.nf
use Sun::Solaris::Lgrp qw(:ALL);

# initialize lgroup interface
my $cookie = lgrp_init(LGRP_VIEW_OS | LGRP_VIEW_CALLER);
my $l = Sun::Solaris::Lgrp->new(LGRP_VIEW_OS |
     LGRP_VIEW_CALLER);

my $version = lgrp_version(LGRP_VER_CURRENT | LGRP_VER_NONE);
$version = $l->version(LGRP_VER_CURRENT | LGRP_VER_NONE);

$home = lgrp_home(P_PID, P_MYID);
$home = l->home(P_PID, P_MYID);

lgrp_affinity_set(P_PID, $\fIpid\fR, $\fIlgrp\fR,
      LGRP_AFF_STRONG | LGRP_AFF_WEAK | LGRP_AFF_NONE);
$l->affinity_set(P_PID, $\fIpid\fR, $\fIlgrp\fR,
      LGRP_AFF_STRONG | LGRP_AFF_WEAK | LGRP_AFF_NONE);

my $affinity = lgrp_affinity_get(P_PID, $\fIpid\fR, $\fIlgrp\fR);
$affinity = $l->affinity_get(P_PID, $\fIpid\fR, $\fIlgrp\fR);

my $nlgrps = lgrp_nlgrps($\fIcookie\fR);
$nlgrps = $l->nlgrps();

my $root = lgrp_root($\fIcookie\fR);
$root = l->root();

$latency = lgrp_latency($\fIlgrp1\fR, $\fIlgrp2\fR);
$latency = $l->latency($\fIlgrp1\fR, $\fIlgrp2\fR);

my @children = lgrp_children($\fIcookie\fR, $\fIlgrp\fR);
@children = l->children($lgrp);

my @parents = lgrp_parents($\fIcookie\fR, $\fIlgrp\fR);
@parents = l->parents($\fIlgrp\fR);

my @lgrps = lgrp_lgrps($\fIcookie\fR);
@lgrps = l->lgrps();

@lgrps = lgrp_lgrps($\fIcookie\fR, $\fIlgrp\fR);
@lgrps = l->lgrps($\fIlgrp\fR);

my @leaves = lgrp_leaves($\fIcookie\fR);
@leaves = l->leaves();

my $is_leaf = lgrp_isleaf($\fIcookie\fR, $\fIlgrp\fR);
$is_leaf = $l->is_leaf($\fIlgrp\fR);

my @cpus = lgrp_cpus($\fIcookie\fR, $\fIlgrp\fR,
```

```
     LGRP_CONTENT_HIERARCHY | LGRP_CONTENT_DIRECT);
@cpus = l->cpus($\fIlgrp\fR, LGRP_CONTENT_HIERARCHY |
     LGRP_CONTENT_DIRECT);

my $memsize = lgrp_mem_size($\fIcookie\fR, $\fIlgrp\fR,
      LGRP_MEM_SZ_INSTALLED | LGRP_MEM_SZ_FREE,
      LGRP_CONTENT_HIERARCHY | LGRP_CONTENT_DIRECT);
$memsize = l->mem_size($\fIlgrp\fR,
      LGRP_MEM_SZ_INSTALLED | LGRP_MEM_SZ_FREE,
      LGRP_CONTENT_HIERARCHY | LGRP_CONTENT_DIRECT);

my $is_stale = lgrp_cookie_stale($\fIcookie\fR);
$stale = l->stale();

lgrp_fini($\fIcookie\fR);

# The following is available for API version greater than 1:
my @lgrps = lgrp_resources($\fIcookie\fR, $\fIlgrp\fR, LGRP_RSRC_CPU);

# Get latencies from cookie
$latency = lgrp_latency_cookie($\fIcookie\fR, $\fIfrom\fR, $\fIto\fR);
.fi

.SH DESCRIPTION
.LP
This module provides access to the \fBliblgrp\fR(3LIB) library and to various
constants and functions defined in <\fBsys/lgrp_sys.h\fR>. It provides both the
procedural and object interface to the library. The procedural interface
requires (in most cases) passing around a transparent cookie. The object
interface hides all the cookie manipulations from the user.
.sp
.LP
Functions returning a scalar value indicate an error by returning \fBundef\fR.
The caller can examine the \fB$!\fR variable to get the error value.
.sp
.LP
Functions returning a list value return the number of elements in the list when
called in scalar context. In the event of error, the empty list is returned in
the array context and \fBundef\fR is returned in the scalar context.
.SS "Constants"
.LP
The constants are exported with \fB:CONSTANTS\fR or \fB:ALL\fR tags:
.sp
.in +2
.nf
use Sun::Solaris::Lgrp ':ALL';
.fi
.in -2

.sp
.LP
or
.sp
.in +2
.nf
use Sun::Solaris::Lgrp ':CONSTANTS';
.fi
.in -2

.sp
.LP
The following constants are available for use in Perl programs:
.br
.in +2
\fBLGRP_NONE\fR
.in -2
```

```
.br
.in +2
\fBLGRP_VER_CURRENT\fR
.in -2
.br
.in +2
\fBLGRP_VER_NONE\fR
.in -2
.br
.in +2
\fBLGRP_VIEW_CALLER\fR
.in -2
.br
.in +2
\fBLGRP_VIEW_OS\fR
.in -2
.br
.in +2
\fBLGRP_AFF_NONE\fR
.in -2
.br
.in +2
\fBLGRP_AFF_STRONG\fR
.in -2
.br
.in +2
\fBLGRP_AFF_WEAK\fR
.in -2
.br
.in +2
\fBLGRP_CONTENT_DIRECT\fR
.in -2
.br
.in +2
\fBLGRP_CONTENT_HIERARCHY\fR
.in -2
.br
.in +2
\fBLGRP_MEM_SZ_FREE\fR
.in -2
.br
.in +2
\fBLGRP_MEM_SZ_FREE\fR
.in -2
.br
.in +2
\fBLGRP_RSRC_CPU\fR (1)
.in -2
.br
.in +2
\fBLGRP_RSRC_MEM\fR (1)
.in -2
.br
.in +2
\fBLGRP_CONTENT_ALL\fR (1)
.in -2
.br
.in +2
\fBLGRP_LAT_CPU_TO_MEM\fR (1)
.in -2
.br
.in +2
\fBP_PID\fR
.in -2
.br
.in +2
```

```
\fBP_LWPID\fR
.in -2
.br
.in +2
\fBP_MYID\fR
.in -2
.sp
.LP
(1) Available for versions of the \fBliblgrp\fR(3LIB) API greater than 1.
.SS "Functions"
.LP
A detailed description of each function follows. Since this module is intended
to provide a Perl interface to the functions in \fBliblgrp\fR(3LIB), a very
short description is given for the corresponding functions in this module and a
reference is given to the complete description in the \fBliblgrp\fR manual
pages. Any differences or additional functionality in the Perl module are
highlighted and fully documented here.
.sp
.ne 2
.na
\fB\fBlgrp_init([LGRP_VIEW_CALLER | LGRP_VIEW_OS])\fR\fR
.ad
.sp .6
.RS 4n
This function initializes the lgroup interface and takes a snapshot of the
lgroup hierarchy with the given view. Given the view, \fBlgrp_init()\fR returns
a cookie representing this snapshot of the lgroup hierarchy. This cookie should
be used with other routines in the lgroup interface needing the lgroup
hierarchy. The \fBlgrp_fini()\fR function should be called with the cookie when
it is no longer needed. Unlike \fBlgrp_init\fR(3LGRP), \fBLGRP_VIEW_OS\fR is
assumed as the default if no view is provided.
.sp
Upon successful completion, \fBlgrp_init()\fR returns a cookie. Otherwise it
returns \fBundef\fR and sets \fB$!\fR to indicate the error.
.sp
See \fBlgrp_init\fR(3LGRP) for more information.
.RE

.sp
.ne 2
.na
\fB\fBlgrp_fini\fR($\fIcookie\fR)\fR
.ad
.sp .6
.RS 4n
This function takes a cookie, frees the snapshot of the lgroup hierarchy
created by \fBlgrp_init()\fR, and cleans up anything else set up by
\fBlgrp_init()\fR. After this function is called, the cookie returned by the
lgroup interface might no longer be valid and should not be used.
.sp
Upon successful completion, 1 is returned. Otherwise, \fBundef\fR is returned
and \fB$!\fR is set to indicate the error.
.sp
See \fBlgrp_fini\fR(3LGRP) for more information.
.RE

.sp
.ne 2
.na
\fB\fBlgrp_view\fR($\fIcookie\fR)\fR
.ad
.sp .6
.RS 4n
This function takes a cookie representing the snapshot of the lgroup hierarchy
and returns the snapshot’s view of the lgroup hierarchy.
.sp
```

```
If the given view is \fBLGRP_VIEW_CALLER\fR, the snapshot contains only the
resources that are available to the caller (such as those with respect to
processor sets). When the view is \fBLGRP_VIEW_OS\fR, the snapshot contains
what is available to the operating system.
.sp
Upon successful completion, the function returns the view for the snapshot of
the lgroup hierarchy represented by the given cookie.  Otherwise, \fBundef\fR
is returned and \fB$!\fR is set to indicate the error.
.sp
See \fBlgrp_view\fR(3LGRP) for more information.
.RE

.sp
.ne 2
.na
\fB\fBlgrp_home\fR($\fIidtype\fR, $\fIid\fR)\fR
.ad
.sp .6
.RS 4n
This function returns the home lgroup for the given process or thread. The
$\fIidtype\fR argument should be \fBP_PID\fR to specify a process and the
$\fIid\fR argument should be its process ID. Otherwise, the $\fIidtype\fR
argument should be \fBP_LWPID\fR to specify a thread and the $\fIid\fR argument
should be its LWP ID. The value \fBP_MYID\fR can be used for the $\fIid\fR
argument to specify the current process or thread.
.sp
Upon successful completion, \fBlgrp_home()\fR returns the ID of the home lgroup
of the specified process or thread. Otherwise, \fBundef\fR is returned and
\fB$!\fR is set to indicate the error.
.sp
See \fBlgrp_home\fR(3LGRP) for more information.
.RE

.sp
.ne 2
.na
\fB\fBlgrp_cookie_stale\fR($\fIcookie\fR)\fR
.ad
.sp .6
.RS 4n
Upon successful completion, this function returns whether the cookie is stale.
Otherwise, it returns \fBundef\fR and sets \fB$!\fR to indicate the error.
.sp
The \fBlgrp_cookie_stale()\fR function will fail with \fBEINVAL\fR if the
cookie is not valid.
.sp
See \fBlgrp_cookie_stale\fR(3LGRP) for more information.
.RE

.sp
.ne 2
.na
\fB\fBlgrp_cpus\fR($\fIcookie\fR, $\fIlgrp\fR, $\fIcontext\fR)\fR
.ad
.sp .6
.RS 4n
This function takes a cookie representing a snapshot of the lgroup hierarchy
and returns the list of CPUs in the lgroup specified by $\fIlgrp\fR. The
$\fIcontext\fR argument should be set to one of the following values to specify
whether the direct contents or everything in this lgroup including its children
should be returned:
.sp
.ne 2
.na
\fB\fBLGRP_CONTENT_HIERARCHY\fR\fR
.ad
```

```
.RS 26n
everything within this hierarchy
.RE

.sp
.ne 2
.na
\fB\fBLGRP_CONTENT_DIRECT\fR\fR
.ad
.RS 26n
directly contained in lgroup
.RE

When called in scalar context, \fBlgrp_cpus()\fR function returns the number of
CPUs contained in the specified lgroup.
.sp
In the event of error, \fBundef\fR is returned in scalar context and \fB$!\fR
is set to indicate the error. In list context, the empty list is returned and
\fB$!\fR is set.
.sp
See \fBlgrp_cpus\fR(3LGRP) for more information.
.RE

.sp
.ne 2
.na
\fB\fBlgrp_children\fR($\fIcookie\fR, $\fIlgrp\fR)\fR
.ad
.sp .6
.RS 4n
This function takes a cookie representing a snapshot of the lgroup hierarchy
and returns the list of lgroups that are children of the specified lgroup.
.sp
When called in scalar context, \fBlgrp_children()\fR returns the number of
children lgroups for the specified lgroup.
.sp
In the event of error, \fBundef\fR or empty list is returned and \fB$!\fR is
set to indicate the error.
.sp
See \fBlgrp_children\fR(3LGRP) for more information.
.RE

.sp
.ne 2
.na
\fB\fBlgrp_parents\fR($\fIcookie\fR, $\fIlgrp\fR)\fR
.ad
.sp .6
.RS 4n
This function takes a cookie representing a snapshot of the lgroup hierarchy
and returns the list of parents of the specified lgroup.
.sp
When called in scalar context, \fBlgrp_parents()\fR returns the number of
parent lgroups for the specified lgroup.
.sp
In the event of error, \fBundef\fR or an empty list is returned and \fB$!\fR is
set to indicate the error.
.sp
See \fBlgrp_parents\fR(3LGRP) for more information.
.RE

.sp
.ne 2
.na
\fB\fBlgrp_nlgrps\fR($\fIcookie\fR)\fR
.ad
```

```
.sp .6
.RS 4n
This function takes a cookie representing a snapshot of the lgroup hierarchy.
It returns the number of lgroups in the hierarchy, where the number is always
at least one.
.sp
In the event of error, \fBundef\fR is returned and \fB$!\fR is set to
\fBEINVAL\fR, indicating that the cookie is not valid.
.sp
See \fBlgrp_nlgrps\fR(3LGRP) for more information.
.RE

.sp
.ne 2
.na
\fB\fBlgrp_root\fR($\fIcookie\fR)\fR
.ad
.sp .6
.RS 4n
This function returns the root lgroup ID.
.sp
In the event of error, \fBundef\fR is returned and \fB$!\fR is set to
\fBEINVAL\fR, indicatng that the cookie is not valid.
.sp
See \fBlgrp_root\fR(3LGRP) for more information.
.RE

.sp
.ne 2
.na
\fB\fBlgrp_mem_size\fR($\fIcookie\fR, $\fIlgrp\fR, $\fItype\fR,
$\fIcontent\fR)\fR
.ad
.sp .6
.RS 4n
This function takes a cookie representing a snapshot of the lgroup hierarchy.
The function returns the memory size of the given lgroup in bytes. The
$\fItype\fR argument should be set to one of the following values:
.sp
.ne 2
.na
\fB\fBLGRP_MEM_SZ_FREE\fR\fR
.ad
.RS 25n
free memory
.RE

.sp
.ne 2
.na
\fB\fBLGRP_MEM_SZ_INSTALLED\fR\fR
.ad
.RS 25n
installed memory
.RE

The $\fIcontent\fR argument should be set to one of the following values to
specify whether the direct contents or everything in this lgroup including its
children should be returned:
.sp
.ne 2
.na
\fB\fBLGRP_CONTENT_HIERARCHY\fR\fR
.ad
.RS 26n
Return everything within this hierarchy.
```

```
.RE

.sp
.ne 2
.na
\fB\fBLGRP_CONTENT_DIRECT\fR\fR
.ad
.RS 26n
Return that which is directly contained in this lgroup.
.RE

The total sizes include all the memory in the lgroup including its children,
while the others reflect only the memory contained directly in the given
lgroup.
.sp
Upon successful completion, the size in bytes is returned. Otherwise,
\fBundef\fR is returned and \fB$!\fR is set to indicate the error.
.sp
See \fBlgrp_mem_size\fR(3LGRP) for more information.
.RE

.sp
.ne 2
.na
\fB\fBlgrp_version\fR([$\fIversion\fR])\fR
.ad
.sp .6
.RS 4n
This function takes an interface version number, $\fIversion\fR, as an argument
and returns an lgroup interface version. The $\fIversion\fR argument should be
the value of \fBLGRP_VER_CURRENT\fR or \fBLGRP_VER_NONE\fR to find out the
current lgroup interface version on the running system.
.sp
If $\fIversion\fR is still supported by the implementation, then
\fBlgrp_version()\fR returns the requested version. If \fBLGRP_VER_NONE\fR is
returned, the implementation cannot support the requested version.
.sp
If $\fIversion\fR is \fBLGRP_VER_NONE\fR, \fBlgrp_version()\fR returns the
current version of the library.
.sp
The following example tests whether the version of the interface used by the
caller is supported:
.sp
.in +2
.nf
lgrp_version(LGRP_VER_CURRENT) == LGRP_VER_CURRENT or
    die("Built with unsupported lgroup interface");
.fi
.in -2

See \fBlgrp_version\fR(3LGRP) for more information.
.RE

.sp
.ne 2
.na
\fB\fBlgrp_affinity_set\fR($\fIidtype\fR, $\fIid\fR, $\fIlgrp\fR,
$\fIaffinity\fR)\fR
.ad
.sp .6
.RS 4n
This function sets the affinity that the LWP or set of LWPs specified by
$\fIidtype\fR and $\fIid\fR have for the given lgroup. The lgroup affinity can
be set to \fBLGRP_AFF_STRONG\fR, \fBLGRP_AFF_WEAK\fR, or \fBLGRP_AFF_NONE\fR.
.sp
If the $\fIidtype\fR is \fBP_PID\fR, the affinity is retrieved for one of the
```

```
LWPs in the process or set for all the LWPs of the process with process ID
(PID) $\fIid\fR. The affinity is retrieved or set for the LWP of the current
process with LWP ID $\fIid\fR if $\fIidtype\fR is \fBP_LWPID\fR. If $\fIid\fR
is \fBP_MYID\fR, then the current LWP or process is specified.
.sp
There are different levels of affinity that can be specified by a thread for a
particular lgroup. The levels of affinity are the following from strongest to
particuliar lgroup. The levels of affinity are the following from strongest to
weakest:
.sp
.ne 2
.na
\fB\fBLGRP_AFF_STRONG\fR\fR
.ad
.RS 19n
strong affinity
.RE

.sp
.ne 2
.na
\fB\fBLGRP_AFF_WEAK\fR\fR
.ad
.RS 19n
weak affinity
.RE

.sp
.ne 2
.na
\fB\fBLGRP_AFF_NONE\fR\fR
.ad
.RS 19n
no affinity
.RE

Upon successful completion, \fBlgrp_affinity_set()\fR returns 1. Otherwise, it
returns \fBundef\fR and set \fB$!\fR to indicate the error.
.sp
See \fBlgrp_affinity_set\fR(3LGRP) for more information.
.RE

.sp
.ne 2
.na
\fB\fBlgrp_affinity_get\fR($\fIidtype\fR, $\fIid\fR, $\fIlgrp\fR)\fR
.ad
.sp .6
.RS 4n
This function returns the affinity that the LWP has to a given lgroup.
.sp
See \fBlgrp_affinity_get\fR(3LGRP) for more information.
.RE

.sp
.ne 2
.na
\fB\fBlgrp_latency_cookie\fR($\fIcookie\fR, $\fIfrom\fR, $\fIto\fR,
[$\fIbetween\fR=\fBLGRP_LAT_CPU_TO_MEM\fR])\fR
.ad
.sp .6
.RS 4n
This function takes a cookie representing a snapshot of the lgroup hierarchy
and returns the latency value between a hardware resource in the $\fIfrom\fR
lgroup to a hardware resource in the $\fIto\fR lgroup. If $\fIfrom\fR is the
same lgroup as $\fIto\fR, the latency value within that lgroup is returned.
```

```
.sp
The optional $\fIbetween\fR argument should be set to \fBLGRP_LAT_CPU_TO_MEM\fR
to specify between which hardware resources the latency should be measured. The
only valid value is \fBLGRP_LAT_CPU_TO_MEM\fR, which represents latency from
CPU to memory.
.sp
Upon successful completion, \fBlgrp_latency_cookie()\fR return 1.  Otherwise,
it returns \fBundef\fR and set \fB$!\fR to indicate the error. For LGRP API
version 1, the \fBlgrp_latency_cookie()\fR is an alias for
\fBlgrp_latency.()\fR
.sp
See \fBlgrp_latency_cookie\fR(3LGRP) for more information.
.RE

.sp
.ne 2
.na
\fB\fBlgrp_latency\fR($\fIfrom\fR, $\fIto\fR)\fR
.ad
.sp .6
.RS 4n
This function is similar to the \fBlgrp_latency_cookie()\fR function, but
returns the latency between the given lgroups at the given instant in time.
Since lgroups can be freed and reallocated, this function might not be able to
provide a consistent answer across calls. For that reason,
\fBlgrp_latency_cookie()\fR should be used in its place.
.sp
See \fBlgrp_latency\fR(3LGRP) for more information.
.RE

.sp
.ne 2
.na
\fB\fBlgrp_resources\fR($\fIcookie\fR, $\fIlgrp\fR, $\fItype\fR)\fR
.ad
.sp .6
.RS 4n
This function returns the list of lgroups directly containing resources of the
specified type. The resources are represented by a set of lgroups in which each
lgroup directly contains CPU and/or memory resources.
.sp
The \fItype\fR can be specified as:
.sp
.ne 2
.na
\fB\fBLGRP_RSRC_CPU\fR\fR
.ad
.RS 17n
CPU resources
.RE

.sp
.ne 2
.na
\fB\fBLGRP_RSRC_MEM\fR\fR
.ad
.RS 17n
memory resources
.RE

In the event of error, \fBundef\fR or an empty list is returned and \fB$!\fR is
set to indicate the error.
.sp
This function is available only for API version 2 and returns \fBundef\fR or an
empty list for API version 1 and sets \fB$!\fR to \fBEINVAL\fR.
.sp
```

```
See \fBlgrp_resources\fR(3LGRP) for more information.
.RE

.sp
.ne 2
.na
\fB\fBlgrp_lgrps\fR($\fIcookie\fR, [$\fIlgrp\fR])\fR
.ad
.sp .6
.RS 4n
This function returns a list of all lgroups in a hierarchy starting from
$\fIlgrp\fR. If $\fIlgrp\fR is not specified, uses the value of
\fBlgrp_root\fR($\fIcookie\fR).  This function returns the empty list on
failure.
.sp
When called in scalar context, this function returns the total number of
lgroups in the system.
.RE

.sp
.ne 2
.na
\fB\fBlgrp_leaves\fR($\fIcookie\fR, [$\fIlgrp\fR])\fR
.ad
.sp .6
.RS 4n
This function returns a list of all leaf lgroups in a hierarchy starting from
$\fIlgrp\fR. If $\fIlgrp\fR is not specified, this function uses the value of
\fBlgrp_root\fR($\fIcookie\fR). It returns \fBundef\fR or an empty list on
failure.
.sp
When called in scalar context, this function returns the total number of leaf
lgroups in the system.
.RE

.sp
.ne 2
.na
\fB\fBlgrp_isleaf\fR($\fIcookie\fR, $\fIlgrp\fR)\fR
.ad
.sp .6
.RS 4n
This function returns True if $\fIlgrp\fR is a leaf (has no children).
Otherwise it returns False.
.RE

.SS "Object methods"
.ne 2
.na
\fB\fBnew\fR([$\fIview\fR])\fR
.ad
.sp .6
.RS 4n
This method creates a new \fBSun::Solaris::Lgrp\fR object. An optional argument
is passed to the \fBlgrp_init()\fR function. By default this method uses
\fBLGRP_VIEW_OS\fR.
.RE

.sp
.ne 2
.na
\fB\fBcookie()\fR\fR
.ad
.sp .6
.RS 4n
This method returns a transparent cookie that can be passed to functions
```

