```
**********************************************************
   12797 Tue Sep 10 18:35:17 2013
new/usr/src/man/man1/ar.1
4023 - Typo in file(1) manpage and various others
**********************************************************
   1 '\" te
   2 .\"  Copyright 1989 AT&T
   3 .\"  Portions Copyright (c) 1992, X/Open Company Limited  All Rights Reserved
   4 .\" Copyright (c) 2009, Sun Microsystems, Inc.  All Rights Reserved
   5 .\" Sun Microsystems, Inc. gratefully acknowledges The Open Group for permission
   6 .\" The Institute of Electrical and Electronics Engineers and The Open Group, ha
   7 .\" are reprinted and reproduced in electronic form in the Sun OS Reference Manu
   8 .\" and Electronics Engineers, Inc and The Open Group. In the event of any discr
   9 .\"  This notice shall appear on any product containing this material.
  10 .\" The contents of this file are subject to the terms of the Common Development
  11 .\"  See the License for the specific language governing permissions and limitat
  12 .\" the fields enclosed by brackets "[]" replaced with your own identifying info
  13 .TH AR 1 "Sep 10, 2013"
  13 .TH AR 1 "Aug 24, 2009"
  14 .SH NAME
  15 ar \- maintain portable archive or library
  16 .SH SYNOPSIS
  17 .LP
  18 .nf
  19 \fB/usr/bin/ar\fR \fB-d\fR [\fB-Vv\fR] \fIarchive\fR \fIfile\fR...
  20 .fi

  22 .LP
  23 .nf
  24 \fB/usr/bin/ar\fR \fB-m\fR [\fB-abiVv\fR] [\fIposname\fR] \fIarchive\fR \fIfile\
  25 .fi

  27 .LP
  28 .nf
  29 \fB/usr/bin/ar\fR \fB-p\fR [\fB-sVv\fR] \fIarchive\fR [\fIfile\fR]...
  30 .fi

  32 .LP
  33 .nf
  34 \fB/usr/bin/ar\fR \fB-q\fR [\fB-cVv\fR] \fIarchive\fR \fIfile\fR...
  35 .fi

  37 .LP
  38 .nf
  39 \fB/usr/bin/ar\fR \fB-r\fR [\fB-abciuVv\fR] [\fIposname\fR] \fIarchive\fR \fIfil
  40 .fi

  42 .LP
  43 .nf
  44 \fB/usr/bin/ar\fR \fB-t\fR [\fB-sVv\fR] \fIarchive\fR [\fIfile\fR]...
  45 .fi

  47 .LP
  48 .nf
  49 \fB/usr/bin/ar\fR \fB-x\fR [\fB-CsTVv\fR] \fIarchive\fR [\fIfile\fR]...
  50 .fi

  52 .LP
  53 .nf
  54 \fB/usr/xpg4/bin/ar\fR \fB-d\fR [\fB-Vv\fR] \fIarchive\fR \fIfile\fR...
  55 .fi

  57 .LP
  58 .nf
  59 \fB/usr/xpg4/bin/ar\fR \fB-m\fR [\fB-abiVv\fR] [\fIposname\fR] \fIarchive\fR \fI
  60 .fi
```



```
  62 .LP
  63 .nf
  64 \fB/usr/xpg4/bin/ar\fR \fB-p\fR [\fB-sVv\fR] \fIarchive\fR [\fIfile\fR]...
  65 .fi

  67 .LP
  68 .nf
  69 \fB/usr/xpg4/bin/ar\fR \fB-q\fR [\fB-cVv\fR] \fIarchive\fR \fIfile\fR...
  70 .fi

  72 .LP
  73 .nf
  74 \fB/usr/xpg4/bin/ar\fR \fB-r\fR [\fB-abciuVv\fR] [\fIposname\fR] \fIarchive\fR \
  75 .fi

  77 .LP
  78 .nf
  79 \fB/usr/xpg4/bin/ar\fR \fB-t\fR [\fB-sVv\fR] \fIarchive\fR [\fIfile\fR]...