```
accepting the cookie.
.RE

.sp
.ne 2
.na
\fB\fBversion\fR([$\fIversion\fR])\fR
.ad
.sp .6
.RS 4n
Without the argument, this method returns the current version of the
\fBliblgrp\fR(3LIB) library. This method is a wrapper for \fBlgrp_version()\fR
with \fBLGRP_VER_NONE\fR as the default version argument.
.RE

.sp
.ne 2
.na
\fB\fBstale()\fR\fR
.ad
.sp .6
.RS 4n
This method returns T if the lgroup information in the object is stale and F
otherwise. It is a wrapper for \fBlgrp_cookie_stale()\fR.
.RE

.sp
.ne 2
.na
\fB\fBview()\fR\fR
.ad
.sp .6
.RS 4n
This method returns the snapshot’s view of the lgroup hierarchy. It is a
wrapper for \fBlgrp_view()\fR.
.RE

.sp
.ne 2
.na
\fB\fBroot()\fR\fR
.ad
.sp .6
.RS 4n
This method returns the root lgroup. It is a wrapper for \fBlgrp_root()\fR.
.RE

.sp
.ne 2
.na
\fB\fBchildren\fR($\fIlgrp\fR)\fR
.ad
.sp .6
.RS 4n
This method returns the list of lgroups that are children of the specified
lgroup. It is a wrapper for \fBlgrp_children()\fR.
.RE

.sp
.ne 2
.na
\fB\fBparents\fR($\fIlgrp\fR)\fR
.ad
.sp .6
.RS 4n
This method returns the list of lgroups that are parents of the specified
```

```
lgroup. It is a wrapper for \fBlgrp_parents()\fR.
.RE

.sp
.ne 2
.na
\fB\fBnlgrps()\fR\fR
.ad
.sp .6
.RS 4n
This method returns the number of lgroups in the hierarchy. It is a wrapper for
\fBlgrp_nlgrps()\fR.
.RE

.sp
.ne 2
.na
\fB\fBmem_size\fR($\fIlgrp\fR, $\fItype\fR, $\fIcontent\fR)\fR
.ad
.sp .6
.RS 4n
This method returns the memory size of the given lgroup in bytes. It is a
wrapper for \fBlgrp_mem_size()\fR.
.RE

.sp
.ne 2
.na
\fB\fBcpus\fR($\fIlgrp\fR, $\fIcontext\fR)\fR
.ad
.sp .6
.RS 4n
This method returns the list of CPUs in the lgroup specified by $lgrp. It is a
wrapper for \fBlgrp_cpus()\fR.
.RE

.sp
.ne 2
.na
\fB\fBresources\fR($\fIlgrp\fR, $\fItype\fR)\fR
.ad
.sp .6
.RS 4n
This method returns the list of lgroups directly containing resources of the
specified type. It is a wrapper for \fBlgrp_resources()\fR.
.RE

.sp
.ne 2
.na
\fB\fBhome\fR($\fIidtype\fR, $\fIid\fR)\fR
.ad
.sp .6
.RS 4n
This method returns the home lgroup for the given process or thread. It is a
wrapper for \fBlgrp_home()\fR.
.RE

.sp
.ne 2
.na
\fB\fBaffinity_get\fR($\fIidtype\fR, $\fIid\fR, $\fIlgrp\fR)\fR
.ad
.sp .6
.RS 4n
This method returns the affinity that the LWP has to a given lgrp. It is a
```

```
wrapper for \fBlgrp_affinity_get()\fR.
.RE

.sp
.ne 2
.na
\fB\fBaffinity_set\fR($\fIidtype\fR, $\fIid\fR, $\fIlgrp\fR,
$\fIaffinity\fR)\fR
.ad
.sp .6
.RS 4n
This method sets the affinity that the LWP or set of LWPs specified by
$\fIidtype\fR and $\fIid\fR have for the given lgroup. It is a wrapper for
lgrp_affinity_set.
.RE

.sp
.ne 2
.na
\fB\fBlgrps\fR([$\fIlgrp\fR])\fR
.ad
.sp .6
.RS 4n
This method returns list of all lgroups in a hierarchy starting from
$\fIlgrp\fR or the \fBlgrp_root()\fR if $\fIlgrp\fR is not specified. It is a
wrapper for \fBlgrp_lgrps()\fR.
.RE

.sp
.ne 2
.na
\fB\fBleaves\fR([$\fIlgrp\fR])\fR
.ad
.sp .6
.RS 4n
This method returns a list of all leaf lgroups in a hierarchy starting from
$\fIlgrp\fR.  If $\fIlgrp\fR is not specified, this method uses the value of
\fBlgrp_root()\fR. It is a wrapper for \fBlgrp_leaves()\fR.
.RE

.sp
.ne 2
.na
\fB\fBisleaf\fR($\fIlgrp\fR)\fR
.ad
.sp .6
.RS 4n
This method returns True if $\fIlgrp\fR is leaf (has no children) and False
otherwise. It is a wrapper for \fBlgrp_isleaf()\fR.
.RE

.sp
.ne 2
.na
\fB\fBlatency\fR($\fIfrom\fR, $\fIto\fR)\fR
.ad
.sp .6
.RS 4n
This method returns the latency value between a hardware resource in the
$\fIfrom\fR lgroup to a hardware resource in the $\fIto\fR lgroup. It uses
\fBlgrp_latency()\fR for version 1 of \fBliblgrp\fR and
\fBlgrp_latency_cookie()\fR for newer versions.
.RE

.SS "Exports"
.LP
```

```
By default nothing is exported from this module. The following tags can be used
to selectively import constants and functions defined in this module:
.sp
.ne 2
.na
\fB\fB:LGRP_CONSTANTS\fR\fR
.ad
.RS 19n
\fBLGRP_AFF_NONE\fR, \fBLGRP_AFF_STRONG\fR, \fBLGRP_AFF_WEAK\fR,
\fBLGRP_CONTENT_DIRECT\fR, \fBLGRP_CONTENT_HIERARCHY\fR,
\fBLGRP_MEM_SZ_FREE\fR, \fBLGRP_MEM_SZ_INSTALLED\fR, \fBLGRP_VER_CURRENT\fR,
\fBLGRP_VER_NONE\fR, \fBLGRP_VIEW_CALLER\fR, \fBLGRP_VIEW_OS\fR,
\fBLGRP_NONE\fR, \fBLGRP_RSRC_CPU\fR, \fBLGRP_RSRC_MEM\fR,
\fBLGRP_CONTENT_ALL\fR, \fBLGRP_LAT_CPU_TO_MEM\fR
.RE

.sp
.ne 2
.na
\fB\fB:PROC_CONSTANTS\fR\fR
.ad
.RS 19n
\fBP_PID\fR, \fBP_LWPID\fR, \fBP_MYID\fR
.RE

.sp
.ne 2
.na
\fB\fB:CONSTANTS\fR\fR
.ad
.RS 19n
\fB:LGRP_CONSTANTS\fR, \fB:PROC_CONSTANTS\fR
.RE

.sp
.ne 2
.na
\fB\fB:FUNCTIONS\fR\fR
.ad
.RS 19n
\fBlgrp_affinity_get()\fR, \fBlgrp_affinity_set()\fR, \fBlgrp_children()\fR,
\fBlgrp_cookie_stale()\fR, \fBlgrp_cpus()\fR, \fBlgrp_fini()\fR,
\fBlgrp_home()\fR, \fBlgrp_init()\fR, \fBlgrp_latency()\fR,
\fBlgrp_latency_cookie()\fR, \fBlgrp_mem_size()\fR, \fBlgrp_nlgrps()\fR,
\fBlgrp_parents()\fR, \fBlgrp_root()\fR, \fBlgrp_version()\fR,
\fBlgrp_view()\fR, \fBlgrp_resources()\fR, \fBlgrp_lgrps()\fR,
\fBlgrp_leaves()\fR, \fBlgrp_isleaf()\fR
.RE

.sp
.ne 2
.na
\fB\fB:ALL\fR\fR
.ad
.RS 19n
\fB:CONSTANTS\fR, \fB:FUNCTIONS\fR
.RE

.SS "Error values"
.LP
The functions in this module return \fBundef\fR or an empty list when an
underlying library function fails. The \fB$!\fR is set to provide more
information values for the error. The following error codes are possible:
.sp
.ne 2
.na
```

```
\fB\fBEINVAL\fR\fR
.ad
.RS 10n
The value supplied is not valid.
.RE

.sp
.ne 2
.na
\fB\fBENOMEM\fR\fR
.ad
.RS 10n
There was not enough system memory to complete an operation.
.RE

.sp
.ne 2
.na
\fB\fBEPERM\fR\fR
.ad
.RS 10n
The effective user of the calling process does not have appropriate privileges,
and its real or effective user ID does not match the real or effective user ID
of one of the threads.
.RE

.sp
.ne 2
.na
\fB\fBESRCH\fR\fR
.ad
.RS 10n
The specified process or thread was not found.
.RE

.SS "Difference in the API versions"
.LP
The \fBliblgrp\fR(3LIB) library is versioned. The exact version that was used
to compile a module is available through the \fBlgrp_version()\fR function.
.sp
.LP
Version 2 of the \fBlgrp_user\fR API introduced the following constants and
functions not present in version 1:
.br
.in +2
\fBLGRP_RSRC_CPU\fR constant
.in -2
.br
.in +2
\fBLGRP_RSRC_MEM\fR constant
.in -2
.br
.in +2
\fBLGRP_CONTENT_ALL\fR constant
.in -2
.br
.in +2
\fBLGRP_LAT_CPU_TO_MEM\fR constant
.in -2
.br
.in +2
\fBlgrp_resources()\fR function
.in -2
.br
.in +2
\fBlgrp_latency_cookie()\fR function
```

```
.in -2
.sp
.LP
The \fBLGRP_RSRC_CPU\fR and \fBLGRP_RSRC_MEM\fR constants are not defined for
version 1. The \fBlgrp_resources()\fR function is defined for version 1 but
always returns an empty list. The \fBlgrp_latency_cookie()\fR function is an
alias for \fBlgrp_latency()\fR for version 1.
.SH ATTRIBUTES
.LP
See \fBattributes\fR(5) for descriptions of the following attributes:
.sp

.sp
.TS
box;
c | c
l | l .
ATTRIBUTE TYPE	ATTRIBUTE VALUE
_
Interface Stability	Unstable
.TE

.SH SEE ALSO
.LP
\fBlgrp_affinity_get\fR(3LGRP), \fBlgrp_affinity_set\fR(3LGRP),
\fBlgrp_children\fR(3LGRP), \fBlgrp_cookie_stale\fR(3LGRP),
\fBlgrp_cpus\fR(3LGRP), \fBlgrp_fini\fR(3LGRP), \fBlgrp_home\fR(3LGRP),
\fBlgrp_init\fR(3LGRP), \fBlgrp_latency\fR(3LGRP),
\fBlgrp_latency_cookie\fR(3LGRP), \fBlgrp_mem_size\fR(3LGRP),
\fBlgrp_nlgrps\fR(3LGRP), \fBlgrp_parents\fR(3LGRP),
\fBlgrp_resources\fR(3LGRP), \fBlgrp_root\fR(3LGRP), \fBlgrp_version\fR(3LGRP),
\fBlgrp_view\fR(3LGRP), \fBliblgrp\fR(3LIB), \fBattributes\fR(5)
```

```
**********************************************************
   85737 Tue Dec 11 09:48:09 2018
new/usr/src/man/man3tecla/gl_get_line.3tecla
10057 Man page misspellings ouput particuliar overriden
Reviewed by: Gerg¯  Mih^¡ly Doma <domag02@gmail.com>
**********************************************************
______unchanged_portion_omitted_
.fi
.in -2

.sp
.LP
In the example, first the resources needed by the \fBgl_get_line()\fR function
are created by calling \fBnew_GetLine()\fR. This allocates the memory used in
subsequent calls to the \fBgl_get_line()\fR function, including the history
buffer for recording previously entered lines. Then one or more lines are read
from the user, until either an error occurs, or the user types exit. Then
finally the resources that were allocated by \fBnew_GetLine()\fR, are returned
to the system by calling \fBdel_GetLine()\fR. Note the use of the \fINULL\fR
return value of \fBdel_GetLine()\fR to make \fIgl\fR \fINULL\fR. This is a
safety precaution. If the program subsequently attempts to pass \fIgl\fR to
\fBgl_get_line()\fR, said function will complain, and return an error, instead
of attempting to use the deleted resource object.
.SS "The Functions Used In The Example"
.LP
The \fBnew_GetLine()\fR function creates the resources used by the
\fBgl_get_line()\fR function and returns an opaque pointer to the object that
contains them. The maximum length of an input line is specified by the
\fIlinelen\fR argument, and the number of bytes to allocate for storing history
lines is set by the \fIhistlen\fR argument. History lines are stored
back-to-back in a single buffer of this size. Note that this means that the
number of history lines that can be stored at any given time, depends on the
lengths of the individual lines. If you want to place an upper limit on the
number of lines that can be stored, see the description of the
\fBgl_limit_history()\fR function. If you do not want history at all, specify
\fIhistlen\fR as zero, and no history buffer will be allocated.
.sp
.LP
On error, a message is printed to \fBstderr\fR and \fINULL\fR is returned.
.sp
.LP
The \fBdel_GetLine()\fR function deletes the resources that were returned by a
previous call to \fBnew_GetLine()\fR. It always returns \fINULL\fR (for
example, a deleted object). It does nothing if the \fIgl\fR argument is
\fINULL\fR.
.sp
.LP
The \fBgl_get_line()\fR function can be called any number of times to read
input from the user. The gl argument must have been previously returned by a
call to \fBnew_GetLine()\fR. The \fIprompt\fR argument should be a normal
null-terminated string, specifying the prompt to present the user with. By
default prompts are displayed literally, but if enabled with the
\fBgl_prompt_style()\fR function, prompts can contain directives to do
underlining, switch to and from bold fonts, or turn highlighting on and off.
.sp
.LP
If you want to specify the initial contents of the line for the user to edit,
pass the desired string with the \fIstart_line\fR argument. You can then
specify which character of this line the cursor is initially positioned over by
using the \fIstart_pos\fR argument. This should be -1 if you want the cursor to
follow the last character of the start line. If you do not want to preload the
line in this manner, send \fIstart_line\fR as \fINULL\fR, and set
\fIstart_pos\fR to -1.
.sp
.LP
The \fBgl_get_line()\fR function returns a pointer to the line entered by the
```

```
user, or \fINULL\fR on error or at the end of the input. The returned pointer
is part of the specified \fIgl\fR resource object, and thus should not be freed
by the caller, or assumed to be unchanging from one call to the next. When
reading from a user at a terminal, there will always be a newline character at
the end of the returned line. When standard input is being taken from a pipe or
a file, there will similarly be a newline unless the input line was too long to
store in the internal buffer. In the latter case you should call
\fBgl_get_line()\fR again to read the rest of the line. Note that this behavior
makes \fBgl_get_line()\fR similar to \fBfgets\fR(3C). When \fBstdin\fR is not
connected to a terminal, \fBgl_get_line()\fR simply calls \fBfgets()\fR.
.SS "The Return Status Of \fBgl_get_line()\fR"
.LP
The \fBgl_get_line()\fR function has two possible return values: a pointer to
the completed input line, or \fINULL\fR. Additional information about what
caused \fBgl_get_line()\fR to return is available both by inspecting
\fBerrno\fR and by calling the \fBgl_return_status()\fR function.
.sp
.LP
The following are the possible enumerated values returned by
\fBgl_return_status()\fR:
.sp
.ne 2
.na
\fB\fBGLR_NEWLINE\fR\fR
.ad
.RS 15n
The last call to \fBgl_get_line()\fR successfully returned a completed input
line.
.RE

.sp
.ne 2
.na
\fB\fBGLR_BLOCKED\fR\fR
.ad
.RS 15n
The \fBgl_get_line()\fR function was in non-blocking server mode, and returned
early to avoid blocking the process while waiting for terminal I/O. The
\fBgl_pending_io()\fR function can be used to see what type of I/O
\fBgl_get_line()\fR was waiting for. See the \fBgl_io_mode\fR(3TECLA).
.RE

.sp
.ne 2
.na
\fB\fBGLR_SIGNAL\fR\fR
.ad
.RS 15n
A signal was caught by \fBgl_get_line()\fR that had an after-signal disposition
of \fBGLS_ABORT\fR. See \fBgl_trap_signal()\fR.
.RE

.sp
.ne 2
.na
\fB\fBGLR_TIMEOUT\fR\fR
.ad
.RS 15n
The inactivity timer expired while \fBgl_get_line()\fR was waiting for input,
and the timeout callback function returned \fBGLTO_ABORT\fR. See
\fBgl_inactivity_timeout()\fR for information about timeouts.
.RE

.sp
.ne 2
.na
```

```
\fB\fBGLR_FDABORT\fR\fR
.ad
.RS 15n
An application I/O callback returned \fBGLFD_ABORT\fR. Ssee
\fBgl_watch_fd()\fR.
.RE

.sp
.ne 2
.na
\fB\fBGLR_EOF\fR\fR
.ad
.RS 15n
End of file reached. This can happen when input is coming from a file or a
pipe, instead of the terminal. It also occurs if the user invokes the
list-or-eof or del-char-or-list-or-eof actions at the start of a new line.
.RE

.sp
.ne 2
.na
\fB\fBGLR_ERROR\fR\fR
.ad
.RS 15n
An unexpected error caused \fBgl_get_line()\fR to abort (consult \fBerrno\fR
and/or \fBgl_error_message()\fR for details.
.RE

.sp
.LP
When \fBgl_return_status()\fR returns \fBGLR_ERROR\fR and the value of
\fBerrno\fR is not sufficient to explain what happened, you can use the
\fBgl_error_message()\fR function to request a description of the last error
that occurred.
.sp
.LP
The return value of \fBgl_error_message()\fR is a pointer to the message that
occurred. If the \fIbuff\fR argument is \fINULL\fR, this will be a pointer to a
buffer within \fIgl\fR whose value will probably change on the next call to any
function associated with \fBgl_get_line()\fR. Otherwise, if a non-null
\fIbuff\fR argument is provided, the error message, including a '\e0'
terminator, will be written within the first \fIn\fR elements of this buffer,
and the return value will be a pointer to the first element of this buffer. If
the message will not fit in the provided buffer, it will be truncated to fit.
.SS "Optional Prompt Formatting"
.LP
Whereas by default the prompt string that you specify is displayed literally
without any special interpretation of the characters within it, the
\fBgl_prompt_style()\fR function can be used to enable optional formatting
directives within the prompt.
.sp
.LP
The \fIstyle\fR argument, which specifies the formatting style, can take any of
the following values:
.sp
.ne 2
.na
\fB\fBGL_FORMAT_PROMPT\fR\fR
.ad
.RS 21n
In this style, the formatting directives described below, when included in
prompt strings, are interpreted as follows:
.sp
.ne 2
.na
\fB\fB%B\fR\fR
```

```
.ad
.RS 6n
Display subsequent characters with a bold font.
.RE

.sp
.ne 2
.na
\fB\fB%b\fR\fR
.ad
.RS 6n
Stop displaying characters with the bold font.
.RE

.sp
.ne 2
.na
\fB\fB%F\fR\fR
.ad
.RS 6n
Make subsequent characters flash.
.RE

.sp
.ne 2
.na
\fB\fB%f\fR\fR
.ad
.RS 6n
Turn off flashing characters.
.RE

.sp
.ne 2
.na
\fB\fB%U\fR\fR
.ad
.RS 6n
Underline subsequent characters.
.RE

.sp
.ne 2
.na
\fB\fB%u\fR\fR
.ad
.RS 6n
Stop underlining characters.
.RE

.sp
.ne 2
.na
\fB\fB%P\fR\fR
.ad
.RS 6n
Switch to a pale (half brightness) font.
.RE

.sp
.ne 2
.na
\fB\fB%p\fR\fR
.ad
.RS 6n
Stop using the pale font.
```

```
.RE

.sp
.ne 2
.na
\fB\fB%S\fR\fR
.ad
.RS 6n
Highlight subsequent characters (also known as standout mode).
.RE

.sp
.ne 2
.na
\fB\fB%s\fR\fR
.ad
.RS 6n
Stop highlighting characters.
.RE

.sp
.ne 2
.na
\fB\fB%V\fR\fR
.ad
.RS 6n
Turn on reverse video.
.RE

.sp
.ne 2
.na
\fB\fB%v\fR\fR
.ad
.RS 6n
Turn off reverse video.
.RE

.sp
.ne 2
.na
\fB\fB%%\fR\fR
.ad
.RS 6n
Display a single % character.
.RE

For example, in this mode, a prompt string like "%UOK%u$" would display the
prompt "OK$", but with the OK part underlined.
.sp
Note that although a pair of characters that starts with a % character, but
does not match any of the above directives is displayed literally, if a new
directive is subsequently introduced which does match, the displayed prompt
will change, so it is better to always use %% to display a literal %.
.sp
Also note that not all terminals support all of these text attributes, and that
some substitute a different attribute for missing ones.
.RE

.sp
.ne 2
.na
\fB\fBGL_LITERAL_PROMPT\fR\fR
.ad
.RS 21n
In this style, the prompt string is printed literally. This is the default
```

```
style.
.RE

.SS "Alternate Configuration Sources"
.LP
By default users have the option of configuring the behavior of
\fBgl_get_line()\fR with a configuration file called \fB\&.teclarc\fR in their
home directories. The fact that all applications share this same configuration
file is both an advantage and a disadvantage. In most cases it is an advantage,
since it encourages uniformity, and frees the user from having to configure
each application separately. In some applications, however, this single means
of configuration is a problem. This is particularly true of embedded software,
where there’s no filesystem to read a configuration file from, and also in
applications where a radically different choice of keybindings is needed to
emulate a legacy keyboard interface. To cater for such cases, the
\fBgl_configure_getline()\fR function allows the application to control where
configuration information is read from.
.sp
.LP
The \fBgl_configure_getline()\fR function allows the configuration commands
that would normally be read from a user’s \fB~/.teclarc\fR file, to be read
from any or none of, a string, an application specific configuration file,
and/or a user-specific configuration file. If this function is called before
the first call to \fBgl_get_line()\fR, the default behavior of reading
\fB~/.teclarc\fR on the first call to \fBgl_get_line()\fR is disabled, so all
configurations must be achieved using the configuration sources specified with
this function.
.sp
.LP
If \fIapp_string\fR != \fINULL\fR, then it is interpreted as a string
containing one or more configuration commands, separated from each other in the
string by embedded newline  characters. If \fIapp_file\fR != \fINULL\fR then it
is interpreted as the full pathname of an application-specific configuration
file. If user_file != \fINULL\fR then it is interpreted as the full path name
of a user-specific configuration file, such as \fB~/.teclarc\fR. For example,
in the call
.sp
.in +2
.nf
gl_configure_getline(gl, "edit-mode vi \en nobeep",
                     "/usr/share/myapp/teclarc", "~/.teclarc");
.fi
.in -2

.sp
.LP
The \fIapp_string\fR argument causes the calling application to start in
\fBvi\fR(1) edit-mode, instead of the default \fBemacs\fR mode, and turns off
the use of the terminal bell by the library. It then attempts to read
system-wide configuration commands from an optional file called
\fB/usr/share/myapp/teclarc\fR, then finally reads user-specific configuration
commands from an optional \fB\&.teclarc\fR file in the user’s home directory.
Note that the arguments are listed in ascending order of priority, with the
contents of \fIapp_string\fR being potentially over riden by commands in
\fIapp_file\fR, and commands in \fIapp_file\fR potentially being overridden by
\fIapp_file\fR, and commands in \fIapp_file\fR potentially being overriden by
commands in \fIuser_file\fR.
.sp
.LP
You can call this function as many times as needed, the results being
cumulative, but note that copies of any file names specified with the
\fIapp_file\fR and \fIuser_file\fR arguments are recorded internally for
subsequent use by the read-init-files key-binding function, so if you plan to
call this function multiple times, be sure that the last call specifies the
filenames that you want re-read when the user requests that the configuration
files be re-read.
```

```
.sp
.LP
Individual key sequences can also be bound and unbound using the
\fBgl_bind_keyseq()\fR function. The \fIorigin\fR argument specifies the
priority of the binding, according to whom it is being established for, and
must be one of the following two values.
.sp
.ne 2
.na
\fB\fBGL_USER_KEY\fR\fR
.ad
.RS 15n
The user requested this key-binding.
.RE

.sp
.ne 2
.na
\fB\fBGL_APP_KEY\fR\fR
.ad
.RS 15n
This is a default binding set by the application.
.RE

.sp
.LP
When both user and application bindings for a given key sequence have been
specified, the user binding takes precedence. The application's binding is
subsequently reinstated if the user's binding is later unbound with either
another call to this function, or a call to \fBgl_configure_getline()\fR.
.sp
.LP
The \fIkeyseq\fR argument specifies the key sequence to be bound or unbound,
and is expressed in the same way as in a \fB~/.teclarc\fR configuration file.
The \fIaction\fR argument must either be a string containing the name of the
action to bind the key sequence to, or it must be \fINULL\fR or \fB""\fR to
unbind the key sequence.
.SS "Customized Word Completion"
.LP
If in your application you would like to have TAB completion complete other
things in addition to or instead of filenames, you can arrange this by
registering an alternate completion callback function with a call to the
\fBgl_customize_completion()\fR function.
.sp
.LP
The \fIdata\fR argument provides a way for your application to pass arbitrary,
application-specific information to the callback function. This is passed to
the callback every time that it is called. It might for example point to the
symbol table from which possible completions are to be sought. The
\fImatch_fn\fR argument specifies the callback function to be called. The
\fICplMatchFn\fR function type is defined in <\fBlibtecla.h\fR>, as is a
\fBCPL_MATCH_FN()\fR macro that you can use to declare and prototype callback
functions. The declaration and responsibilities of callback functions are
described in depth on the \fBcpl_complete_word\fR(3TECLA) manual page.
.sp
.LP
The callback function is responsible for looking backwards in the input line
from the point at which the user pressed TAB, to find the start of the word
being completed. It then must lookup possible completions of this word, and
record them one by one in the \fBWordCompletion\fR object that is passed to it
as an argument, by calling the \fBcpl_add_completion()\fR function. If the
callback function wants to provide filename completion in addition to its own
specific completions, it has the option of itself calling the builtin filename
completion callback. This also is documented on the
\fBcpl_complete_word\fR(3TECLA) manual page.
.sp
```

```
.LP
If you would like \fBgl_get_line()\fR to return the current input line when a
successful completion is been made, you can arrange this when you call
\fBcpl_add_completion()\fR by making the last character of the continuation
suffix a newline character. The input line will be updated to display the
completion, together with any contiuation suffix up to the newline character,
and \fBgl_get_line()\fR will return this input line.
.sp
.LP
If your callback function needs to write something to the terminal, it must
call \fBgl_normal_io()\fR before doing so. This will start a new line after the
input line that is currently being edited, reinstate normal terminal I/O, and
notify \fBgl_get_line()\fR that the input line will need to be redrawn when the
callback returns.
.SS "Adding Completion Actions"
.LP
In the previous section the ability to customize the behavior of the only
default completion action, complete-word, was described. In this section the
ability to install additional action functions, so that different types of word
completion can be bound to different key sequences, is described. This is
achieved by using the \fBgl_completion_action()\fR function.
.sp
.LP
The \fIdata\fR and \fImatch_fn\fR arguments are as described on the
\fBcpl_complete_word\fR(3TECLA) manual page, and specify the callback function
that should be invoked to identify possible completions. The \fIlist_only\fR
argument determines whether the action that is being defined should attempt to
complete the word as far as possible in the input line before displaying any
possible ambiguous completions, or whether it should simply display the list of
possible completions without touching the input line. The former option is
selected by specifying a value of 0, and the latter by specifying a value of 1.
The \fIname\fR argument specifies the name by which configuration files and
future invocations of this function should refer to the action. This must
either be the name of an existing completion action to be changed, or be a new
unused name for a new action. Finally, the \fIkeyseq\fR argument specifies the
default key sequence to bind the action to. If this is \fINULL\fR, no new key
sequence will be bound to the action.
.sp
.LP
Beware that in order for the user to be able to change the key sequence that is
bound to actions that are installed in this manner, you shouldcall
\fBgl_completion_action()\fR to install a given action for the first time
between calling \fBnew_GetLine()\fR and the first call to \fBgl_get_line()\fR.
Otherwise, when the user's configuration file is read on the first call to
\fBgl_get_line()\fR, the name of the your additional action will not be known,
and any reference to it in the configuration file will generate an error.
.sp
.LP
As discussed for \fBgl_customize_completion()\fR, if your callback function
needs to write anything to the terminal, it must call \fBgl_normal_io()\fR
before doing so.
.SS "Defining Custom Actions"
.LP
Although the built-in key-binding actions are sufficient for the needs of most
applications, occasionally a specialized application may need to define one or
more custom actions, bound to application-specific key sequences. For example,
a sales application would benefit from having a key sequence that displayed the
part name that corresponded to a part number preceding the cursor. Such a
feature is clearly beyond the scope of the built-in action functions. So for
such special cases, the \fBgl_register_action()\fR function is provided.
.sp
.LP
The \fBgl_register_action()\fR function lets the application register an
external function, \fIfn\fR, that will thereafter be called whenever either the
specified key sequence, \fIkeyseq\fR, is entered by the user, or the user
enters any other key sequence that the user subsequently binds to the specified
```

```
action name, \fIname\fR, in their configuration file. The \fIdata\fR argument
can be a pointer to anything that the application wants to have passed to the
action function, \fIfn\fR, whenever that function is invoked.
.sp
.LP
The action function, \fIfn\fR, should be declared using the
\fBGL_ACTION_FN()\fR macro, which is defined in <\fBlibtecla.h\fR>.
.sp
.in +2
.nf
#define GL_ACTION_FN(fn) GlAfterAction (fn)(GetLine *gl, \e
                       void *data, int count, size_t curpos, \e
                       const char *line)
.fi
.in -2

.sp
.LP
The \fIgl\fR and \fIdata\fR arguments are those that were previously passed to
\fBgl_register_action()\fR when the action function was registered. The
\fIcount\fR argument is a numeric argument which the user has the option of
entering using the digit-argument action, before invoking the action. If the
user does not enter a number, then the \fIcount\fR argument is set to 1.
Nominally this argument is interpreted as a repeat count, meaning that the
action should be repeated that many times. In practice however, for some
actions a repeat count makes little sense. In such cases, actions can either
simply ignore the \fIcount\fR argument, or use its value for a different
purpose.
.sp
.LP
A copy of the current input line is passed in the read-only \fIline\fR
argument. The current cursor position within this string is given by the index
contained in the \fIcurpos\fR argument. Note that direct manipulation of the
input line and the cursor position is not permitted because the rules dictated
by various modes (such as \fBvi\fR mode versus \fBemacs\fR mode, no-echo mode,
and insert mode versus overstrike mode) make it too complex for an application
writer to write a conforming editing action, as well as constrain future
changes to the internals of \fBgl_get_line()\fR. A potential solution to this
dilemma would be to allow the action function to edit the line using the
existing editing actions. This is currently under consideration.
.sp
.LP
If the action function wishes to write text to the terminal without this
getting mixed up with the displayed text of the input line, or read from the
terminal without having to handle raw terminal I/O, then before doing either of
these operations, it must temporarily suspend line editing by calling the
\fBgl_normal_io()\fR function. This function flushes any pending output to the
terminal, moves the cursor to the start of the line that follows the last
terminal line of the input line, then restores the terminal to a state that is
suitable for use with the C \fBstdio\fR facilities. The latter includes such
things as restoring the normal mapping of \en to \er\en, and, when in server
mode, restoring the normal blocking form of terminal I/O. Having called this
function, the action function can read from and write to the terminal without
the fear of creating a mess. It is not necessary for the action function to
restore the original editing environment before it returns. This is done
automatically by \fBgl_get_line()\fR after the action function returns. The
following is a simple example of an action function which writes the sentence
"Hello world" on a new terminal line after the line being edited. When this
function returns, the input line is redrawn on the line that follows the "Hello
world" line, and line editing resumes.
.sp
.in +2
.nf
static GL_ACTION_FN(say_hello_fn)
{
    if(gl_normal_io(gl))   /* Temporarily suspend editing */
```

```
        return GLA_ABORT;
    printf("Hello world\en");
    return GLA_CONTINUE;
}
.fi
.in -2

.sp
.LP
Action functions must return one of the following values, to tell
\fBgl_get_line()\fR how to proceed.
.sp
.ne 2
.na
\fB\fBGLA_ABORT\fR\fR
.ad
.RS 16n
Cause \fBgl_get_line()\fR to return \fINULL\fR.
.RE

.sp
.ne 2
.na
\fB\fBGLA_RETURN\fR\fR
.ad
.RS 16n
Cause \fBgl_get_line()\fR to return the completed input line
.RE

.sp
.ne 2
.na
\fB\fBGLA_CONTINUE\fR\fR
.ad
.RS 16n
Resume command-line editing.
.RE

.sp
.LP
Note that the \fIname\fR argument of \fBgl_register_action()\fR specifies the
name by which a user can refer to the action in their configuration file. This
allows them to re-bind the action to an alternate key-sequence. In order for
this to work, it is necessary to call \fBgl_register_action()\fR between
calling \fBnew_GetLine()\fR and the first call to \fBgl_get_line()\fR.
.SS "History Files"
.LP
To save the contents of the history buffer before quitting your application and
subsequently restore them when you next start the application, the
\fBgl_save_history()\fR and \fBgl_load_history()\fR functions are provided.
.sp
.LP
The \fIfilename\fR argument specifies the name to give the history file when
saving, or the name of an existing history file, when loading. This may contain
home directory and environment variable expressions, such as
\fB~/.myapp_history\fR or \fB$HOME/.myapp_history\fR.
.sp
.LP
Along with each history line, additional information about it, such as its
nesting level and when it was entered by the user, is recorded as a comment
preceding the line in the history file. Writing this as a comment allows the
history file to double as a command file, just in case you wish to replay a
whole session using it. Since comment prefixes differ in different languages,
the comment argument is provided for specifying the comment prefix. For
example, if your application were a UNIX shell, such as the Bourne shell, you
would specify "#" here. Whatever you choose for the comment character, you must
```

```
specify the same prefix to \fBgl_load_history()\fR that you used when you
called \fBgl_save_history()\fR to write the history file.
.sp
.LP
The \fImax_lines\fR argument must be either -1 to specify that all lines in the
history list be saved, or a positive number specifying a ceiling on how many of
the most recent lines should be saved.
.sp
.LP
Both fuctions return non-zero on error, after writing an error message to
\fBstderr\fR. Note that \fBgl_load_history()\fR does not consider the
non-existence of a file to be an error.
.SS "Multiple History Lists"
.LP
If your application uses a single \fBGetLine\fR object for entering many
different types of input lines, you might want \fBgl_get_line()\fR to
distinguish the different types of lines in the history list, and only recall
lines that match the current type of line. To support this requirement,
\fBgl_get_line()\fR marks lines being recorded in the history list with an
integer identifier chosen by the application. Initially this identifier is set
to 0 by \fBnew_GetLine()\fR, but it can be changed subsequently by calling
\fBgl_group_history()\fR.
.sp
.LP
The integer identifier ID can be any number chosen by the application, but note
that \fBgl_save_history()\fR and \fBgl_load_history()\fR preserve the
association between identifiers and historical input lines between program
invocations, so you should choose fixed identifiers for the different types of
input line used by your application.
.sp
.LP
Whenever \fBgl_get_line()\fR appends a new input line to the history list, the
current history identifier is recorded with it, and when it is asked to recall
a historical input line, it only recalls lines that are marked with the current
identifier.
.SS "Displaying History"
.LP
The history list can be displayed by calling \fBgl_show_history()\fR. This
function displays the current contents of the history list to the \fBstdio\fR
output stream \fIfp\fR. If the \fImax_lines\fR argument is greater than or
equal to zero, then no more than this number of  the most recent lines will be
displayed. If the \fIall_groups\fR argument is non-zero, lines from all history
groups are displayed. Otherwise only those of the currently selected history
group are displayed. The format string argument, \fIfmt\fR, determines how the
line is displayed. This can contain arbitrary characters which are written
verbatim, interleaved with any of the following format directives:
.sp
.ne 2
.na
\fB\fB%D\fR\fR
.ad
.RS 6n
The date on which the line was originally entered, formatted like 2001-11-20.
.RE

.sp
.ne 2
.na
\fB\fB%T\fR\fR
.ad
.RS 6n
The time of day when the line was entered, formatted like 23:59:59.
.RE

.sp
.ne 2
```

```
.na
\fB\fB%N\fR\fR
.ad
.RS 6n
The sequential entry number of the line in the history buffer.
.RE

.sp
.ne 2
.na
\fB\fB%G\fR\fR
.ad
.RS 6n
The number of the history group which the line belongs to.
.RE

.sp
.ne 2
.na
\fB\fB%%\fR\fR
.ad
.RS 6n
A literal % character.
.RE

.sp
.ne 2
.na
\fB\fB%H\fR\fR
.ad
.RS 6n
The history line itself.
.RE

.sp
.LP
Thus a format string like "%D %T %H0" would output something like:
.sp
.in +2
.nf
2001-11-20 10:23:34  Hello world
.fi
.in -2

.sp
.LP
Note the inclusion of an explicit newline character in the format string.
.SS "Looking Up History"
.LP
The \fBgl_lookup_history()\fR function allows the calling application to look
up lines in the history list.
.sp
.LP
The \fIid\fR argument indicates which line to look up, where the first line
that was entered in the history list after \fBnew_GetLine()\fR was called is
denoted by 0, and subsequently entered lines are denoted with successively
higher numbers. Note that the range of lines currently preserved in the history
list can be queried by calling the \fBgl_range_of_history()\fR function. If the
requested line is in the history list, the details of the line are recorded in
the variable pointed to by the \fIhline\fR argument, and 1 is returned.
Otherwise 0 is returned, and the variable pointed to by \fIhline\fR is left
unchanged.
.sp
.LP
Beware that the string returned in \fIhline\fR->\fIline\fR is part of the
history buffer, so it must not be modified by the caller, and will be recycled
```

```
on the next call to any function that takes \fIgl\fR as its argument. Therefore
you should make a private copy of this string if you need to keep it.
.SS "Manual History Archival"
.LP
By default, whenever a line is entered by the user, it is automatically
appended to the history list, just before \fBgl_get_line()\fR returns the line
to the caller. This is convenient for the majority of applications, but there
are also applications that need finer-grained control over what gets added to
the history list. In such cases, the automatic addition of entered lines to the
history list can be turned off by calling the \fBgl_automatic_history()\fR
function.
.sp
.LP
If this function is called with its \fIenable\fR argument set to 0,
\fBgl_get_line()\fR will not automatically archive subsequently entered lines.
Automatic archiving can be reenabled at a later time by calling this function
again, with its \fIenable\fR argument set to 1. While automatic history
archiving is disabled, the calling application can use the
\fBgl_append_history()\fR to append lines to the history list as needed.
.sp
.LP
The \fIline\fR argument specifies the line to be added to the history list.
This must be a normal '\e0 ' terminated string. If this string contains any
newline characters, the line that gets archived in the history list will be
terminated by the first of these. Otherwise it will be terminated by the '\e0 '
terminator. If the line is longer than the maximum input line length that was
specified when \fBnew_GetLine()\fR was called, it will be truncated to the
actual \fBgl_get_line()\fR line length when the line is recalled.
.sp
.LP
If successful, \fBgl_append_history()\fR returns 0. Otherwise it returns
non-zero and sets \fBerrno\fR to one of the following values.
.sp
.ne 2
.na
\fB\fBEINVAL\fR\fR
.ad
.RS 10n
One of the arguments passed to \fBgl_append_history()\fR was \fINULL\fR.
.RE

.sp
.ne 2
.na
\fB\fBENOMEM\fR\fR
.ad
.RS 10n
The specified line was longer than the allocated size of the history buffer (as
specified when \fBnew_GetLine()\fR was called), so it could not be archived.
.RE

.sp
.LP
A textual description of the error can optionally be obtained by calling
\fBgl_error_message()\fR. Note that after such an error, the history list
remains in a valid state to receive new history lines, so there is little harm
in simply ignoring the return status of \fBgl_append_history()\fR.
.SS "Miscellaneous History Configuration"
.LP
If you wish to change the size of the history buffer that was originally
specified in the call to \fBnew_GetLine()\fR, you can do so with the
\fBgl_resize_history()\fR function.
.sp
.LP
The \fIhistlen\fR argument specifies the new size in bytes, and if you specify
this as 0, the buffer will be deleted.
```

```
.sp
.LP
As mentioned in the discussion of \fBnew_GetLine()\fR, the number of lines that
can be stored in the history buffer, depends on the lengths of the individual
lines. For example, a 1000 byte buffer could equally store 10 lines of average
length 100 bytes, or 20 lines of average length 50 bytes. Although the buffer
is never expanded when new lines are added, a list of pointers into the buffer
does get expanded when needed to accommodate the number of lines currently
stored in the buffer. To place an upper limit on the number of lines in the
buffer, and thus a ceiling on the amount of memory used in this list, you can
call the \fBgl_limit_history()\fR function.
.sp
.LP
The \fImax_lines\fR should either be a positive number >= 0, specifying an
upper limit on the number of lines in the buffer, or be -1 to cancel any
previously specified limit. When a limit is in effect, only the \fImax_lines\fR
most recently appended lines are kept in the buffer. Older lines are discarded.
.sp
.LP
To discard lines from the history buffer, use the \fBgl_clear_history()\fR
function.
.sp
.LP
The \fIall_groups\fR argument tells the function whether to delete just the
lines associated with the current history group (see \fBgl_group_history()\fR)
or all historical lines in the buffer.
.sp
.LP
The \fBgl_toggle_history()\fR function allows you to toggle history on and off
without losing the current contents of the history list.
.sp
.LP
Setting the \fIenable\fR argument to 0 turns off the history mechanism, and
setting it to 1 turns it back on. When history is turned off, no new lines will
be added to the history list, and history lookup key-bindings will act as
though there is nothing in the history buffer.
.SS "Querying History Information"
.LP
The configured state of the history list can be queried with the
\fBgl_history_state()\fR function. On return, the status information is
recorded in the variable pointed to by the \fIstate\fR argument.
.sp
.LP
The \fBgl_range_of_history()\fR function returns the number and range of lines
in the history list. The return values are recorded in the variable pointed to
by the range argument. If the \fInlines\fR member of this structure is greater
than zero, then the oldest and newest members report the range of lines in the
list, and \fInewest\fR=\fIoldest\fR+\fInlines\fR-1. Otherwise they are both
zero.
.sp
.LP
The \fBgl_size_of_history()\fR function returns the total size of the history
buffer and the amount of the buffer that is currently occupied.
.sp
.LP
On return, the size information is recorded in the variable pointed to by the
\fIsize\fR argument.
.SS "Changing Terminals"
.LP
The \fBnew_GetLine()\fR constructor function assumes that input is to be read
from \fBstdin\fR and output written to \fBstdout\fR. The following function
allows you to switch to different input and output streams.
.sp
.LP
The \fIgl\fR argument is the object that was returned by \fBnew_GetLine()\fR.
The \fIinput_fp\fR argument specifies the stream to read from, and
```

```
\fIoutput_fp\fR specifies the stream to be written to. Only if both of these
refer to a terminal, will interactive terminal input be enabled. Otherwise
\fBgl_get_line()\fR will simply call \fBfgets()\fR to read command input. If
both streams refer to a terminal, then they must refer to the same terminal,
and the type of this terminal must be specified with the \fIterm\fR argument.
The value of the \fIterm\fR argument is looked up in the terminal information
database (\fBterminfo\fR or \fBtermcap\fR), in order to determine which special
control sequences are needed to control various aspects of the terminal.
\fBnew_GetLine()\fR for example, passes the return value of
\fBgetenv\fR("TERM") in this argument. Note that if one or both of
\fIinput_fp\fR and \fIoutput_fp\fR do not refer to a terminal, then it is legal
to pass \fINULL\fR instead of a terminal type.
.sp
.LP
Note that if you want to pass file descriptors to \fBgl_change_terminal()\fR,
you can do this by creating \fBstdio\fR stream wrappers using the POSIX
\fBfdopen\fR(3C) function.
.SS "External Event Handling"
.LP
By default, \fBgl_get_line()\fR does not return until either a complete input
line has been entered by the user, or an error occurs. In programs that need to
watch for I/O from other sources than the terminal, there are two options.
.RS +4
.TP
.ie t \(bu
.el o
Use the functions described in the \fBgl_io_mode\fR(3TECLA) manual page to
switch \fBgl_get_line()\fR into non-blocking server mode. In this mode,
\fBgl_get_line()\fR becomes a non-blocking, incremental line-editing function
that can safely be called from an external event loop. Although this is a very
versatile method, it involves taking on some responsibilities that are normally
performed behind the scenes by \fBgl_get_line()\fR.
.RE
.RS +4
.TP
.ie t \(bu
.el o
While \fBgl_get_line()\fR is waiting for keyboard input from the user, you can
ask it to also watch for activity on arbitrary file descriptors, such as
network sockets or pipes, and have it call functions of your choosing when
activity is seen. This works on any system that has the select system call,
which is most, if not all flavors of UNIX.
.RE
.sp
.LP
Registering a file descriptor to be watched by \fBgl_get_line()\fR involves
calling the \fBgl_watch_fd()\fR function. If this returns non-zero, then it
means that either your arguments are invalid, or that this facility is not
supported on the host system.
.sp
.LP
The \fIfd\fR argument is the file descriptor to be watched. The event argument
specifies what type of activity is of interest, chosen from the following
enumerated values:
.sp
.ne 2
.na
\fB\fBGLFD_READ\fR\fR
.ad
.RS 15n
Watch for the arrival of data to be read.
.RE

.sp
.ne 2
.na
```

```
\fB\fBGLFD_WRITE\fR\fR
.ad
.RS 15n
Watch for the ability to write to the file descriptor without blocking.
.RE

.sp
.ne 2
.na
\fB\fBGLFD_URGENT\fR\fR
.ad
.RS 15n
Watch for the arrival of urgent out-of-band data on the file descriptor.
.RE

.sp
.LP
The \fIcallback\fR argument is the function to call when the selected activity
is seen. It should be defined with the following macro, which is defined in
libtecla.h.
.sp
.in +2
.nf
#define GL_FD_EVENT_FN(fn) GlFdStatus (fn)(GetLine *gl, \
                                    void *data, int fd, GlFdEvent event)
.fi
.in -2

.sp
.LP
The data argument of the \fBgl_watch_fd()\fR function is passed to the callback
function for its own use, and can point to anything you like, including
\fINULL\fR. The file descriptor and the event argument are also passed to the
callback function, and this potentially allows the same callback function to be
registered to more than one type of event and/or more than one file descriptor.
The return value of the callback function should be one of the following
values.
.sp
.ne 2
.na
\fB\fBGLFD_ABORT\fR\fR
.ad
.RS 17n
Tell \fBgl_get_line()\fR to abort. When this happens, \fBgl_get_line()\fR
returns \fINULL\fR, and a following call to \fBgl_return_status()\fR will
return \fBGLR_FDABORT\fR. Note that if the application needs \fBerrno\fR always
to have a meaningful value when \fBgl_get_line()\fR returns \fINULL\fR, the
callback function should set \fBerrno\fR appropriately.
.RE

.sp
.ne 2
.na
\fB\fBGLFD_REFRESH\fR\fR
.ad
.RS 17n
Redraw the input line  then continue waiting for input. Return this if your
callback wrote to the terminal.
.RE

.sp
.ne 2
.na
\fB\fBGLFD_CONTINUE\fR\fR
.ad
.RS 17n
```

```
Continue to wait for input, without redrawing the line.
.RE

.sp
.LP
Note that before calling the callback, \fBgl_get_line()\fR blocks most signals
and leaves its own signal handlers installed, so if you need to catch a
particular signal you will need to both temporarily install your own signal
handler, and unblock the signal. Be sure to re-block the signal (if it was
originally blocked) and reinstate the original signal handler, if any, before
returning.
.sp
.LP
Your callback should not try to read from the terminal, which is left in raw
mode as far as input is concerned. You can write to the terminal as usual,
since features like conversion of newline to carriage-return/linefeed are
re-enabled while the callback is running. If your callback function does write
to the terminal, be sure to output a newline first, and when your callback
returns, tell \fBgl_get_line()\fR that the input line needs to be redrawn, by
returning the \fBGLFD_REFRESH\fR status code.
.sp
.LP
To remove a callback function that you previously registered for a given file
descriptor and event, simply call \fBgl_watch_fd()\fR with the same \fIfd\fR
and \fIevent\fR arguments, but with a \fIcallback\fR argument of 0. The
\fIdata\fR argument is ignored in this case.
.SS "Setting An Inactivity Timeout"
.LP
The \fBgl_inactivity_timeout()\fR function can be used to set or cancel an
inactivity timeout. Inactivity in this case refers both to keyboard input, and
to I/O on any file descriptors registered by prior and subsequent calls to
\fBgl_watch_fd()\fR.
.sp
.LP
The timeout is specified in the form of an integral number of seconds and an
integral number of nanoseconds, specified by the \fIsec\fR and \fInsec\fR
arguments, respectively. Subsequently, whenever no activity is seen for this
time period, the function specified by the \fIcallback\fR argument is called.
The \fIdata\fR argument of \fBgl_inactivity_timeout()\fR is passed to this
callback function whenever it is invoked, and can thus be used to pass
arbitrary application-specific information to the callback. The following macro
is provided in <\fBlibtecla.h\fR> for applications to use to declare and
prototype timeout callback functions.
.sp
.in +2
.nf
#define GL_TIMEOUT_FN(fn) GlAfterTimeout (fn)(GetLine *gl, void *data)
.fi
.in -2

.sp
.LP
On returning, the application's callback is expected to return one of the
following enumerators to tell \fBgl_get_line()\fR how to proceed after the
timeout has been handled by the callback.
.sp
.ne 2
.na
\fB\fBGLTO_ABORT\fR\fR
.ad
.RS 17n
Tell \fBgl_get_line()\fR to abort. When this happens, \fBgl_get_line()\fR will
return \fINULL\fR, and a following call to \fBgl_return_status()\fR will return
\fBGLR_TIMEOUT\fR. Note that if the application needs \fBerrno\fR always to
have a meaningful value when \fBgl_get_line()\fR returns \fINULL\fR, the
callback function should set \fBerrno\fR appropriately.
```

```
.RE

.sp
.ne 2
.na
\fB\fBGLTO_REFRESH\fR\fR
.ad
.RS 17n
Redraw the input line, then continue waiting for input. You should return this
value if your callback wrote to the terminal.
.RE

.sp
.ne 2
.na
\fB\fBGLTO_CONTINUE\fR\fR
.ad
.RS 17n
In normal blocking-I/O mode, continue to wait for input, without redrawing the
user's input line. In non-blocking server I/O mode (see
\fBgl_io_mode\fR(3TECLA)), \fBgl_get_line()\fR acts as though I/O blocked. This
means that \fBgl_get_line()\fR will immediately return \fINULL\fR, and a
following call to \fBgl_return_status()\fR will return \fBGLR_BLOCKED\fR.
.RE

.sp
.LP
Note that before calling the callback, \fBgl_get_line()\fR blocks most signals
and leaves its own signal handlers installed, so if you need to catch a
particular signal you will need to both temporarily install your own signal
handler and unblock the signal. Be sure to re-block the signal (if it was
originally blocked) and reinstate the original signal handler, if any, before
returning.
.sp
.LP
Your callback should not try to read from the terminal, which is left in raw
mode as far as input is concerned. You can however write to the terminal as
usual, since features like conversion of newline to carriage-return/linefeed
are re-enabled while the callback is running. If your callback function does
write to the terminal, be sure to output a newline first, and when your
callback returns, tell \fBgl_get_line()\fR that the input line needs to be
redrawn, by returning the \fBGLTO_REFRESH\fR status code.
.sp
.LP
Finally, note that although the timeout arguments include a nanosecond
component, few computer clocks presently have resolutions that are finer than a
few milliseconds, so asking for less than a few milliseconds is equivalent to
requesting zero seconds on many systems. If this would be a problem, you should
base your timeout selection on the actual resolution of the host clock (for
example, by calling \fBsysconf\fR(\fB_SC_CLK_TCK\fR)).
.sp
.LP
To turn off timeouts, simply call \fBgl_inactivity_timeout()\fR with a
\fIcallback\fR argument of 0. The \fIdata\fR argument is ignored in this case.
.SS "Signal Handling Defaults"
.LP
By default, the \fBgl_get_line()\fR function intercepts a number of signals.
This is particularly important for signals that would by default terminate the
process, since the terminal needs to be restored to a usable state before this
happens. This section describes the signals that are trapped by default and how
\fBgl_get_line()\fR responds to them. Changing these defaults is the topic of
the following section.
.sp
.LP
When the following subset of signals are caught, \fBgl_get_line()\fR first
restores the terminal settings and signal handling to how they were before
```

```
\fBgl_get_line()\fR was called, resends the signal to allow the calling
application's signal handlers to handle it, then, if the process still exists,
returns \fINULL\fR and sets \fBerrno\fR as specified below.
.sp
.ne 2
.na
\fB\fBSIGINT\fR\fR
.ad
.RS 11n
This signal is generated both by the keyboard interrupt key (usually \fB^C\fR),
and the keyboard break key. The \fBerrno\fR value is \fBEINTR\fR.
.RE

.sp
.ne 2
.na
\fB\fBSIGHUP\fR\fR
.ad
.RS 11n
This signal is generated when the controlling terminal exits. The \fBerrno\fR
value is \fBENOTTY\fR.
.RE

.sp
.ne 2
.na
\fB\fBSIGPIPE\fR\fR
.ad
.RS 11n
This signal is generated when a program attempts to write to a pipe whose
remote end is not being read by any process. This can happen for example if you
have called \fBgl_change_terminal()\fR to redirect output to a pipe hidden
under a pseudo terminal. The \fBerrno\fR value is \fBEPIPE\fR.
.RE

.sp
.ne 2
.na
\fB\fBSIGQUIT\fR\fR
.ad
.RS 11n
This signal is generated by the keyboard quit key (usually \fB^\e\fR). The
\fBerrno\fR value is \fBEINTR\fR.
.RE

.sp
.ne 2
.na
\fB\fBSIGABRT\fR\fR
.ad
.RS 11n
This signal is generated by the standard C, abort function. By default it both
terminates the process and generates a core dump. The \fBerrno\fR value is
\fBEINTR\fR.
.RE

.sp
.ne 2
.na
\fB\fBSIGTERM\fR\fR
.ad
.RS 11n
This is the default signal that the UNIX kill command sends to processes. The
\fBerrno\fR value is \fBEINTR\fR.
.RE
```

```
.sp
.LP
Note that in the case of all of the above signals, POSIX mandates that by
default the process is terminated, with the addition of a core dump in the case
of the \fBSIGQUIT\fR signal. In other words, if the calling application does
not override the default handler by supplying its own signal handler, receipt
of the corresponding signal will terminate the application before
\fBgl_get_line()\fR returns.
.sp
.LP
If \fBgl_get_line()\fR aborts with \fBerrno\fR set to \fBEINTR\fR, you can find
out what signal caused it to abort, by calling the \fBgl_last_signal()\fR
function. This returns the numeric code (for example, \fBSIGINT\fR) of the last
signal that was received during the most recent call to \fBgl_get_line()\fR, or
-1 if no signals were received.
.sp
.LP
On systems that support it, when a \fBSIGWINCH\fR (window change) signal is
received, \fBgl_get_line()\fR queries the terminal to find out its new size,
redraws the current input line to accommodate the new size, then returns to
waiting for keyboard input from the user. Unlike other signals, this signal is
not resent to the application.
.sp
.LP
Finally, the following signals cause \fBgl_get_line()\fR to first restore the
terminal and signal environment to that which prevailed before
\fBgl_get_line()\fR was called, then resend the signal to the application. If
the process still exists after the signal has been delivered, then
\fBgl_get_line()\fR then re-establishes its own signal handlers, switches the
terminal back to raw mode, redisplays the input line, and goes back to awaiting
terminal input from the user.
.sp
.ne 2
.na
\fB\fBSIGCONT\fR\fR
.ad
.RS 13n
This signal is generated when a suspended process is resumed.
.RE

.sp
.ne 2
.na
\fB\fBSIGPOLL\fR\fR
.ad
.RS 13n
On SVR4 systems, this signal notifies the process of an asynchronous I/O event.
Note that under 4.3+BSD, \fBSIGIO\fR and \fBSIGPOLL\fR are the same. On other
systems, \fBSIGIO\fR is ignored by default, so \fBgl_get_line()\fR does not
trap it by default.
.RE

.sp
.ne 2
.na
\fB\fBSIGPWR\fR\fR
.ad
.RS 13n
This signal is generated when a power failure occurs (presumably when the
system is on a UPS).
.RE

.sp
.ne 2
.na
\fB\fBSIGALRM\fR\fR
```

```
.ad
.RS 13n
This signal is generated when a timer expires.
.RE

.sp
.ne 2
.na
\fB\fBSIGUSR1\fR\fR
.ad
.RS 13n
An application specific signal.
.RE

.sp
.ne 2
.na
\fB\fBSIGUSR2\fR\fR
.ad
.RS 13n
Another application specific signal.
.RE

.sp
.ne 2
.na
\fB\fBSIGVTALRM\fR\fR
.ad
.RS 13n
This signal is generated when a virtual timer expires. See \fBsetitimer\fR(2).
.RE

.sp
.ne 2
.na
\fB\fBSIGXCPU\fR\fR
.ad
.RS 13n
This signal is generated when a process exceeds its soft CPU time limit.
.RE

.sp
.ne 2
.na
\fB\fBSIGXFSZ\fR\fR
.ad
.RS 13n
This signal is generated when a process exceeds its soft file-size limit.
.RE

.sp
.ne 2
.na
\fB\fBSIGTSTP\fR\fR
.ad
.RS 13n
This signal is generated by the terminal suspend key, which is usually
\fB^Z\fR, or the delayed terminal suspend key, which is usually \fB^Y\fR.
.RE

.sp
.ne 2
.na
\fB\fBSIGTTIN\fR\fR
.ad
.RS 13n
```

```
This signal is generated if the program attempts to read from the terminal
while the program is running in the background.
.RE

.sp
.ne 2
.na
\fB\fBSIGTTOU\fR\fR
.ad
.RS 13n
This signal is generated if the program attempts to write to the terminal while
the program is running in the background.
.RE

.sp
.LP
Obviously not all of the above signals are supported on all systems, so code to
support them is conditionally compiled into the tecla library.
.sp
.LP
Note that if \fBSIGKILL\fR or \fBSIGPOLL\fR, which by definition cannot be
caught, or any of the hardware generated exception signals, such as
\fBSIGSEGV\fR, \fBSIGBUS\fR, and \fBSIGFPE\fR, are received and unhandled while
\fBgl_get_line()\fR has the terminal in raw mode, the program will be
terminated without the terminal having been restored to a usable state. In
practice, job-control shells usually reset the terminal settings when a process
relinquishes the controlling terminal, so this is only a problem with older
shells.
.SS "Customized Signal Handling"
.LP
The previous section listed the signals that \fBgl_get_line()\fR traps by
default, and described how it responds to them. This section describes how to
both add and remove signals from the list of trapped signals, and how to
specify how \fBgl_get_line()\fR should respond to a given signal.
.sp
.LP
If you do not need \fBgl_get_line()\fR to do anything in response to a signal
that it normally traps, you can tell to \fBgl_get_line()\fR to ignore that
signal by calling \fBgl_ignore_signal()\fR.
.sp
.LP
The \fIsigno\fR argument is the number of the signal (for example,
\fBSIGINT\fR) that you want to have ignored. If the specified signal is not
currently one of those being trapped, this function does nothing.
.sp
.LP
The \fBgl_trap_signal()\fR function allows you to either add a new signal to
the list that \fBgl_get_line()\fR traps or modify how it responds to a signal
that it already traps.
.sp
.LP
The \fIsigno\fR argument is the number of the signal that you want to have
trapped. The \fIflags\fR argument is a set of flags that determine the
environment in which the application’s signal handler is invoked. The
\fIafter\fR argument tells \fBgl_get_line()\fR what to do after the
application’s signal handler returns. The \fIerrno_value\fR tells
\fBgl_get_line()\fR what to set \fBerrno\fR to if told to abort.
.sp
.LP
The \fIflags\fR argument is a bitwise OR of zero or more of the following
enumerators:
.sp
.ne 2
.na
\fB\fBGLS_RESTORE_SIG\fR\fR
.ad
```

```
.RS 20n
Restore the caller’s signal environment while handling the signal.
.RE

.sp
.ne 2
.na
\fB\fBGLS_RESTORE_TTY\fR\fR
.ad
.RS 20n
Restore the caller’s terminal settings while handling the signal.
.RE

.sp
.ne 2
.na
\fB\fBGLS_RESTORE_LINE\fR\fR
.ad
.RS 20n
Move the cursor to the start of the line following the input line before
invoking the application’s signal handler.
.RE

.sp
.ne 2
.na
\fB\fBGLS_REDRAW_LINE\fR\fR
.ad
.RS 20n
Redraw the input line when the application’s signal handler returns.
.RE

.sp
.ne 2
.na
\fB\fBGLS_UNBLOCK_SIG\fR\fR
.ad
.RS 20n
Normally, if the calling program has a signal blocked (see
\fBsigprocmask\fR(2)), \fBgl_get_line()\fR does not trap that signal. This flag
tells \fBgl_get_line()\fR to trap the signal and unblock it for the duration of
the call to \fBgl_get_line()\fR.
.RE

.sp
.ne 2
.na
\fB\fBGLS_DONT_FORWARD\fR\fR
.ad
.RS 20n
If this flag is included, the signal will not be forwarded to the signal
handler of the calling program.
.RE

.sp
.LP
Two commonly useful flag combinations are also enumerated as follows:
.sp
.ne 2
.na
\fB\fBGLS_RESTORE_ENV\fR\fR
.ad
.RS 21n
\fBGLS_RESTORE_SIG\fR | \fBGLS_RESTORE_TTY\fR |\fBGLS_REDRAW_LINE\fR
.RE
```

```
.sp
.ne 2
.na
\fB\fBGLS_SUSPEND_INPUT\fR\fR
.ad
.RS 21n
\fBGLS_RESTORE_ENV\fR | \fBGLS_RESTORE_LINE\fR
.RE

.sp
.LP
If your signal handler, or the default system signal handler for this signal,
if you have not overridden it, never either writes to the terminal, nor
suspends or terminates the calling program, then you can safely set the
\fIflags\fR argument to 0.
.RS +4
.TP
.ie t \(bu
.el o
The cursor does not get left in the middle of the input line.
.RE
.RS +4
.TP
.ie t \(bu
.el o
So that the user can type in input and have it echoed.
.RE
.RS +4
.TP
.ie t \(bu
.el o
So that you do not need to end each output line with \er\en, instead of just
\en.
.RE
.sp
.LP
The \fBGL_RESTORE_ENV\fR combination is the same as \fBGL_SUSPEND_INPUT\fR,
except that it does not move the cursor. If your signal handler does not read
or write anything to the terminal, the user will not see any visible indication
that a signal was caught. This can be useful if you have a signal handler that
only occasionally writes to the terminal, where using \fBGL_SUSPEND_LINE\fR
would cause the input line to be unnecessarily duplicated when nothing had been
written to the terminal. Such a signal handler, when it does write to the
terminal, should be sure to start a new line at the start of its first write,
by writing a new line before returning. If the signal arrives while the user is
entering a line that only occupies a signal terminal line, or if the cursor is
on the last terminal line of a longer input line, this will have the same
effect as \fBGL_SUSPEND_INPUT\fR. Otherwise it will start writing on a line
that already contains part of the displayed input line. This does not do any
harm, but it looks a bit ugly, which is why the \fBGL_SUSPEND_INPUT\fR
combination is better if you know that you are always going to be writting to
the terminal.
.sp
.LP
The \fIafter\fR argument, which determines what \fBgl_get_line()\fR does after
the application’s signal handler returns (if  it returns), can take any one of
the following values:
.sp
.ne 2
.na
\fB\fBGLS_RETURN\fR\fR
.ad
.RS 16n
Return the completed input line, just as though the user had pressed the return
key.
.RE
```

```
.sp
.ne 2
.na
\fB\fBGLS_ABORT\fR\fR
.ad
.RS 16n
Cause \fBgl_get_line()\fR to abort. When this happens, \fBgl_get_line()\fR
returns \fINULL\fR, and a following call to \fBgl_return_status()\fR will
return \fBGLR_SIGNAL\fR. Note that if the application needs \fBerrno\fR always
to have a meaningful value when \fBgl_get_line()\fR returns \fINULL\fR, the
callback function should set \fBerrno\fR appropriately.
.RE

.sp
.ne 2
.na
\fB\fBGLS_CONTINUE\fR\fR
.ad
.RS 16n
Resume command line editing.
.RE

.sp
.LP
The \fIerrno_value\fR argument is intended to be combined with the
\fBGLS_ABORT\fR option, telling \fBgl_get_line()\fR what to set the standard
\fBerrno\fR variable to before returning \fINULL\fR to the calling program. It
can also, however, be used with the \fBGL_RETURN\fR option, in case you want to
have a way to distinguish between an input line that was entered using the
return key, and one that was entered by the receipt of a signal.
.SS "Reliable Signal Handling"
.LP
Signal handling is surprisingly hard to do reliably without race conditions. In
\fBgl_get_line()\fR a lot of care has been taken to allow applications to
perform reliable signal handling around \fBgl_get_line()\fR. This section
explains how to make use of this.
.sp
.LP
As an example of the problems that can arise if the application is not written
correctly, imagine that one’s application has a \fBSIGINT\fR signal handler
that sets a global flag. Now suppose that the application tests this flag just
before invoking \fBgl_get_line()\fR. If a \fBSIGINT\fR signal happens to be
received in the small window of time between the statement that tests the value
of this flag, and the statement that calls \fBgl_get_line()\fR, then
\fBgl_get_line()\fR will not see the signal, and will not be interrupted. As a
result, the application will not be able to respond to the signal until the
user gets around to finishing entering the input line and \fBgl_get_line()\fR
returns. Depending on the application, this might or might not be a disaster,
but at the very least it would puzzle the user.
.sp
.LP
The way to avoid such problems is to do the following.
.RS +4
.TP
1.
If needed, use the \fBgl_trap_signal()\fR function to configure
\fBgl_get_line()\fR to abort when important signals are caught.
.RE
.RS +4
.TP
2.
Configure \fBgl_get_line()\fR such that if any of the signals that it
catches are blocked when \fBgl_get_line()\fR is called, they will be unblocked
automatically during times when \fBgl_get_line()\fR is waiting for I/O. This
can be done either on a per signal basis, by calling the \fBgl_trap_signal()\fR
```

```
function, and specifying the \fBGLS_UNBLOCK\fR attribute of the signal, or
globally by calling the \fBgl_catch_blocked()\fR function. This function simply
adds the \fBGLS_UNBLOCK\fR attribute to all of the signals that it is currently
configured to trap.
.RE
.RS +4
.TP
3.
Just before calling \fBgl_get_line()\fR, block delivery of all of the
signals that \fBgl_get_line()\fR is configured to trap. This can be done using
the POSIX sigprocmask function in conjunction with the \fBgl_list_signals()\fR
function. This function returns the set of signals that it is currently
configured to catch in the set argument, which is in the form required by
\fBsigprocmask\fR(2).
.RE
.RS +4
.TP
4.
In the example, one would now test the global flag that the signal handler
sets, knowing that there is now no danger of this flag being set again until
\fBgl_get_line()\fR unblocks its signals while performing I/O.
.RE
.RS +4
.TP
5.
Eventually \fBgl_get_line()\fR returns, either because a signal was caught,
an error occurred, or the user finished entering their input line.
.RE
.RS +4
.TP
6.
Now one would check the global signal flag again, and if it is set, respond
to it, and zero the flag.
.RE
.RS +4
.TP
7.
Use \fBsigprocmask()\fR to unblock the signals that were blocked in step 3.
.RE
.sp
.LP
The same technique can be used around certain POSIX signal-aware functions,
such as \fBsigsetjmp\fR(3C) and \fBsigsuspend\fR(2), and in particular, the
former of these two functions can be used in conjunction with
\fBsiglongjmp\fR(3C) to implement race-condition free signal handling around
other long-running system calls. The \fBgl_get_line()\fR function manages to
reliably trap signals around calls to functions like \fBread\fR(2) and
\fBselect\fR(3C) without race conditions.
.sp
.LP
The \fBgl_get_line()\fR function first uses the POSIX \fBsigprocmask()\fR
function to block the delivery of all of the signals that it is currently
configured to catch. This is redundant if the application has already blocked
them, but it does no harm. It undoes this step just before returning.
.sp
.LP
Whenever \fBgl_get_line()\fR needs to call read or select to wait for input
from the user, it first calls the POSIX \fBsigsetjmp()\fR function, being sure
to specify a non-zero value for its \fIsavemask\fR argument.
.sp
.LP
If \fBsigsetjmp()\fR returns zero, \fBgl_get_line()\fR then does the following.
.RS +4
.TP
1.
It uses the POSIX \fBsigaction\fR(2) function to register a temporary signal
```

```
handler to all of the signals that it is configured to catch. This signal
handler does two things.
.RS +4
.TP
a.
It records the number of the signal that was received in a file-scope
variable.
.RE
.RS +4
.TP
b.
It then calls the POSIX \fBsiglongjmp()\fR function using the buffer that
was passed to \fBsigsetjmp()\fR for its first argument and a non-zero value for
its second argument.
.RE
When this signal handler is registered, the \fIsa_mask\fR member of the
\fBstruct sigaction\fR \fIact\fR argument of the call to \fBsigaction()\fR is
configured to contain all of the signals that \fBgl_get_line()\fR is catching.
This ensures that only one signal will be caught at once by our signal handler,
which in turn ensures that multiple instances of our signal handler do not
tread on each other’s toes.
.RE
.RS +4
.TP
2.
Now that the signal handler has been set up, \fBgl_get_line()\fR unblocks
all of the signals that it is configured to catch.
.RE
.RS +4
.TP
3.
It then calls the \fBread()\fR or \fBselect()\fR function to wait for
keyboard input.
.RE
.RS +4
.TP
4.
If this function returns (that is, no signal is received),
\fBgl_get_line()\fR blocks delivery of the signals of interest again.
.RE
.RS +4
.TP
5.
It then reinstates the signal handlers that were displaced by the one that
was just installed.
.RE
.sp
.LP
Alternatively, if \fBsigsetjmp()\fR returns non-zero, this means that one of
the signals being trapped was caught while the above steps were executing. When
this happens, \fBgl_get_line()\fR does the following.
.sp
.LP
First, note that when a call to \fBsiglongjmp()\fR causes \fBsigsetjmp()\fR to
return, provided that the \fIsavemask\fR argument of \fBsigsetjmp()\fR was
non-zero, the signal process mask is restored to how it was when
\fBsigsetjmp()\fR was called. This is the important difference between
\fBsigsetjmp()\fR and the older problematic \fBsetjmp\fR(3C), and is the
essential ingredient that makes it possible to avoid signal handling race
conditions. Because of this we are guaranteed that all of the signals that we
blocked before calling \fBsigsetjmp()\fR are blocked again as soon as any
signal is caught. The following statements, which are then executed, are thus
guaranteed to be executed without any further signals being caught.
.RS +4
.TP
1.
```

```
If so instructed by the \fBgl_get_line()\fR configuration attributes of the
signal that was caught, \fBgl_get_line()\fR restores the terminal attributes to
the state that they had when \fBgl_get_line()\fR was called. This is
particularly important for signals that suspend or terminate the process, since
otherwise the terminal would be left in an unusable state.
.RE
.RS +4
.TP
2.
It then reinstates the application’s signal handlers.
.RE
.RS +4
.TP
3.
Then it uses the C standard-library \fBraise\fR(3C) function to re-send the
application the signal that was caught.
.RE
.RS +4
.TP
4.
Next it unblocks delivery of the signal that we just sent. This results in
the signal that was just sent by \fBraise()\fR being caught by the
application’s original signal handler, which can now handle it as it sees fit.
.RE
.RS +4
.TP
5.
If the signal handler returns (that is, it does not terminate the process),
\fBgl_get_line()\fR blocks delivery of the above signal again.
.RE
.RS +4
.TP
6.
It then undoes any actions performed in the first of the above steps and
redisplays the line, if the signal configuration calls for this.
.RE
.RS +4
.TP
7.
\fBgl_get_line()\fR then either resumes trying to read a character, or
aborts, depending on the configuration of the signal that was caught.
.RE
.sp
.LP
What the above steps do in essence is to take asynchronously delivered signals
and handle them synchronously, one at a time, at a point in the code where
\fBgl_get_line()\fR has complete control over its environment.
.SS "The Terminal Size"
.LP
On most systems the combination of the \fBTIOCGWINSZ\fR ioctl and the
\fBSIGWINCH\fR signal is used to maintain an accurate idea of the terminal
size. The terminal size is newly queried every time that \fBgl_get_line()\fR is
called and whenever a \fBSIGWINCH\fR signal is received.
.sp
.LP
On the few systems where this mechanism is not available, at startup
\fBnew_GetLine()\fR first looks for the \fBLINES\fR and \fBCOLUMNS\fR
environment variables. If these are not found, or they contain unusable values,
then if a terminal information database like \fBterminfo\fR or \fBtermcap\fR is
available, the default size of the terminal is looked up in this database. If
this too fails to provide the terminal size, a default size of 80 columns by 24
lines is used.
.sp
.LP
Even on systems that do support ioctl(\fBTIOCGWINSZ\fR), if the terminal is on
the other end of a serial line, the terminal driver generally has no way of
```

```
detecting when a resize occurs or of querying what the current size is. In such
cases no \fBSIGWINCH\fR is sent to the process, and the dimensions returned by
ioctl(\fBTIOCGWINSZ\fR) are not correct. The only way to handle such instances
is to provide a way for the user to enter a command that tells the remote
system what the new size is. This command would then call the
\fBgl_set_term_size()\fR function to tell \fBgl_get_line()\fR about the change
in size.
.sp
.LP
The \fIncolumn\fR and \fInline\fR arguments are used to specify the new
dimensions of the terminal, and must not be less than 1. On systems that do
support ioctl(\fBTIOCGWINSZ\fR), this function first calls
ioctl(\fBTIOCSWINSZ\fR) to tell the terminal driver about the change in size.
In non-blocking server-I/O mode, if a line is currently being input, the input
line is then redrawn to accommodate the changed size. Finally the new values are
recorded in \fIgl\fR for future use by \fBgl_get_line()\fR.
.sp
.LP
The \fBgl_terminal_size()\fR function allows you to query the current size of
the terminal, and install an alternate fallback size for cases where the size
is not available. Beware that the terminal size will not be available if
reading from a pipe or a file, so the default values can be important even on
systems that do support ways of finding out the terminal size.
.sp
.LP
This function first updates \fBgl_get_line()\fR’s fallback terminal dimensions,
then records its findings in the return value.
.sp
.LP
The \fIdef_ncolumn\fR and \fIdef_nline\fR arguments specify the default number
of terminal columns and lines to use if the terminal size cannot be determined
by ioctl(\fBTIOCGWINSZ\fR) or environment variables.
.SS "Hiding What You Type"
.LP
When entering sensitive information, such as passwords, it is best not to have
the text that you are entering echoed on the terminal. Furthermore, such text
should not be recorded in the history list, since somebody finding your
terminal unattended could then recall it, or somebody snooping through your
directories could see it in your history file. With this in mind, the
\fBgl_echo_mode()\fR function allows you to toggle on and off the display and
archival of any text that is subsequently entered in calls to
\fBgl_get_line()\fR.
.sp
.LP
The \fIenable\fR argument specifies whether entered text should be visible or
not. If it is 0, then subsequently entered lines will not be visible on the
terminal, and will not be recorded in the history list. If it is 1, then
subsequent input lines will be displayed as they are entered, and provided that
history has not been turned off with a call to \fBgl_toggle_history()\fR, then
they will also be archived in the history list. Finally, if the enable argument
is -1, then the echoing mode is left unchanged, which allows you to
non-destructively query the current setting through the return value. In all
cases, the return value of the function is 0 if echoing was disabled before the
function was called, and 1 if it was enabled.
.sp
.LP
When echoing is turned off, note that although tab completion will invisibly
complete your prefix as far as possible, ambiguous completions will not be
displayed.
.SS "Single Character Queries"
.LP
Using \fBgl_get_line()\fR to query the user for a single character reply, is
inconvenient for the user, since they must hit the enter or return key before
the character that they typed is returned to the program. Thus the
\fBgl_query_char()\fR function has been provided for single character queries
like this.
```

```
.sp
.LP
This function displays the specified prompt at the start of a new line, and
waits for the user to type a character. When the user types a character,
\fBgl_query_char()\fR displays it to the right of the prompt, starts a newline,
then returns the character to the calling program. The return value of the
function is the character that was typed. If the read had to be aborted for
some reason, EOF is returned instead. In the latter case, the application can
call the previously documented \fBgl_return_status()\fR, to find out what went
wrong. This could, for example, have been the reception of a signal, or the
optional inactivity timer going off.
.sp
.LP
If the user simply hits enter, the value of the \fIdefchar\fR argument is
substituted. This means that when the user hits either newline or return, the
character specified in \fIdefchar\fR, is displayed after the prompt, as though
the user had typed it, as well as being returned to the calling application. If
such a replacement is not important, simply pass ’\en’ as the value of
\fIdefchar\fR.
.sp
.LP
If the entered character is an unprintable character, it is displayed
symbolically. For example, control-A is displayed as \fB^A\fR, and characters
beyond 127 are displayed in octal, preceded by a backslash.
.sp
.LP
As with \fBgl_get_line()\fR, echoing of the entered character can be disabled
using the \fBgl_echo_mode()\fR function.
.sp
.LP
If the calling process is suspended while waiting for the user to type their
response, the cursor is moved to the line following the prompt line, then when
the process resumes, the prompt is redisplayed, and \fBgl_query_char()\fR
resumes waiting for the user to type a character.
.sp
.LP
Note that in non-blocking server mode, if an incomplete input line is in the
process of being read when \fBgl_query_char()\fR is called, the partial input
line is discarded, and erased from the terminal, before the new prompt is
displayed. The next call to \fBgl_get_line()\fR will thus start editing a new
line.
.SS "Reading Raw Characters"
.LP
Whereas the \fBgl_query_char()\fR function visibly prompts the user for a
character, and displays what they typed, the \fBgl_read_char()\fR function
reads a signal character from the user, without writing anything to the
terminal, or perturbing any incompletely entered input line. This means that it
can be called not only from between calls to \fBgl_get_line()\fR, but also from
callback functions that the application has registered to be called by
\fBgl_get_line()\fR.
.sp
.LP
On success, the return value of \fBgl_read_char()\fR is the character that was
read. On failure, EOF is returned, and the \fBgl_return_status()\fR function
can be called to find out what went wrong. Possibilities include the optional
inactivity timer going off, the receipt of a signal that is configured to abort
\fBgl_get_line()\fR, or terminal I/O blocking, when in non-blocking server-I/O
mode.
.sp
.LP
Beware that certain keyboard keys, such as function keys, and cursor keys,
usually generate at least three characters each, so a single call to
\fBgl_read_char()\fR will not be enough to identify such keystrokes.
.SS "Clearing The Terminal"
.LP
The calling program can clear the terminal by calling
```

```
\fBgl_erase_terminal()\fR. In non-blocking server-I/O mode, this function also
arranges for the current input line to be redrawn from scratch when
\fBgl_get_line()\fR is next called.
.SS "Displaying Text Dynamically"
.LP
Between calls to \fBgl_get_line()\fR, the \fBgl_display_text()\fR function
provides a convenient way to display paragraphs of text, left-justified and
split over one or more terminal lines according to the constraints of the
current width of the terminal. Examples of the use of this function may be
found in the demo programs, where it is used to display introductions. In those
examples the advanced use  of optional prefixes, suffixes and filled lines to
draw a box around the text is also illustrated.
.sp
.LP
If \fIgl\fR is not currently connected to a terminal, for example if the output
of a program that uses \fBgl_get_line()\fR is being piped to another program or
redirected to a file, then the value of the \fIdef_width\fR parameter is used
as the terminal width.
.sp
.LP
The \fIindentation\fR argument specifies the number of characters to use to
indent each line of output. The \fIfill_char\fR argument specifies the character
indent each line of ouput. The \fIfill_char\fR argument specifies the character
that will be used to perform this indentation.
.sp
.LP
The \fIprefix\fR argument can be either \fINULL\fR or a string to place at the
beginning of each new line (after any indentation). Similarly, the \fIsuffix\fR
argument can be either \fINULL\fR or a string to place at the end of each line.
The suffix is placed flush against the right edge of the terminal, and any
space between its first character and the last word on that line is filled with
the character specified by the \fIfill_char\fR argument. Normally the
fill-character is a space.
.sp
.LP
The \fIstart\fR argument tells \fBgl_display_text()\fR how many characters have
already been written to the current terminal line, and thus tells it the
starting column index of the cursor. Since the return value of
\fBgl_display_text()\fR is the ending column index of the cursor, by passing
the return value of one call to the start argument of the next call, a
paragraph that is broken between more than one string can be composed by
calling \fBgl_display_text()\fR for each successive portion of the paragraph.
Note that literal newline characters are necessary at the end of each paragraph
to force a new line to be started.
.sp
.LP
On error, \fBgl_display_text()\fR returns -1.
.SS "Callback Function Facilities"
.LP
Unless otherwise stated, callback functions such as tab completion callbacks
and event callbacks should not call any functions in this module. The following
functions, however, are designed specifically to be used by callback functions.
.sp
.LP
Calling the \fBgl_replace_prompt()\fR function from a callback tells
\fBgl_get_line()\fR to display a different prompt when the callback returns.
Except in non-blocking server mode, it has no effect if used between calls to
\fBgl_get_line()\fR. In non-blocking server mode, when used between two calls
to \fBgl_get_line()\fR that are operating on the same input line, the current
input line will be re-drawn with the new prompt on the following call to
\fBgl_get_line()\fR.
.SS "International Character Sets"
.LP
Since \fBlibtecla\fR(3LIB) version 1.4.0, \fBgl_get_line()\fR has been 8-bit
clean. This means that all 8-bit characters that are printable in the user's
current locale are now displayed verbatim and included in the returned input
```

```
line. Assuming that the calling program correctly contains a call like the
following,
.sp
.in +2
.nf
setlocale(LC_CTYPE, "")
.fi
.in -2

.sp
.LP
then the current locale is determined by the first of the environment variables
\fBLC_CTYPE\fR, \fBLC_ALL\fR, and \fBLANG\fR that is found to contain a valid
locale name. If none of these variables are defined, or the program neglects to
call \fBsetlocale\fR(3C), then the default C locale is used, which is US 7-bit
ASCII. On most UNIX-like platforms, you can get a list of valid locales by
typing the command:
.sp
.in +2
.nf
locale -a
.fi
.in -2
.sp

.sp
.LP
at the shell prompt. Further documentation on how the user can make use of this
to enter international characters can be found in the \fBtecla\fR(5) man page.
.SS "Thread Safety"
.LP
Unfortunately neither \fBterminfo\fR nor \fBtermcap\fR were designed to be
reentrant, so you cannot safely use the functions of the getline module in
multiple threads (you can use the separate file-expansion and word-completion
modules in multiple threads, see the corresponding man pages for details).
However due to the use of POSIX reentrant functions for looking up home
directories, it is safe to use this module from a single thread of a
multi-threaded program, provided that your other threads do not use any
\fBtermcap\fR or \fBterminfo\fR functions.
.SH ATTRIBUTES
.LP
See \fBattributes\fR(5) for descriptions of the following attributes:
.sp

.sp
.TS
box;
c | c
l | l .
ATTRIBUTE TYPE  ATTRIBUTE VALUE
_
Interface Stability     Committed
_
MT-Level        MT-Safe
.TE

.SH SEE ALSO
.LP
\fBcpl_complete_word\fR(3TECLA), \fBef_expand_file\fR(3TECLA),
\fBgl_io_mode\fR(3TECLA), \fBlibtecla\fR(3LIB), \fBpca_lookup_file\fR(3TECLA),
\fBattributes\fR(5), \fBtecla\fR(5)
```

```
**********************************************************
   53473 Tue Dec 11 09:48:09 2018
new/usr/src/man/man5/tecla.5
10057 Man page misspellings ouput particuliar overriden
Reviewed by: Gerg¯  Mihˆ¡ly Doma <domag02@gmail.com>
**********************************************************
'\" te
.\" Copyright (c) 2000, 2001, 2002, 2003, 2004 by Martin C. Shepherd. All Rights
.\" Permission is hereby granted, free of charge, to any person obtaining a copy
.\" "Software"), to deal in the Software without restriction, including
.\" without limitation the rights to use, copy, modify, merge, publish,
.\" distribute, and/or sell copies of the Software, and to permit persons
.\" to whom the Software is furnished to do so, provided that the above
.\" copyright notice(s) and this permission notice appear in all copies of
.\" the Software and that both the above copyright notice(s) and this
.\" permission notice appear in supporting documentation.
.\"
.\" THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
.\" OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
.\" MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT
.\" OF THIRD PARTY RIGHTS. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR
.\" HOLDERS INCLUDED IN THIS NOTICE BE LIABLE FOR ANY CLAIM, OR ANY SPECIAL
.\" INDIRECT OR CONSEQUENTIAL DAMAGES, OR ANY DAMAGES WHATSOEVER RESULTING
.\" FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT,
.\" NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION
.\" WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.
.\"
.\" Except as contained in this notice, the name of a copyright holder
.\" shall not be used in advertising or otherwise to promote the sale, use
.\" or other dealings in this Software without prior written authorization
.\" of the copyright holder.
.\" Portions Copyright (c) 2005, Sun Microsystems, Inc. All Rights Reserved.
.TH TECLA 5 "April 9, 2016"
.SH NAME
tecla, teclarc \- User interface provided by the tecla library.
.SH DESCRIPTION
.LP
This man page describes the command-line editing features that are available to
users of programs that read keyboard input via the tecla library. Users of the
\fBtcsh\fR shell will find the default key bindings very familiar. Users of the
\fBtcsh shell\fR will find the default key bindings very familiar. Users of the
\fBbash\fR shell will also find it quite familiar, but with a few minor
differences, most notably in how forward and backward searches through the list
of historical commands are performed. There are two major editing modes, one
with \fBemacs\fR-like key bindings and another with \fBvi\fR-like key bindings.
By default \fBemacs\fR mode is enabled, but \fBvi\fR(1) mode can alternatively
be selected via the user's configuration file. This file can also be used to
change the bindings of individual keys to suit the user's preferences. By
default, tab completion is provided. If the application hasn't reconfigured
this to complete other types of symbols, then tab completion completes file
names.
.SS "Key Sequence Notation"
.LP
In the rest of this man page, and also in all tecla configuration files, key
sequences are expressed as follows.
.sp
.ne 2
.na
\fB\fB^A\fR or \fBC-a\fR\fR
.ad
.RS 13n
This is a 'CONTROL-A', entered by pressing the CONTROL key at the same time as
the 'A' key.
.RE

.sp
```

```
.ne 2
.na
\fB\eE\fR or \fBM-\fR
.ad
.RS 13n
In key sequences, both of these notations can be entered either by pressing the
ESCAPE key, then the following key, or by pressing the META key at the same
time as the following key. Thus the key sequence \fBM-p\fR can be typed in two
ways, by pressing the ESCAPE key, followed by pressing 'P', or by pressing the
META key at the same time as 'P'.
.RE

.sp
.ne 2
.na
\fBup\fR
.ad
.RS 13n
This refers to the up-arrow key.
.RE

.sp
.ne 2
.na
\fBdown\fR
.ad
.RS 13n
This refers to the down-arrow key.
.RE

.sp
.ne 2
.na
\fBleft\fR
.ad
.RS 13n
This refers to the left-arrow key.
.RE

.sp
.ne 2
.na
\fBright\fR
.ad
.RS 13n
This refers to the right-arrow key.
.RE

.sp
.ne 2
.na
\fBa\fR
.ad
.RS 13n
This is just a normal 'A' key.
.RE

.SS "The Tecla Configuration File"
.LP
By default, tecla looks for a file called \fB\&.teclarc\fR in your home
directory (ie. \fB~/.teclarc\fR). If it finds this file, it reads it,
interpreting each line as defining a new key binding or an editing
configuration option. Since the \fBemacs\fR key-bindings are installed by
default, if you want to use the non-default \fBvi\fR editing mode, the most
important item to go in this file is the following line:
.sp
```

```
.in +2
.nf
edit-mode vi
.fi
.in -2

.sp
.LP
This will re-configure the default bindings for \fBvi\fR-mode. The complete set
of arguments that this command accepts are:
.sp
.ne 2
.na
\fBvi\fR
.ad
.RS 9n
Install key bindings like those of the \fBvi\fR editor.
.RE

.sp
.ne 2
.na
\fBemacs\fR
.ad
.RS 9n
Install key bindings like those of the \fBemacs\fR editor. This is the default.
.RE

.sp
.ne 2
.na
\fBnone\fR
.ad
.RS 9n
Use just the native line editing facilities provided by the terminal driver.
.RE

.sp
.LP
To prevent the terminal bell from being rung, such as when an unrecognized
control-sequence is typed, place the following line in the configuration file:
.sp
.in +2
.nf
nobeep
.fi
.in -2

.sp
.LP
An example of a key binding line in the configuration file is the following.
.sp
.in +2
.nf
bind M-[2~ insert-mode
.fi
.in -2

.sp
.LP
On many keyboards, the above key sequence is generated when one presses the
insert key, so with this key binding, one can toggle between the
\fBemacs\fR-mode insert and overwrite modes by hitting one key. One could also
do it by typing out the above sequence of characters one by one. As explained
above, the \fBM-\fR part of this sequence can be typed either by pressing the
ESCAPE key before the following key, or by pressing the META key at the same
```

```
time as the following key. Thus if you had set the above key binding, and the
insert key on your keyboard didn't generate the above key sequence, you could
still type it in either of the following 2 ways.
.RS +4
.TP
1.
Hit the ESCAPE key momentarily, then press '[', then '2', then finally '~'.
.RE
.RS +4
.TP
2.
Press the META key at the same time as pressing the '[' key, then press '2',
then '~'.
.RE
.sp
.LP
If you set a key binding for a key sequence that is already bound to a
function, the new binding overrides the old one. If in the new binding you omit
the name of the new function to bind to the key sequence, the original binding
becomes undefined.
.sp
.LP
Starting with versions of \fBlibtecla\fR later than 1.3.3 it is now possible to
bind key sequences that begin with a printable character. Previously key
sequences were required to start with a CONTROL or META character.
.sp
.LP
Note that the special keywords "up", "down", "left", and "right" refer to the
arrow keys, and are thus not treated as key sequences. So, for example, to
rebind the up and down arrow keys to use the history search mechanism instead
of the simple history recall method, you could place the following in your
configuration file:
.sp
.in +2
.nf
bind up history-search-backwards
bind down history-search-backwards
.fi
.in -2

.sp
.LP
To unbind an existing binding, you can do this with the bind command by
omitting to name any action to rebind the key sequence to. For example, by not
specifying an action function, the following command unbinds the default
beginning-of-line action from the \fB^A\fR key sequence:
.sp
.in +2
.nf
bind ^A
.fi
.in -2

.sp
.LP
If you create a \fB~/.teclarc\fR configuration file, but it appears to have no
effect on the program, check the documentation of the program to see if the
author chose a different name for this file.
.SS "Filename and Tilde Completion"
.LP
With the default key bindings, pressing the TAB key (aka. \fB^I\fR) results in
tecla attempting to complete the incomplete file name that precedes the cursor.
Tecla searches backwards from the cursor, looking for the start of the file
name, stopping when it hits either a space or the start of the line. If more
than one file has the specified prefix, then tecla completes the file name up
to the point at which the ambiguous matches start to differ, then lists the
```

```
possible matches.
.sp
.LP
In addition to literally written file names, tecla can complete files that
start with \fB~/\fR and \fB~user/\fR expressions and that contain \fB$envvar\fR
expressions. In particular, if you hit TAB within an incomplete \fB~user\fR,
expression, tecla will attempt to complete the username, listing any ambiguous
matches.
.sp
.LP
The completion binding is implemented using the \fBcpl_complete_word()\fR
function, which is also available separately to users of this library. See the
\fBcpl_complete_word\fR(3TECLA) man page for more details.
.SS "Filename Expansion"
.LP
With the default key bindings, pressing \fB^X*\fR causes tecla to expand the
file name that precedes the cursor, replacing \fB~/\fR and \fB~user/\fR
expressions with the corresponding home directories, and replacing
\fB$envvar\fR expressions with the value of the specified environment variable,
then if there are any wildcards, replacing the so far expanded file name with a
space-separated list of the files which match the wild cards.
.sp
.LP
The expansion binding is implemented using the \fBef_expand_file()\fR function.
See the \fBef_expand_file\fR(3TECLA) man page for more details.
.SS "Recalling Previously Typed Lines"
.LP
Every time that a new line is entered by the user, it is appended to a list of
historical input lines maintained within the \fBGetLine\fR resource object. You
can traverse up and down this list using the up and down arrow keys.
Alternatively, you can do the same with the \fB^P\fR, and \fB^N\fR keys, and in
\fBvi\fR command mode you can alternatively use the k and j characters. Thus
pressing up-arrow once, replaces the current input line with the previously
entered line. Pressing up-arrow again, replaces this with the line that was
entered before it, etc.. Having gone back one or more lines into the history
list, one can return to newer lines by pressing down-arrow one or more times.
If you do this sufficient times, you will return to the original line that you
were entering when you first hit up-arrow.
.sp
.LP
Note that in \fBvi\fR mode, all of the history recall functions switch the
library into command mode.
.sp
.LP
In \fBemacs\fR mode the \fBM-p\fR and \fBM-n\fR keys work just like the
\fB^P\fR and \fB^N\fR keys, except that they skip all but those historical
lines which share the prefix that precedes the cursor. In \fBvi\fR command mode
the upper case 'K' and 'J' characters do the same thing, except that the string
that they search for includes the character under the cursor as well as what
precedes it.
.sp
.LP
Thus for example, suppose that you were in \fBemacs\fR mode, and you had just
entered the following list of commands in the order shown:
.sp
.in +2
.nf
ls ~/tecla/
cd ~/tecla
ls -l getline.c
\fBemacs\fR ~/tecla/getline.c
.fi
.in -2

.sp
.LP
```

```
If you next typed:
.sp
.in +2
.nf
ls
.fi
.in -2

.sp
.LP
and then hit \fBM-p\fR, then rather than returning the previously typed
\fBemacs\fR line, which doesn't start with "ls", tecla would recall the "ls -l
getline.c" line. Pressing \fBM-p\fR again would recall the "ls ~/tecla/" line.
.sp
.LP
Note that if the string that you are searching for, contains any of the special
characters, *, ?, or '[', then it is interpreted as a pattern to be matched.
Thus, continuing with the above example, after typing in the list of commands
shown, if you then typed:
.sp
.in +2
.nf
*tecla*
.fi
.in -2

.sp
.LP
and hit \fBM-p\fR, then the "\fBemacs\fR ~/tecla/getline.c" line would be
recalled first, since it contains the word tecla somewhere in the line,
Similarly, hitting \fBM-p\fR again, would recall the "ls ~/tecla/" line, and
hitting it once more would recall the "ls ~/tecla/" line. The pattern syntax is
the same as that described for file name expansion, in the
\fBef_expand_file\fR(3TECLA).
.SS "History Files"
.LP
Authors of programs that use the tecla library have the option of saving
historical command-lines in a file before exiting, and subsequently reading
them back in from this file when the program is next started. There is no
standard name for this file, since it makes sense for each application to use
its own history file, so that commands from different applications don't get
mixed up.
.SS "International Character Sets"
.LP
Since \fBlibtecla\fR version 1.4.0, tecla has been 8-bit clean. This means that
all 8-bit characters that are printable in the user's current locale are now
displayed verbatim and included in the returned input line. Assuming that the
calling program correctly contains a call like the following,
.sp
.in +2
.nf
setlocale(LC_CTYPE, "");
.fi
.in -2

.sp
.LP
then the current locale is determined by the first of the environment variables
\fBLC_CTYPE\fR, \fBLC_ALL\fR, and \fBLANG\fR, that is found to contain a valid
locale name. If none of these variables are defined, or the program neglects to
call \fBsetlocale\fR, then the default C locale is used, which is US 7-bit
ASCII. On most unix-like platforms, you can get a list of valid locales by
typing the command:
.sp
.in +2
.nf
```

```
locale -a
.fi
.in -2

.sp
.LP
at the shell prompt.
.SS "Meta Keys and Locales"
.LP
Beware that in most locales other than the default C locale, META characters
become printable, and they are then no longer considered to match \fBM-c\fR
style key bindings. This allows international characters to be entered with the
compose key without unexpectedly triggering META key bindings. You can still
invoke META bindings, since there are actually two ways to do this. For example
the binding \fBM-c\fR can also be invoked by pressing the ESCAPE key
momentarily, then pressing the c key, and this will work regardless of locale.
Moreover, many modern terminal emulators, such as gnome's gnome-terminal's and
KDE's konsole terminals, already generate escape pairs like this when you use
the META key, rather than a real meta character, and other emulators usually
have a way to request this behavior, so you can continue to use the META key on
most systems.
.sp
.LP
For example, although xterm terminal emulators generate real 8-bit meta
characters by default when you use the META key, they can be configured to
output the equivalent escape pair by setting their \fBEightBitInput\fR X
resource to False. You can either do this by placing a line like the following
in your \fB~/.Xdefaults\fR file,
.sp
.in +2
.nf
XTerm*EightBitInput: False
.fi
.in -2

.sp
.LP
or by starting an \fBxterm\fR with an \fB-xrm\fR \&'*EightBitInput: False'
command-line argument. In recent versions of xterm you can toggle this feature
on and off with the 'Meta Sends Escape' option in the menu that is displayed
when you press the left mouse button and the CONTROL key within an xterm
window. In CDE, dtterms can be similarly coerced to generate escape pairs in
place of meta characters, by setting the \fBDtterm*KshMode\fR resource to True.
.SS "Entering International Characters"
.LP
If you don't have a keyboard that generates all of the international characters
that you need, there is usually a compose key that will allow you to enter
special characters, or a way to create one. For example, under X windows on
unix-like systems, if your keyboard doesn't have a compose key, you can
designate a redundant key to serve this purpose with the xmodmap command. For
example, on many PC keyboards there is a microsoft-windows key, which is
otherwise useless under Linux. On a laptop, for example, the \fBxev\fR program
might report that pressing this key generates keycode 115. To turn this key
into a COMPOSE  key, do the following:
.sp
.in +2
.nf
xmodmap -e 'keycode 115 = Multi_key'
.fi
.in -2

.sp
.LP
Type this key followed by a " character to enter an 'I' with a umlaut over it.
.SS "The Available Key Binding Functions"
.LP
```

```
The following is a list of the editing functions provided by the tecla library.
The names in the leftmost column of the list can be used in configuration files
to specify which function a given key or combination of keys should invoke.
They are also used in the next two sections to list the default key bindings in
\fBemacs\fR and \fBvi\fR modes.
.sp
.ne 2
.na
\fBuser-interrupt\fR
.ad
.RS 30n
Send a SIGINT signal to the parent process.
.RE

.sp
.ne 2
.na
\fBsuspend\fR
.ad
.RS 30n
Suspend the parent process.
.RE

.sp
.ne 2
.na
\fBstop-output\fR
.ad
.RS 30n
Pause terminal output.
.RE

.sp
.ne 2
.na
\fBstart-output\fR
.ad
.RS 30n
Resume paused terminal output.
.RE

.sp
.ne 2
.na
\fBliteral-next\fR
.ad
.RS 30n
Arrange for the next character to be treated as a normal character. This allows
control characters to be entered.
.RE

.sp
.ne 2
.na
\fBcursor-right\fR
.ad
.RS 30n
Move the cursor one character right.
.RE

.sp
.ne 2
.na
\fBcursor-left\fR
.ad
.RS 30n
```

```
Move the cursor one character left.
.RE

.sp
.ne 2
.na
\fBinsert-mode\fR
.ad
.RS 30n
Toggle between insert mode and overwrite mode.
.RE

.sp
.ne 2
.na
\fBbeginning-of-line\fR
.ad
.RS 30n
Move the cursor to the beginning of the line.
.RE

.sp
.ne 2
.na
\fBend-of-line\fR
.ad
.RS 30n
Move the cursor to the end of the line.
.RE

.sp
.ne 2
.na
\fBdelete-line\fR
.ad
.RS 30n
Delete the contents of the current line.
.RE

.sp
.ne 2
.na
\fBkill-line\fR
.ad
.RS 30n
Delete everything that follows the cursor.
.RE

.sp
.ne 2
.na
\fBbackward-kill-line\fR
.ad
.RS 30n
Delete all characters between the cursor and the start of the line.
.RE

.sp
.ne 2
.na
\fBforward-word\fR
.ad
.RS 30n
Move to the end of the word which follows the cursor.
.RE
```

```
.sp
.ne 2
.na
\fBforward-to-word\fR
.ad
.RS 30n
Move the cursor to the start of the word that follows the cursor.
.RE

.sp
.ne 2
.na
\fBbackward-word\fR
.ad
.RS 30n
Move to the start of the word which precedes the cursor.
.RE

.sp
.ne 2
.na
\fBgoto-column\fR
.ad
.RS 30n
Move the cursor to the 1-relative column in the line specified by any preceding
digit-argument sequences (see Entering Repeat Counts below).
.RE

.sp
.ne 2
.na
\fBfind-parenthesis\fR
.ad
.RS 30n
If the cursor is currently over a parenthesis character, move it to the
matching parenthesis character. If not over a parenthesis character move right
to the next close parenthesis.
.RE

.sp
.ne 2
.na
\fBforward-delete-char\fR
.ad
.RS 30n
Delete the character under the cursor.
.RE

.sp
.ne 2
.na
\fBbackward-delete-char\fR
.ad
.RS 30n
Delete the character which precedes the cursor.
.RE

.sp
.ne 2
.na
\fBlist-or-eof\fR
.ad
.RS 30n
This is intended for binding to \fB^D\fR. When invoked when the cursor is
within the line it displays all possible completions then redisplays the line
unchanged. When invoked on an empty line, it signals end-of-input (EOF) to the
```

```
caller of \fBgl_get_line()\fR.
.RE

.sp
.ne 2
.na
\fBdel-char-or-list-or-eof\fR
.ad
.RS 30n
This is intended for binding to \fB^D\fR. When invoked when the cursor is
within the line it invokes forward-delete-char. When invoked at the end of the
line it displays all possible completions then redisplays the line unchanged.
When invoked on an empty line, it signals end-of-input (EOF) to the caller of
\fBgl_get_line()\fR.
.RE

.sp
.ne 2
.na
\fBforward-delete-word\fR
.ad
.RS 30n
Delete the word which follows the cursor.
.RE

.sp
.ne 2
.na
\fBbackward-delete-word\fR
.ad
.RS 30n
Delete the word which precedes the cursor.
.RE

.sp
.ne 2
.na
\fBupcase-word\fR
.ad
.RS 30n
Convert all of the characters of the word which follows the cursor, to upper
case.
.RE

.sp
.ne 2
.na
\fBdowncase-word\fR
.ad
.RS 30n
Convert all of the characters of the word which follows the cursor, to lower
case.
.RE

.sp
.ne 2
.na
\fBcapitalize-word\fR
.ad
.RS 30n
Capitalize the word which follows the cursor.
.RE

.sp
.ne 2
.na
```

```
\fBchange-case\fR
.ad
.RS 30n
If the next character is upper case, toggle it to lower case and vice versa.
.RE

.sp
.ne 2
.na
\fBredisplay\fR
.ad
.RS 30n
Redisplay the line.
.RE

.sp
.ne 2
.na
\fBclear-screen\fR
.ad
.RS 30n
Clear the terminal, then redisplay the current line.
.RE

.sp
.ne 2
.na
\fBtranspose-chars\fR
.ad
.RS 30n
Swap the character under the cursor with the character just before the cursor.
.RE

.sp
.ne 2
.na
\fBset-mark\fR
.ad
.RS 30n
Set a mark at the position of the cursor.
.RE

.sp
.ne 2
.na
\fBexchange-point-and-mark\fR
.ad
.RS 30n
Move the cursor to the last mark that was set, and move the mark to where the
cursor used to be.
.RE

.sp
.ne 2
.na
\fBkill-region\fR
.ad
.RS 30n
Delete the characters that lie between the last mark that was set, and the
cursor.
.RE

.sp
.ne 2
.na
\fBcopy-region-as-kill\fR
```

```
.ad
.RS 30n
Copy the text between the mark and the cursor to the cut buffer, without
deleting the original text.
.RE

.sp
.ne 2
.na
\fByank\fR
.ad
.RS 30n
Insert the text that was last deleted, just before the current position of the
cursor.
.RE

.sp
.ne 2
.na
\fBappend-yank\fR
.ad
.RS 30n
Paste the current contents of the cut buffer, after the cursor.
.RE

.sp
.ne 2
.na
\fBup-history\fR
.ad
.RS 30n
Recall the next oldest line that was entered. Note that in \fBvi\fR mode you
are left in command mode.
.RE

.sp
.ne 2
.na
\fBdown-history\fR
.ad
.RS 30n
Recall the next most recent line that was entered. If no history recall session
is currently active, the next line from a previous recall session is recalled.
Note that in vi mode you are left in command mode.
.RE

.sp
.ne 2
.na
\fBhistory-search-backward\fR
.ad
.RS 30n
Recall the next oldest line who’s prefix matches the string which currently
precedes the cursor (in \fBvi\fR command-mode the character under the cursor is
also included in the search string). Note that in \fBvi\fR mode you are left in
command mode.
.RE

.sp
.ne 2
.na
\fBhistory-search-forward\fR
.ad
.RS 30n
Recall the next newest line who’s prefix matches the string which currently
precedes the cursor (in \fBvi\fR command-mode the character under the cursor is
```

```
also included in the search string). Note that in \fBvi\fR mode you are left in
command mode.
.RE

.sp
.ne 2
.na
\fBhistory-re-search-backward\fR
.ad
.RS 30n
Recall the next oldest line who’s prefix matches that established by the last
invocation of either history-search-forward or history-search-backward.
.RE

.sp
.ne 2
.na
\fBhistory-re-search-forward\fR
.ad
.RS 30n
Recall the next newest line who’s prefix matches that established by the last
invocation of either history-search-forward or history-search-backward.
.RE

.sp
.ne 2
.na
\fBcomplete-word\fR
.ad
.RS 30n
Attempt to complete the incomplete word which precedes the cursor. Unless the
host program has customized word completion, file name completion is attempted.
In \fBvi\fR command mode the character under the cursor is also included in
the word being completed, and you are left in \fBvi\fR insert mode.
.RE

.sp
.ne 2
.na
\fBexpand-filename\fR
.ad
.RS 30n
Within the command line, expand wild cards, tilde expressions and dollar
expressions in the file name which immediately precedes the cursor. In \fBvi\fR
command mode the character under the cursor is also included in the file name
being expanded, and you are left in \fBvi\fR insert mode.
.RE

.sp
.ne 2
.na
\fBlist-glob\fR
.ad
.RS 30n
List any file names which match the wild-card, tilde and dollar expressions in
the file name which immediately precedes the cursor, then redraw the input line
unchanged.
.RE

.sp
.ne 2
.na
\fBlist-history\fR
.ad
.RS 30n
Display the contents of the history list for the current history group. If a
```

```
repeat count of \fB> 1\fR is specified, only that many of the most recent lines
are displayed. See the Entering Repeat Counts section.
.RE

.sp
.ne 2
.na
\fBread-from-file\fR
.ad
.RS 30n
Temporarily switch to reading input from the file who’s name precedes the
cursor.
.RE

.sp
.ne 2
.na
\fBread-init-files\fR
.ad
.RS 30n
Re-read \fBteclarc\fR configuration files.
.RE

.sp
.ne 2
.na
\fBbeginning-of-history\fR
.ad
.RS 30n
Move to the oldest line in the history list. Note that in \fBvi\fR mode you are
left in command mode.
.RE

.sp
.ne 2
.na
\fBend-of-history\fR
.ad
.RS 30n
Move to the newest line in the history list (ie. the current line). Note that
in \fBvi\fR mode this leaves you in command mode.
.RE

.sp
.ne 2
.na
\fBdigit-argument\fR
.ad
.RS 30n
Enter a repeat count for the next key binding function. For details, see the
Entering Repeat Counts section.
.RE

.sp
.ne 2
.na
\fBnewline\fR
.ad
.RS 30n
Terminate and return the current contents of the line, after appending a
newline character. The newline character is normally ’\en’, but will be the
first character of the key sequence that invoked the newline action, if this
happens to be a printable character. If the action was invoked by the ’\en’
newline character or the ’\er’ carriage return character, the line is appended
to the history buffer.
.RE
```

```
.sp
.ne 2
.na
\fBrepeat-history\fR
.ad
.RS 30n
Return the line that is being edited, then arrange for the next most recent
entry in the history buffer to be recalled when tecla is next called.
Repeatedly invoking this action causes successive historical input lines to be
re-executed. Note that this action is equivalent to the ’Operate’ action in
ksh.
.RE

.sp
.ne 2
.na
\fBring-bell\fR
.ad
.RS 30n
Ring the terminal bell, unless the bell has been silenced via the nobeep
configuration option (see The Tecla Configuration File section).
.RE

.sp
.ne 2
.na
\fBforward-copy-char\fR
.ad
.RS 30n
Copy the next character into the cut buffer (NB. use repeat counts to copy more
than one).
.RE

.sp
.ne 2
.na
\fBbackward-copy-char\fR
.ad
.RS 30n
Copy the previous character into the cut buffer.
.RE

.sp
.ne 2
.na
\fBforward-copy-word\fR
.ad
.RS 30n
Copy the next word into the cut buffer.
.RE

.sp
.ne 2
.na
\fBbackward-copy-word\fR
.ad
.RS 30n
Copy the previous word into the cut buffer.
.RE

.sp
.ne 2
.na
\fBforward-find-char\fR
.ad
```

```
.RS 30n
Move the cursor to the next occurrence of the next character that you type.
.RE

.sp
.ne 2
.na
\fBbackward-find-char\fR
.ad
.RS 30n
Move the cursor to the last occurrence of the next character that you type.
.RE

.sp
.ne 2
.na
\fBforward-to-char\fR
.ad
.RS 30n
Move the cursor to the character just before the next occurrence of the next
character that the user types.
.RE

.sp
.ne 2
.na
\fBbackward-to-char\fR
.ad
.RS 30n
Move the cursor to the character just after the last occurrence before the
cursor of the next character that the user types.
.RE

.sp
.ne 2
.na
\fBrepeat-find-char\fR
.ad
.RS 30n
Repeat the last backward-find-char, forward-find-char, backward-to-char or
forward-to-char.
.RE

.sp
.ne 2
.na
\fBinvert-refind-char\fR
.ad
.RS 30n
Repeat the last backward-find-char, forward-find-char, backward-to-char, or
forward-to-char in the opposite direction.
.RE

.sp
.ne 2
.na
\fBdelete-to-column\fR
.ad
.RS 30n
Delete the characters from the cursor up to the column that is specified by the
repeat count.
.RE

.sp
.ne 2
.na
```

```
\fBdelete-to-parenthesis\fR
.ad
.RS 30n
Delete the characters from the cursor up to and including the matching
parenthesis, or next close parenthesis.
.RE

.sp
.ne 2
.na
\fBforward-delete-find\fR
.ad
.RS 30n
Delete the characters from the cursor up to and including the following
occurrence of the next character typed.
.RE

.sp
.ne 2
.na
\fBbackward-delete-find\fR
.ad
.RS 30n
Delete the characters from the cursor up to and including the preceding
occurrence of the next character typed.
.RE

.sp
.ne 2
.na
\fBforward-delete-to\fR
.ad
.RS 30n
Delete the characters from the cursor up to, but not including, the following
occurrence of the next character typed.
.RE

.sp
.ne 2
.na
\fBbackward-delete-to\fR
.ad
.RS 30n
Delete the characters from the cursor up to, but not including, the preceding
occurrence of the next character typed.
.RE

.sp
.ne 2
.na
\fBdelete-refind\fR
.ad
.RS 30n
Repeat the last *-delete-find or *-delete-to action.
.RE

.sp
.ne 2
.na
\fBdelete-invert-refind\fR
.ad
.RS 30n
Repeat the last *-delete-find or *-delete-to action, in the opposite direction.
.RE

.sp
```

```
.ne 2
.na
\fBcopy-to-column\fR
.ad
.RS 30n
Copy the characters from the cursor up to the column that is specified by the
repeat count, into the cut buffer.
.RE

.sp
.ne 2
.na
\fBcopy-to-parenthesis\fR
.ad
.RS 30n
Copy the characters from the cursor up to and including the matching
parenthesis, or next close parenthesis, into the cut buffer.
.RE

.sp
.ne 2
.na
\fBforward-copy-find\fR
.ad
.RS 30n
Copy the characters from the cursor up to and including the following occurrence
of the next character typed, into the cut buffer.
.RE

.sp
.ne 2
.na
\fBbackward-copy-find\fR
.ad
.RS 30n
Copy the characters from the cursor up to and including the preceding occurrence
of the next character typed, into the cut buffer.
.RE

.sp
.ne 2
.na
\fBforward-copy-to\fR
.ad
.RS 30n
Copy the characters from the cursor up to, but not including, the following
occurrence of the next character typed, into the cut buffer.
.RE

.sp
.ne 2
.na
\fBbackward-copy-to\fR
.ad
.RS 30n
Copy the characters from the cursor up to, but not including, the preceding
occurrence of the next character typed, into the cut buffer.
.RE

.sp
.ne 2
.na
\fBcopy-refind\fR
.ad
.RS 30n
Repeat the last *-copy-find or *-copy-to action.
```

```
.RE

.sp
.ne 2
.na
\fBcopy-invert-refind\fR
.ad
.RS 30n
Repeat the last *-copy-find or *-copy-to action, in the opposite direction.
.RE

.sp
.ne 2
.na
\fBvi-mode\fR
.ad
.RS 30n
Switch to \fBvi\fR mode from emacs mode.
.RE

.sp
.ne 2
.na
\fBemacs-mode\fR
.ad
.RS 30n
Switch to \fBemacs\fR mode from \fBvi\fR mode.
.RE

.sp
.ne 2
.na
\fBvi-insert\fR
.ad
.RS 30n
From \fBvi\fR command mode, switch to insert mode.
.RE

.sp
.ne 2
.na
\fBvi-overwrite\fR
.ad
.RS 30n
From \fBvi\fR command mode, switch to overwrite mode.
.RE

.sp
.ne 2
.na
\fBvi-insert-at-bol\fR
.ad
.RS 30n
From \fBvi\fR command mode, move the cursor to the start of the line and switch
to insert mode.
.RE

.sp
.ne 2
.na
\fBvi-append-at-eol\fR
.ad
.RS 30n
From \fBvi\fR command mode, move the cursor to the end of the line and switch
to append mode.
.RE
```

```
.sp
.ne 2
.na
\fBvi-append\fR
.ad
.RS 30n
From \fBvi\fR command mode, move the cursor one position right, and switch to
insert mode.
.RE

.sp
.ne 2
.na
\fBvi-replace-char\fR
.ad
.RS 30n
From \fBvi\fR command mode, replace the character under the cursor with the
next character entered.
.RE

.sp
.ne 2
.na
\fBvi-forward-change-char\fR
.ad
.RS 30n
From \fBvi\fR command mode, delete the next character then enter insert mode.
.RE

.sp
.ne 2
.na
\fBvi-backward-change-char\fR
.ad
.RS 30n
From vi command mode, delete the preceding character then enter insert mode.
.RE

.sp
.ne 2
.na
\fBvi-forward-change-word\fR
.ad
.RS 30n
From \fBvi\fR command mode, delete the next word then enter insert mode.
.RE

.sp
.ne 2
.na
\fBvi-backward-change-word\fR
.ad
.RS 30n
From vi command mode, delete the preceding word then enter insert mode.
.RE

.sp
.ne 2
.na
\fBvi-change-rest-of-line\fR
.ad
.RS 30n
From \fBvi\fR command mode, delete from the cursor to the end of the line, then
enter insert mode.
.RE
```

```
.sp
.ne 2
.na
\fBvi-change-line\fR
.ad
.RS 30n
From \fBvi\fR command mode, delete the current line, then enter insert mode.
.RE

.sp
.ne 2
.na
\fBvi-change-to-bol\fR
.ad
.RS 30n
From \fBvi\fR command mode, delete all characters between the cursor and the
beginning of the line, then enter insert mode.
.RE

.sp
.ne 2
.na
\fBvi-change-to-column\fR
.ad
.RS 30n
From \fBvi\fR command mode, delete the characters from the cursor up to the
column that is specified by the repeat count, then enter insert mode.
.RE

.sp
.ne 2
.na
\fBvi-change-to-parenthesis\fR
.ad
.RS 30n
Delete the characters from the cursor up to and including the matching
parenthesis, or next close parenthesis, then enter \fBvi\fR insert mode.
.RE

.sp
.ne 2
.na
\fBvi-forward-change-find\fR
.ad
.RS 30n
From \fBvi\fR command mode, delete the characters from the cursor up to and
including the following occurrence of the next character typed, then enter
insert mode.
.RE

.sp
.ne 2
.na
\fBvi-backward-change-find\fR
.ad
.RS 30n
From vi command mode, delete the characters from the cursor up to and including
the preceding occurrence of the next character typed, then enter insert mode.
.RE

.sp
.ne 2
.na
\fBvi-forward-change-to\fR
.ad
```

```
.RS 30n
From \fBvi\fR command mode, delete the characters from the cursor up to, but
not including, the following occurrence of the next character typed, then enter
insert mode.
.RE

.sp
.ne 2
.na
\fBvi-backward-change-to\fR
.ad
.RS 30n
From \fBvi\fR command mode, delete the characters from the cursor up to, but
not including, the preceding occurrence of the next character typed, then enter
insert mode.
.RE

.sp
.ne 2
.na
\fBvi-change-refind\fR
.ad
.RS 30n
Repeat the last vi-*-change-find or vi-*-change-to action.
.RE

.sp
.ne 2
.na
\fBvi-change-invert-refind\fR
.ad
.RS 30n
Repeat the last vi-*-change-find or vi-*-change-to action, in the opposite
direction.
.RE

.sp
.ne 2
.na
\fBvi-undo\fR
.ad
.RS 30n
In \fBvi\fR mode, undo the last editing operation.
.RE

.sp
.ne 2
.na
\fBvi-repeat-change\fR
.ad
.RS 30n
In \fBvi\fR command mode, repeat the last command that modified the line.
.RE

.SS "Default Key Bindings In \fBemacs\fR Mode"
.LP
The following default key bindings, which can be overridden by the tecla
The following default key bindings, which can be overriden by the tecla
configuration file, are designed to mimic most of the bindings of the unix
\fBtcsh\fR shell, when it is in \fBemacs\fR editing mode.
\fBtcsh shell\fR shell, when it is in \fBemacs\fR editing mode.
.sp
.LP
This is the default editing mode of the tecla library.
.sp
.LP
```

```
Under UNIX the terminal driver sets a number of special keys for certain
functions. The tecla library attempts to use the same key bindings to maintain
consistency. The key sequences shown for the following 6 bindings are thus just
examples of what they will probably be set to. If you have used the stty
command to change these keys, then the default bindings should match.
.sp
.ne 2
.na
\fB\fB^C\fR\fR
.ad
.RS 6n
user-interrupt
.RE

.sp
.ne 2
.na
\fB^\e\fR
.ad
.RS 6n
abort
.RE

.sp
.ne 2
.na
\fB\fB^Z\fR\fR
.ad
.RS 6n
suspend
.RE

.sp
.ne 2
.na
\fB\fB^Q\fR\fR
.ad
.RS 6n
start-output
.RE

.sp
.ne 2
.na
\fB\fB^S\fR\fR
.ad
.RS 6n
stop-output
.RE

.sp
.ne 2
.na
\fB\fB^V\fR\fR
.ad
.RS 6n
literal-next
.RE

.sp
.LP
The cursor keys are referred to by name, as follows. This is necessary because
different types of terminals generate different key sequences when their cursor
keys are pressed.
.sp
.ne 2
```

```
.na
\fBright\fR
.ad
.RS 9n
cursor-right
.RE

.sp
.ne 2
.na
\fBleft\fR
.ad
.RS 9n
cursor-left
.RE

.sp
.ne 2
.na
\fBup\fR
.ad
.RS 9n
up-history
.RE

.sp
.ne 2
.na
\fBdown\fR
.ad
.RS 9n
down-history
.RE

.sp
.LP
The remaining bindings don’t depend on the terminal settings.
.sp
.ne 2
.na
\fB\fB^F\fR\fR
.ad
.RS 21n
cursor-right
.RE

.sp
.ne 2
.na
\fB\fB^B\fR\fR
.ad
.RS 21n
cursor-left
.RE

.sp
.ne 2
.na
\fB\fBM-i\fR\fR
.ad
.RS 21n
insert-mode
.RE

.sp
.ne 2
```

```
.na
\fB\fB^A\fR\fR
.ad
.RS 21n
beginning-of-line
.RE

.sp
.ne 2
.na
\fB\fB^E\fR\fR
.ad
.RS 21n
end-of-line
.RE

.sp
.ne 2
.na
\fB\fB^U\fR\fR
.ad
.RS 21n
delete-line
.RE

.sp
.ne 2
.na
\fB\fB^K\fR\fR
.ad
.RS 21n
kill-line
.RE

.sp
.ne 2
.na
\fB\fBM-f\fR\fR
.ad
.RS 21n
forward-word
.RE

.sp
.ne 2
.na
\fB\fBM-b\fR\fR
.ad
.RS 21n
backward-word
.RE

.sp
.ne 2
.na
\fB\fB^D\fR\fR
.ad
.RS 21n
del-char-or-list-or-eof
.RE

.sp
.ne 2
.na
\fB\fB^H\fR\fR
.ad
```

```
.RS 21n
backward-delete-char
.RE

.sp
.ne 2
.na
\fB\fB^?\fR\fR
.ad
.RS 21n
backward-delete-char
.RE

.sp
.ne 2
.na
\fB\fBM-d\fR\fR
.ad
.RS 21n
forward-delete-word
.RE

.sp
.ne 2
.na
\fB\fBM-^H\fR\fR
.ad
.RS 21n
backward-delete-word
.RE

.sp
.ne 2
.na
\fB\fBM-^?\fR\fR
.ad
.RS 21n
backward-delete-word
.RE

.sp
.ne 2
.na
\fB\fBM-u\fR\fR
.ad
.RS 21n
upcase-word
.RE

.sp
.ne 2
.na
\fB\fBM-l\fR\fR
.ad
.RS 21n
downcase-word
.RE

.sp
.ne 2
.na
\fB\fBM-c\fR\fR
.ad
.RS 21n
capitalize-word
.RE
```

```
.sp
.ne 2
.na
\fB\fB^R\fR\fR
.ad
.RS 21n
redisplay
.RE

.sp
.ne 2
.na
\fB\fB^L\fR\fR
.ad
.RS 21n
clear-screen
.RE

.sp
.ne 2
.na
\fB\fB^T\fR\fR
.ad
.RS 21n
transpose-chars
.RE

.sp
.ne 2
.na
\fB\fB^@\fR\fR
.ad
.RS 21n
set-mark
.RE

.sp
.ne 2
.na
\fB\fB^X^X\fR\fR
.ad
.RS 21n
exchange-point-and-mark
.RE

.sp
.ne 2
.na
\fB\fB^W\fR\fR
.ad
.RS 21n
kill-region
.RE

.sp
.ne 2
.na
\fB\fBM-w\fR\fR
.ad
.RS 21n
copy-region-as-kill
.RE

.sp
.ne 2
```

```
.na
\fB\fB^Y\fR\fR
.ad
.RS 21n
yank
.RE

.sp
.ne 2
.na
\fB\fB^P\fR\fR
.ad
.RS 21n
up-history
.RE

.sp
.ne 2
.na
\fB\fB^N\fR\fR
.ad
.RS 21n
down-history
.RE

.sp
.ne 2
.na
\fB\fBM-p\fR\fR
.ad
.RS 21n
history-search-backward
.RE

.sp
.ne 2
.na
\fB\fBM-n\fR\fR
.ad
.RS 21n
history-search-forward
.RE

.sp
.ne 2
.na
\fB\fB^I\fR\fR
.ad
.RS 21n
complete-word
.RE

.sp
.ne 2
.na
\fB\fB^X*\fR\fR
.ad
.RS 21n
expand-filename
.RE

.sp
.ne 2
.na
\fB\fB^X^F\fR\fR
.ad
```

```
.RS 21n
read-from-file
.RE

.sp
.ne 2
.na
\fB\fB^X^R\fR\fR
.ad
.RS 21n
read-init-files
.RE

.sp
.ne 2
.na
\fB\fB^Xg\fR\fR
.ad
.RS 21n
list-glob
.RE

.sp
.ne 2
.na
\fB\fB^Xh\fR\fR
.ad
.RS 21n
list-history
.RE

.sp
.ne 2
.na
\fB\fBM-<\fR\fR
.ad
.RS 21n
beginning-of-history
.RE

.sp
.ne 2
.na
\fB\fBM->\fR\fR
.ad
.RS 21n
end-of-history
.RE

.sp
.ne 2
.na
\fB\fB\en\fR\fR
.ad
.RS 21n
newline
.RE

.sp
.ne 2
.na
\fB\fB\er\fR\fR
.ad
.RS 21n
newline
.RE
```

```
.sp
.ne 2
.na
\fB\fBM-o\fR\fR
.ad
.RS 21n
repeat-history
.RE

.sp
.ne 2
.na
\fB\fBM-^V\fR\fR
.ad
.RS 21n
\fBvi\fR-mode
.RE

.sp
.ne 2
.na
\fB\fBM-0, M-1, ... M-9\fR\fR
.ad
.RS 21n
digit-argument (see below)
.RE

.sp
.LP
Note that \fB^I\fR is what the TAB key generates, and that \fB^@\fR can be
generated not only by pressing the CONTROL key and the @ key simultaneously,
but also by pressing the CONTROL key and the space bar at the same time.
.SS "Default Key Bindings in \fBvi\fR Mode"
.LP
The following default key bindings are designed to mimic the \fBvi\fR style of
editing as closely as possible. This means that very few editing functions are
provided in the initial character input mode, editing functions instead being
provided by the \fBvi\fR command mode. The \fBvi\fR command mode is entered
whenever the ESCAPE character is pressed, or whenever a key sequence that
starts with a meta character is entered. In addition to mimicing \fBvi\fR,
\fBlibtecla\fR provides bindings for tab completion, wild-card expansion of
file names, and historical line recall.
.sp
.LP
To learn how to tell the tecla library to use \fBvi\fR mode instead of the
default \fBemacs\fR editing mode, see the earlier section entitled The Tecla
Configuration File.
.sp
.LP
Under UNIX the terminal driver sets a number of special keys for certain
functions. The tecla library attempts to use the same key bindings to maintain
consistency, binding them both in input mode and in command mode. The key
sequences shown for the following 6 bindings are thus just examples of what
they will probably be set to. If you have used the \fBstty\fR command to change
these keys, then the default bindings should match.
.sp
.ne 2
.na
\fB\fB^C\fR\fR
.ad
.RS 8n
user-interrupt
.RE

.sp
```

```
.ne 2
.na
\fB^\e\fR
.ad
.RS 8n
abort
.RE

.sp
.ne 2
.na
\fB\fB^Z\fR\fR
.ad
.RS 8n
suspend
.RE

.sp
.ne 2
.na
\fB\fB^Q\fR\fR
.ad
.RS 8n
start-output
.RE

.sp
.ne 2
.na
\fB\fB^S\fR\fR
.ad
.RS 8n
stop-output
.RE

.sp
.ne 2
.na
\fB\fB^V\fR\fR
.ad
.RS 8n
literal-next
.RE

.sp
.ne 2
.na
\fB\fBM-^C\fR\fR
.ad
.RS 8n
user-interrupt
.RE

.sp
.ne 2
.na
\fBM-^\e\fR
.ad
.RS 8n
abort
.RE

.sp
.ne 2
.na
\fB\fBM-^Z\fR\fR
```

```
.ad
.RS 8n
suspend
.RE

.sp
.ne 2
.na
\fB\fBM-^Q\fR\fR
.ad
.RS 8n
start-output
.RE

.sp
.ne 2
.na
\fB\fBM-^S\fR\fR
.ad
.RS 8n
stop-output
.RE

.sp
.LP
Note that above, most of the bindings are defined twice, once as a raw control
code like \fB^C\fR and then a second time as a META character like \fBM-^C\fR.
The former is the binding for \fBvi\fR input mode, whereas the latter is the
binding for \fBvi\fR command mode. Once in command mode all key sequences that
the user types that they don’t explicitly start with an ESCAPE or a META key,
have their first key secretly converted to a META character before the key
sequence is looked up in the key binding table. Thus, once in command mode,
when you type the letter i, for example, the tecla library actually looks up
the binding for \fBM-i\fR.
.sp
.LP
The cursor keys are referred to by name, as follows. This is necessary because
different types of terminals generate different key sequences when their cursor
keys are pressed.
.sp
.ne 2
.na
\fB\fBright\fR\fR
.ad
.RS 9n
cursor-right
.RE

.sp
.ne 2
.na
\fB\fBleft\fR\fR
.ad
.RS 9n
cursor-left
.RE

.sp
.ne 2
.na
\fB\fBup\fR\fR
.ad
.RS 9n
up-history
.RE
```

```
.sp
.ne 2
.na
\fB\fBdown\fR\fR
.ad
.RS 9n
down-history
.RE

.sp
.LP
The cursor keys normally generate a key sequence that start with an ESCAPE
character, so beware that using the arrow keys will put you into command mode
(if you aren’t already in command mode).
.sp
.LP
The following are the terminal-independent key bindings for \fBvi\fR input
mode.
.sp
.ne 2
.na
\fB\fB^D\fR\fR
.ad
.RS 8n
list-or-eof
.RE

.sp
.ne 2
.na
\fB\fB^G\fR\fR
.ad
.RS 8n
list-glob
.RE

.sp
.ne 2
.na
\fB\fB^H\fR\fR
.ad
.RS 8n
backward-delete-char
.RE

.sp
.ne 2
.na
\fB\fB^I\fR\fR
.ad
.RS 8n
complete-word
.RE

.sp
.ne 2
.na
\fB\fB\er\fR\fR
.ad
.RS 8n
newline
.RE

.sp
.ne 2
.na
```

```
\fB\fB\en\fR\fR
.ad
.RS 8n
newline
.RE

.sp
.ne 2
.na
\fB\fB^L\fR\fR
.ad
.RS 8n
clear-screen
.RE

.sp
.ne 2
.na
\fB\fB^N\fR\fR
.ad
.RS 8n
down-history
.RE

.sp
.ne 2
.na
\fB\fB^P\fR\fR
.ad
.RS 8n
up-history
.RE

.sp
.ne 2
.na
\fB\fB^R\fR\fR
.ad
.RS 8n
redisplay
.RE

.sp
.ne 2
.na
\fB\fB^U\fR\fR
.ad
.RS 8n
backward-kill-line
.RE

.sp
.ne 2
.na
\fB\fB^W\fR\fR
.ad
.RS 8n
backward-delete-word
.RE

.sp
.ne 2
.na
\fB\fB^X*\fR\fR
.ad
.RS 8n
```

```
expand-filename
.RE

.sp
.ne 2
.na
\fB\fB^X^F\fR\fR
.ad
.RS 8n
read-from-file
.RE

.sp
.ne 2
.na
\fB\fB^X^R\fR\fR
.ad
.RS 8n
read-init-files
.RE

.sp
.ne 2
.na
\fB\fB^?\fR\fR
.ad
.RS 8n
backward-delete-char
.RE

.sp
.LP
The following are the key bindings that are defined in \fBvi\fR command mode,
this being specified by them all starting with a META character. As mentioned
above, once in command mode the initial meta character is optional. For
example, you might enter command mode by typing ESCAPE, and then press ’H’
twice to move the cursor two positions to the left. Both ’H’ characters get
quietly converted to \fBM-h\fR before being compared to the key binding table,
the first one because ESCAPE followed by a character is always converted to the
equivalent META character, and the second because command mode was already
active.
.sp
.ne 2
.na
\fBM-<space>\fR
.ad
.RS 21n
cursor-right (META-space)
.RE

.sp
.ne 2
.na
\fB\fBM-$\fR\fR
.ad
.RS 21n
end-of-line
.RE

.sp
.ne 2
.na
\fB\fBM-*\fR\fR
.ad
.RS 21n
expand-filename
```

```
.RE

.sp
.ne 2
.na
\fB\fBM-+\fR\fR
.ad
.RS 21n
down-history
.RE

.sp
.ne 2
.na
\fB\fBM--\fR\fR
.ad
.RS 21n
up-history
.RE

.sp
.ne 2
.na
\fB\fBM-<\fR\fR
.ad
.RS 21n
beginning-of-history
.RE

.sp
.ne 2
.na
\fB\fBM->\fR\fR
.ad
.RS 21n
end-of-history
.RE

.sp
.ne 2
.na
\fB\fBM-^\fR\fR
.ad
.RS 21n
beginning-of-line
.RE

.sp
.ne 2
.na
\fB\fBM-\fR\fR
.ad
.RS 21n
repeat-find-char
.RE

.sp
.ne 2
.na
\fB\fBM-,\fR\fR
.ad
.RS 21n
invert-refind-char
.RE

.sp
```

```
.ne 2
.na
\fB\fBM-|\fR\fR
.ad
.RS 21n
goto-column
.RE

.sp
.ne 2
.na
\fB\fBM-~\fR\fR
.ad
.RS 21n
change-case
.RE

.sp
.ne 2
.na
\fB\fBM-.\fR\fR
.ad
.RS 21n
vi-repeat-change
.RE

.sp
.ne 2
.na
\fB\fBM-%\fR\fR
.ad
.RS 21n
find-parenthesis
.RE

.sp
.ne 2
.na
\fB\fBM-a\fR\fR
.ad
.RS 21n
vi-append
.RE

.sp
.ne 2
.na
\fB\fBM-A\fR\fR
.ad
.RS 21n
vi-append-at-eol
.RE

.sp
.ne 2
.na
\fB\fBM-b\fR\fR
.ad
.RS 21n
backward-word
.RE

.sp
.ne 2
.na
\fB\fBM-B\fR\fR
```

```
.ad
.RS 21n
backward-word
.RE

.sp
.ne 2
.na
\fB\fBM-C\fR\fR
.ad
.RS 21n
vi-change-rest-of-line
.RE

.sp
.ne 2
.na
\fB\fBM-cb\fR\fR
.ad
.RS 21n
vi-backward-change-word
.RE

.sp
.ne 2
.na
\fB\fBM-cB\fR\fR
.ad
.RS 21n
vi-backward-change-word
.RE

.sp
.ne 2
.na
\fB\fBM-cc\fR\fR
.ad
.RS 21n
vi-change-line
.RE

.sp
.ne 2
.na
\fB\fBM-ce\fR\fR
.ad
.RS 21n
vi-forward-change-word
.RE

.sp
.ne 2
.na
\fB\fBM-cE\fR\fR
.ad
.RS 21n
vi-forward-change-word
.RE

.sp
.ne 2
.na
\fB\fBM-cw\fR\fR
.ad
.RS 21n
vi-forward-change-word
```

```
.RE

.sp
.ne 2
.na
\fB\fBM-cW\fR\fR
.ad
.RS 21n
vi-forward-change-word
.RE

.sp
.ne 2
.na
\fB\fBM-cF\fR\fR
.ad
.RS 21n
vi-backward-change-find
.RE

.sp
.ne 2
.na
\fB\fBM-cf\fR\fR
.ad
.RS 21n
vi-forward-change-find
.RE

.sp
.ne 2
.na
\fB\fBM-cT\fR\fR
.ad
.RS 21n
vi-backward-change-to
.RE

.sp
.ne 2
.na
\fB\fBM-ct\fR\fR
.ad
.RS 21n
vi-forward-change-to
.RE

.sp
.ne 2
.na
\fB\fBM-c;\fR\fR
.ad
.RS 21n
vi-change-refind
.RE

.sp
.ne 2
.na
\fB\fBM-c,\fR\fR
.ad
.RS 21n
vi-change-invert-refind
.RE

.sp
```

```
.ne 2
.na
\fB\fBM-ch\fR\fR
.ad
.RS 21n
vi-backward-change-char
.RE

.sp
.ne 2
.na
\fB\fBM-c^H\fR\fR
.ad
.RS 21n
vi-backward-change-char
.RE

.sp
.ne 2
.na
\fB\fBM-c^?\fR\fR
.ad
.RS 21n
vi-backward-change-char
.RE

.sp
.ne 2
.na
\fB\fBM-cl\fR\fR
.ad
.RS 21n
vi-forward-change-char
.RE

.sp
.ne 2
.na
\fBM-c<space>\fR
.ad
.RS 21n
vi-forward-change-char (META-c-space)
.RE

.sp
.ne 2
.na
\fB\fBM-c^\fR\fR
.ad
.RS 21n
vi-change-to-bol
.RE

.sp
.ne 2
.na
\fB\fBM-c0\fR\fR
.ad
.RS 21n
vi-change-to-bol
.RE

.sp
.ne 2
.na
\fB\fBM-c$\fR\fR
```

```
.ad
.RS 21n
vi-change-rest-of-line
.RE

.sp
.ne 2
.na
\fB\fBM-c|\fR\fR
.ad
.RS 21n
vi-change-to-column
.RE

.sp
.ne 2
.na
\fB\fBM-c%\fR\fR
.ad
.RS 21n
vi-change-to-parenthesis
.RE

.sp
.ne 2
.na
\fB\fBM-dh\fR\fR
.ad
.RS 21n
backward-delete-char
.RE

.sp
.ne 2
.na
\fB\fBM-d^H\fR\fR
.ad
.RS 21n
backward-delete-char
.RE

.sp
.ne 2
.na
\fB\fBM-d^?\fR\fR
.ad
.RS 21n
backward-delete-char
.RE

.sp
.ne 2
.na
\fB\fBM-dl\fR\fR
.ad
.RS 21n
forward-delete-char
.RE

.sp
.ne 2
.na
\fBM-d<space>\fR
.ad
.RS 21n
forward-delete-char (META-d-space)
```

```
.RE

.sp
.ne 2
.na
\fB\fBM-dd\fR\fR
.ad
.RS 21n
delete-line
.RE

.sp
.ne 2
.na
\fB\fBM-db\fR\fR
.ad
.RS 21n
backward-delete-word
.RE

.sp
.ne 2
.na
\fB\fBM-dB\fR\fR
.ad
.RS 21n
backward-delete-word
.RE

.sp
.ne 2
.na
\fB\fBM-de\fR\fR
.ad
.RS 21n
forward-delete-word
.RE

.sp
.ne 2
.na
\fB\fBM-dE\fR\fR
.ad
.RS 21n
forward-delete-word
.RE

.sp
.ne 2
.na
\fB\fBM-dw\fR\fR
.ad
.RS 21n
forward-delete-word
.RE

.sp
.ne 2
.na
\fB\fBM-dW\fR\fR
.ad
.RS 21n
forward-delete-word
.RE

.sp
```

```
.ne 2
.na
\fB\fBM-dF\fR\fR
.ad
.RS 21n
backward-delete-find
.RE

.sp
.ne 2
.na
\fB\fBM-df\fR\fR
.ad
.RS 21n
forward-delete-find
.RE

.sp
.ne 2
.na
\fB\fBM-dT\fR\fR
.ad
.RS 21n
backward-delete-to
.RE

.sp
.ne 2
.na
\fB\fBM-dt\fR\fR
.ad
.RS 21n
forward-delete-to
.RE

.sp
.ne 2
.na
\fB\fBM-d;\fR\fR
.ad
.RS 21n
delete-refind
.RE

.sp
.ne 2
.na
\fB\fBM-d,\fR\fR
.ad
.RS 21n
delete-invert-refind
.RE

.sp
.ne 2
.na
\fB\fBM-d^\fR\fR
.ad
.RS 21n
backward-kill-line
.RE

.sp
.ne 2
.na
\fB\fBM-d0\fR\fR
```

```
.ad
.RS 21n
backward-kill-line
.RE

.sp
.ne 2
.na
\fB\fBM-d$\fR\fR
.ad
.RS 21n
kill-line
.RE

.sp
.ne 2
.na
\fB\fBM-D\fR\fR
.ad
.RS 21n
kill-line
.RE

.sp
.ne 2
.na
\fB\fBM-d|\fR\fR
.ad
.RS 21n
delete-to-column
.RE

.sp
.ne 2
.na
\fB\fBM-d%\fR\fR
.ad
.RS 21n
delete-to-parenthesis
.RE

.sp
.ne 2
.na
\fB\fBM-e\fR\fR
.ad
.RS 21n
forward-word
.RE

.sp
.ne 2
.na
\fB\fBM-E\fR\fR
.ad
.RS 21n
forward-word
.RE

.sp
.ne 2
.na
\fB\fBM-f\fR\fR
.ad
.RS 21n
forward-find-char
```

```
.RE

.sp
.ne 2
.na
\fB\fBM-F\fR\fR
.ad
.RS 21n
backward-find-char
.RE

.sp
.ne 2
.na
\fB\fBM--\fR\fR
.ad
.RS 21n
up-history
.RE

.sp
.ne 2
.na
\fB\fBM-h\fR\fR
.ad
.RS 21n
cursor-left
.RE

.sp
.ne 2
.na
\fB\fBM-H\fR\fR
.ad
.RS 21n
beginning-of-history
.RE

.sp
.ne 2
.na
\fB\fBM-i\fR\fR
.ad
.RS 21n
vi-insert
.RE

.sp
.ne 2
.na
\fB\fBM-I\fR\fR
.ad
.RS 21n
vi-insert-at-bol
.RE

.sp
.ne 2
.na
\fB\fBM-j\fR\fR
.ad
.RS 21n
down-history
.RE

.sp
```

```
.ne 2
.na
\fB\fBM-J\fR\fR
.ad
.RS 21n
history-search-forward
.RE

.sp
.ne 2
.na
\fB\fBM-k\fR\fR
.ad
.RS 21n
up-history
.RE

.sp
.ne 2
.na
\fB\fBM-K\fR\fR
.ad
.RS 21n
history-search-backward
.RE

.sp
.ne 2
.na
\fB\fBM-l\fR\fR
.ad
.RS 21n
cursor-right
.RE

.sp
.ne 2
.na
\fB\fBM-L\fR\fR
.ad
.RS 21n
end-of-history
.RE

.sp
.ne 2
.na
\fB\fBM-n\fR\fR
.ad
.RS 21n
history-re-search-forward
.RE

.sp
.ne 2
.na
\fB\fBM-N\fR\fR
.ad
.RS 21n
history-re-search-backward
.RE

.sp
.ne 2
.na
\fB\fBM-p\fR\fR
```

```
.ad
.RS 21n
append-yank
.RE

.sp
.ne 2
.na
\fB\fBM-P\fR\fR
.ad
.RS 21n
yank
.RE

.sp
.ne 2
.na
\fB\fBM-r\fR\fR
.ad
.RS 21n
vi-replace-char
.RE

.sp
.ne 2
.na
\fB\fBM-R\fR\fR
.ad
.RS 21n
vi-overwrite
.RE

.sp
.ne 2
.na
\fB\fBM-s\fR\fR
.ad
.RS 21n
vi-forward-change-char
.RE

.sp
.ne 2
.na
\fB\fBM-S\fR\fR
.ad
.RS 21n
vi-change-line
.RE

.sp
.ne 2
.na
\fB\fBM-t\fR\fR
.ad
.RS 21n
forward-to-char
.RE

.sp
.ne 2
.na
\fB\fBM-T\fR\fR
.ad
.RS 21n
backward-to-char
```

```
.RE

.sp
.ne 2
.na
\fB\fBM-u\fR\fR
.ad
.RS 21n
vi-undo
.RE

.sp
.ne 2
.na
\fB\fBM-w\fR\fR
.ad
.RS 21n
forward-to-word
.RE

.sp
.ne 2
.na
\fB\fBM-W\fR\fR
.ad
.RS 21n
forward-to-word
.RE

.sp
.ne 2
.na
\fB\fBM-x\fR\fR
.ad
.RS 21n
forward-delete-char
.RE

.sp
.ne 2
.na
\fB\fBM-X\fR\fR
.ad
.RS 21n
backward-delete-char
.RE

.sp
.ne 2
.na
\fB\fBM-yh\fR\fR
.ad
.RS 21n
backward-copy-char
.RE

.sp
.ne 2
.na
\fB\fBM-y^H\fR\fR
.ad
.RS 21n
backward-copy-char
.RE

.sp
```

```
.ne 2
.na
\fB\fBM-y^?\fR\fR
.ad
.RS 21n
backward-copy-char
.RE

.sp
.ne 2
.na
\fB\fBM-yl\fR\fR
.ad
.RS 21n
forward-copy-char
.RE

.sp
.ne 2
.na
\fBM-y<space>\fR
.ad
.RS 21n
forward-copy-char (META-y-space)
.RE

.sp
.ne 2
.na
\fB\fBM-ye\fR\fR
.ad
.RS 21n
forward-copy-word
.RE

.sp
.ne 2
.na
\fB\fBM-yE\fR\fR
.ad
.RS 21n
forward-copy-word
.RE

.sp
.ne 2
.na
\fB\fBM-yw\fR\fR
.ad
.RS 21n
forward-copy-word
.RE

.sp
.ne 2
.na
\fB\fBM-yW\fR\fR
.ad
.RS 21n
forward-copy-word
.RE

.sp
.ne 2
.na
\fB\fBM-yb\fR\fR
```

```
.ad
.RS 21n
backward-copy-word
.RE

.sp
.ne 2
.na
\fB\fBM-yB\fR\fR
.ad
.RS 21n
backward-copy-word
.RE

.sp
.ne 2
.na
\fB\fBM-yf\fR\fR
.ad
.RS 21n
forward-copy-find
.RE

.sp
.ne 2
.na
\fB\fBM-yF\fR\fR
.ad
.RS 21n
backward-copy-find
.RE

.sp
.ne 2
.na
\fB\fBM-yt\fR\fR
.ad
.RS 21n
forward-copy-to
.RE

.sp
.ne 2
.na
\fB\fBM-yT\fR\fR
.ad
.RS 21n
backward-copy-to
.RE

.sp
.ne 2
.na
\fB\fBM-y;\fR\fR
.ad
.RS 21n
copy-refind
.RE

.sp
.ne 2
.na
\fB\fBM-y,\fR\fR
.ad
.RS 21n
copy-invert-refind
```

```
.RE

.sp
.ne 2
.na
\fB\fBM-y^\fR\fR
.ad
.RS 21n
copy-to-bol
.RE

.sp
.ne 2
.na
\fB\fBM-y0\fR\fR
.ad
.RS 21n
copy-to-bol
.RE

.sp
.ne 2
.na
\fB\fBM-y$\fR\fR
.ad
.RS 21n
copy-rest-of-line
.RE

.sp
.ne 2
.na
\fB\fBM-yy\fR\fR
.ad
.RS 21n
copy-line
.RE

.sp
.ne 2
.na
\fB\fBM-Y\fR\fR
.ad
.RS 21n
copy-line
.RE

.sp
.ne 2
.na
\fB\fBM-y|\fR\fR
.ad
.RS 21n
copy-to-column
.RE

.sp
.ne 2
.na
\fB\fBM-y%\fR\fR
.ad
.RS 21n
copy-to-parenthesis
.RE

.sp
```

```
.ne 2
.na
\fB\fBM-^E\fR\fR
.ad
.RS 21n
emacs-mode
.RE

.sp
.ne 2
.na
\fB\fBM-^H\fR\fR
.ad
.RS 21n
cursor-left
.RE

.sp
.ne 2
.na
\fB\fBM-^?\fR\fR
.ad
.RS 21n
cursor-left
.RE

.sp
.ne 2
.na
\fB\fBM-^L\fR\fR
.ad
.RS 21n
clear-screen
.RE

.sp
.ne 2
.na
\fB\fBM-^N\fR\fR
.ad
.RS 21n
down-history
.RE

.sp
.ne 2
.na
\fB\fBM-^P\fR\fR
.ad
.RS 21n
up-history
.RE

.sp
.ne 2
.na
\fB\fBM-^R\fR\fR
.ad
.RS 21n
redisplay
.RE

.sp
.ne 2
.na
\fB\fBM-^D\fR\fR
```

```
.ad
.RS 21n
list-or-eof
.RE

.sp
.ne 2
.na
\fB\fBM-^I\fR\fR
.ad
.RS 21n
complete-word
.RE

.sp
.ne 2
.na
\fBM-\er\fR
.ad
.RS 21n
newline
.RE

.sp
.ne 2
.na
\fB\fBM-\en\fR\fR
.ad
.RS 21n
newline
.RE

.sp
.ne 2
.na
\fB\fBM-^X^R\fR\fR
.ad
.RS 21n
read-init-files
.RE

.sp
.ne 2
.na
\fB\fBM-^Xh\fR\fR
.ad
.RS 21n
list-history
.RE

.sp
.ne 2
.na
\fB\fBM-0, M-1, ... M-9\fR\fR
.ad
.RS 21n
digit-argument (see below)
.RE

.sp
.LP
Note that \fB^I\fR is what the TAB key generates.
.SS "Entering Repeat Counts"
.LP
Many of the key binding functions described previously, take an optional count,
typed in before the target key sequence. This is interpreted as a repeat count
```

```
by most bindings. A notable exception is the goto-column binding, which
interprets the count as a column number.
.sp
.LP
By default you can specify this count argument by pressing the META key while
typing in the numeric count. This relies on the digit-argument action being
bound to 'META-0', 'META-1' etc. Once any one of these bindings has been
activated, you can optionally take your finger off the META key to type in the
rest of the number, since every numeric digit thereafter is treated as part of
the number, unless it is preceded by the literal-next binding. As soon as a
non-digit, or literal digit key is pressed the repeat count is terminated and
either causes the just typed character to be added to the line that many times,
or causes the next key binding function to be given that argument.
.sp
.LP
For example, in \fBemacs\fR mode, typing:
.sp
.in +2
.nf
M-12a
.fi
.in -2

.sp
.LP
causes the letter 'a' to be added to the line 12 times, whereas
.sp
.in +2
.nf
M-4M-c
.fi
.in -2

.sp
.LP
Capitalizes the next 4 words.
.sp
.LP
In \fBvi\fR command mode the meta modifier is automatically added to all
characters typed in, so to enter a count in \fBvi\fR command-mode, just
involves typing in the number, just as it does in the \fBvi\fR editor itself.
So for example, in vi command mode, typing:
.sp
.in +2
.nf
4w2x
.fi
.in -2

.sp
.LP
moves the cursor four words to the right, then deletes two characters.
.sp
.LP
You can also bind digit-argument to other key sequences. If these end in a
numeric digit, that digit gets appended to the current repeat count. If it
doesn't end in a numeric digit, a new repeat count is started with a value of
zero, and can be completed by typing in the number, after letting go of the key
which triggered the digit-argument action.
.SH FILES
.ne 2
.na
\fB\fB/usr/lib/libtecla.so\fR\fR
.ad
.RS 27n
The tecla library
```

```
.RE

.sp
.ne 2
.na
\fB\fB/usr/include/libtecla.h\fR\fR
.ad
.RS 27n
The tecla header file
.RE

.sp
.ne 2
.na
\fB\fB~/.teclarc\fR\fR
.ad
.RS 27n
The personal tecla customization file
.RE

.SH ATTRIBUTES
.LP
See \fBattributes\fR(5) for descriptions of the following attributes:
.sp

.sp
.TS
box;
c | c
l | l .
ATTRIBUTE TYPE  ATTRIBUTE VALUE
_
Interface Stability     Evolving
.TE

.SH SEE ALSO
.LP
\fBvi\fR(1), \fBcpl_complete_word\fR(3TECLA), \fBef_expand_file\fR(3TECLA),
\fBgl_get_line\fR(3TECLA), \fBgl_io_mode\fR(3TECLA), \fBlibtecla\fR(3LIB),
\fBpca_lookup_file\fR(3TECLA), \fBattributes\fR(5)
```

```
**********************************************************
    7843 Tue Dec 11 09:48:10 2018
new/usr/src/tools/onbld/Checks/SpellCheck.py
10057 Man page misspellings ouput particuliar overriden
Reviewed by: Gerg¯  Mih^¡ly Doma <domag02@gmail.com>
**********************************************************
#
# CDDL HEADER START
#
# The contents of this file are subject to the terms of the
# Common Development and Distribution License (the "License").
# You may not use this file except in compliance with the License.
#
# You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
# or http://www.opensolaris.org/os/licensing.
# See the License for the specific language governing permissions
# and limitations under the License.
#
# When distributing Covered Code, include this CDDL HEADER in each
# file and include the License file at usr/src/OPENSOLARIS.LICENSE.
# If applicable, add the following below this CDDL HEADER, with the
# fields enclosed by brackets "[]" replaced with your own identifying
# information: Portions Copyright [yyyy] [name of copyright owner]
#
# CDDL HEADER END
#

#
# Copyright 2016 Joyent, Inc.
#

import re, sys

spellMsg = ’%s: Line %d contains "%s", a common misspelling of "%s"\n’
altMsg = ’%s: Line %d contains "%s"; please use "%s" instead for consistency wit

misspellings = {
        ’absense’: ’absence’,
        ’accessable’: ’accessible’,
        ’accomodate’: ’accommodate’,
        ’accomodation’: ’accommodation’,
        ’accross’: ’across’,
        ’acheive’: ’achieve’,
        ’addional’: ’additional’,
        ’addres’: ’address’,
        ’admininistrative’: ’administrative’,
        ’adminstered’: ’administered’,
        ’adminstrate’: ’administrate’,
        ’adminstration’: ’administration’,
        ’adminstrative’: ’administrative’,
        ’adminstrator’: ’administrator’,
        ’admissability’: ’admissibility’,
        ’adress’: ’address’,
        ’adressable’: ’addressable’,
        ’adressed’: ’addressed’,
        ’adressing’: ’addressing, dressing’,
        ’aginst’: ’against’,
        ’agression’: ’aggression’,
        ’agressive’: ’aggressive’,
        ’alot’: ’a lot, allot’,
        ’and and’: ’and’,
        ’apparantly’: ’apparently’,
        ’appearence’: ’appearance’,
        ’arguement’: ’argument’,
        ’assasination’: ’assassination’,
        ’auxilliary’: ’auxiliary’,
```

```
        ’basicly’: ’basically’,
        ’begining’: ’beginning’,
        ’belive’: ’believe’,
        ’beteen’: ’between’,
        ’betwen’: ’between’,
        ’beween’: ’between’,
        ’bewteen’: ’between’,
        ’bizzare’: ’bizarre’,
        ’buisness’: ’business’,
        ’calender’: ’calendar’,
        ’cemetary’: ’cemetery’,
        ’chauffer’: ’chauffeur’,
        ’collegue’: ’colleague’,
        ’comming’: ’coming’,
        ’commited’: ’committed’,
        ’commitee’: ’committee’,
        ’commiting’: ’committing’,
        ’comparision’: ’comparison’,
        ’comparisions’: ’comparisons’,
        ’compatability’: ’compatibility’,
        ’compatable’: ’compatible’,
        ’compatablity’: ’compatibility’,
        ’compatiable’: ’compatible’,
        ’compatiblity’: ’compatibility’,
        ’completly’: ’completely’,
        ’concious’: ’conscious’,
        ’condidtion’: ’condition’,
        ’conected’: ’connected’,
        ’conjuction’: ’conjunction’,
        ’continous’: ’continuous’,
        ’curiousity’: ’curiosity’,
        ’deamon’: ’daemon’,
        ’definately’: ’definitely’,
        ’desireable’: ’desirable’,
        ’diffrent’: ’different’,
        ’dilemna’: ’dilemma’,
        ’dissapear’: ’disappear’,
        ’dissapoint’: ’disappoint’,
        ’ecstacy’: ’ecstasy’,
        ’embarass’: ’embarrass’,
        ’enviroment’: ’environment’,
        ’exept’: ’except’,
        ’existance’: ’existence’,
        ’familar’: ’familiar’,
        ’finaly’: ’finally’,
        ’folowing’: ’following’,
        ’foriegn’: ’foreign’,
        ’forseeable’: ’foreseeable’,
        ’fourty’: ’forty’,
        ’foward’: ’forward’,
        ’freind’: ’friend’,
        ’futher’: ’further’,
        ’gaurd’: ’guard’,
        ’glamourous’: ’glamorous’,
        ’goverment’: ’government’,
        ’happend’: ’happened’,
        ’harrassment’: ’harassment’,
        ’hierachical’: ’hierarchical’,
        ’hierachies’: ’hierarchies’,
        ’hierachy’: ’hierarchy’,
        ’hierarcical’: ’hierarchical’,
        ’hierarcy’: ’hierarchy’,
        ’honourary’: ’honorary’,
        ’humourous’: ’humorous’,
        ’idiosyncracy’: ’idiosyncrasy’,
        ’immediatly’: ’immediately’,
        ’inaccessable’: ’inaccessible’,
        ’inbetween’: ’between’,
        ’incidently’: ’incidentally’,
        ’independant’: ’independent’,
        ’infomation’: ’information’,
        ’interupt’: ’interrupt’,
        ’intial’: ’initial’,
        ’intially’: ’initially’,
        ’irresistable’: ’irresistible’,
        ’jist’: ’gist’,
        ’knowlege’: ’knowledge’,
        ’lenght’: ’length’,
        ’liase’: ’liaise’,
        ’liason’: ’liaison’,
        ’libary’: ’library’,
        ’maching’: ’machine, marching, matching’,
        ’millenia’: ’millennia’,
        ’millenium’: ’millennium’,
        ’neccessary’: ’necessary’,
        ’negotation’: ’negotiation’,
        ’nontheless’: ’nonetheless’,
        ’noticable’: ’noticeable’,
        ’occassion’: ’occasion’,
        ’occassional’: ’occasional’,
        ’occassionally’: ’occasionally’,
        ’occurance’: ’occurrence’,
        ’occured’: ’occurred’,
        ’occurence’: ’occurrence’,
        ’occuring’: ’occurring’,
        ’ommision’: ’omission’,
        ’orginal’: ’original’,
        ’orginally’: ’originally’,
        ’ouput’: ’output’,
        ’overriden’: ’overridden’,
        ’particuliar’: ’particular’,
        ’pavillion’: ’pavilion’,
        ’peice’: ’piece’,
        ’persistant’: ’persistent’,
        ’politican’: ’politician’,
        ’posession’: ’possession’,
        ’possiblity’: ’possibility’,
        ’preceed’: ’precede’,
        ’preceeded’: ’preceded’,
        ’preceeding’: ’preceding’,
        ’preceeds’: ’precedes’,
        ’prefered’: ’preferred’,
        ’prefering’: ’preferring’,
        ’presense’: ’presence’,
        ’proces’: ’process’,
        ’propoganda’: ’propaganda’,
        ’psuedo’: ’pseudo’,
        ’publically’: ’publicly’,
        ’realy’: ’really’,
        ’reciept’: ’receipt’,
        ’recieve’: ’receive’,
        ’recieved’: ’received’,
        ’reciever’: ’receiver’,
        ’recievers’: ’receivers’,
        ’recieves’: ’receives’,
        ’recieving’: ’receiving’,
        ’recomend’: ’recommend’,
        ’recomended’: ’recommended’,
        ’recomending’: ’recommending’,
        ’recomends’: ’recommends’,
        ’recurse’: ’recur’,
        ’recurses’: ’recurs’,
```

```
        ’recursing’: ’recurring’,
        ’refered’: ’referred’,
        ’refering’: ’referring’,
        ’religous’: ’religious’,
        ’rember’: ’remember’,
        ’remeber’: ’remember’,
        ’repetion’: ’repetition’,
        ’reponsible’: ’responsible’,
        ’resistence’: ’resistance’,
        ’retreive’: ’retrieve’,
        ’seige’: ’siege’,
        ’sence’: ’since’,
        ’seperate’: ’separate’,
        ’seperated’: ’separated’,
        ’seperately’: ’separately’,
        ’seperates’: ’separates’,
        ’similiar’: ’similar’,
        ’somwhere’: ’somewhere’,
        ’sould’: ’could, should, sold, soul’,
        ’sturcture’: ’structure’,
        ’succesful’: ’successful’,
        ’succesfully’: ’successfully’,
        ’successfull’: ’successful’,
        ’sucessful’: ’successful’,
        ’supercede’: ’supersede’,
        ’supress’: ’suppress’,
        ’supressed’: ’suppressed’,
        ’suprise’: ’surprise’,
        ’suprisingly’: ’surprisingly’,
        ’sytem’: ’system’,
        ’tendancy’: ’tendency’,
        ’the the’: ’the’,
        ’the these’: ’these’,
        ’therefor’: ’therefore’,
        ’threshhold’: ’threshold’,
        ’tolerence’: ’tolerance’,
        ’tommorow’: ’tomorrow’,
        ’tommorrow’: ’tomorrow’,
        ’tounge’: ’tongue’,
        ’tranformed’: ’transformed’,
        ’transfered’: ’transferred’,
        ’truely’: ’truly’,
        ’trustworthyness’: ’trustworthiness’,
        ’uncommited’: ’uncommitted’,
        ’unforseen’: ’unforeseen’,
        ’unfortunatly’: ’unfortunately’,
        ’unsuccessfull’: ’unsuccessful’,
        ’untill’: ’until’,
        ’upto’: ’up to’,
        ’whereever’: ’wherever’,
        ’wich’: ’which’,
        ’wierd’: ’weird’,
        ’wtih’: ’with’,
}
```

______***unchanged_portion_omitted_***