  80 .fi

  82 .LP
  83 .nf
  84 \fB/usr/xpg4/bin/ar\fR \fB-x\fR [\fB-CsTVv\fR] \fIarchive\fR [\fIfile\fR]...
  85 .fi

  87 .SH DESCRIPTION
  88 .sp
  89 .LP
  90 The \fBar\fR utility maintains groups of files combined into a single archive
  91 file. Its main use is to create and update library files. However, it can be
  92 used for any similar purpose. The magic string and the file headers used by
  93 \fBar\fR consist of printable \fBASCII\fR characters. If an archive is composed
  94 of printable files, the entire archive is printable.
  95 .sp
  96 .LP
  97 When \fBar\fR creates an archive, it creates headers in a format that is
  98 portable across all machines. The portable archive format and structure are
  99 described in detail in \fBar.h\fR(3HEAD). The archive symbol table described
 100 there is used by the link editor \fBld\fR(1) to effect multiple passes over
 101 libraries of object files in an efficient manner. An archive symbol table is
 102 only created and maintained by \fBar\fR when there is at least one object file
 103 in the archive. The archive symbol table is in a specially named file that is
 104 always the first file in the archive. This file is never mentioned or
 105 accessible to the user. Whenever the \fBar\fR command is used to create or
 106 update the contents of such an archive, the symbol table is rebuilt. The
 107 \fB-s\fR option described below forces the symbol table to be rebuilt.
 108 .SH OPTIONS
 109 .sp
 110 .LP
 111 The following options are supported:
 112 .sp
 113 .ne 2
 114 .na
 115 \fB\fB-a\fR\fR
 116 .ad
 117 .RS 6n
 118 Positions new \fIfile\fRs in \fIarchive\fR after the file named by the
 119 \fIposname\fR operand.
 120 .RE

 122 .sp
 123 .ne 2
 124 .na
 125 \fB\fB-b\fR\fR
 126 .ad
```

```
.RS 6n
Positions new \fIfile\fRs in \fIarchive\fR before the file named by the
\fIposname\fR operand.
.RE

.sp
.ne 2
.na
\fB\fB-c\fR\fR
.ad
.RS 6n
Suppresses the diagnostic message that is written to standard error by default
when \fIarchive\fR is created.
.RE

.sp
.ne 2
.na
\fB\fB-C\fR\fR
.ad
.RS 6n
Prevents extracted files from replacing like-named files in the file system.
This option is useful when \fB-T\fR is also used to prevent truncated file
names from replacing files with the same prefix.
.RE

.sp
.ne 2
.na
\fB\fB-d\fR\fR
.ad
.RS 6n
Deletes one or more \fIfile\fRs from \fIarchive\fR.
.RE

.sp
.ne 2
.na
\fB\fB-i\fR\fR
.ad
.RS 6n
Positions new \fIfile\fRs in \fIarchive\fR before the file named by the
\fIposname\fR operand. This option is quivalent to \fB-b\fR.
.RE

.sp
.ne 2
.na
\fB\fB-m\fR\fR
.ad
.RS 6n
Moves \fIfile\fRs. If \fB-a\fR, \fB-b\fR, or \fB-i\fR with the \fIposname\fR
operand are specified, the \fB-m\fR option moves \fIfile\fRs to the new
position. Otherwise, \fB-m\fR moves \fIfile\fRs to the end of \fIarchive\fR.
.RE

.sp
.ne 2
.na
\fB\fB-p\fR\fR
.ad
.RS 6n
Prints the contents of \fIfile\fRs in \fIarchive\fR to standard output. If no
\fIfile\fRs are specified, the contents of all files in \fIarchive\fR are
written in the order of the archive.
.RE
```



```
.sp
.ne 2
.na
\fB\fB-q\fR\fR
.ad
.RS 6n
Quickly appends \fIfile\fRs to the end of \fIarchive\fR. Positioning options
\fB-a\fR, \fB-b\fR, and \fB-i\fR are invalid. The command does not check
whether the added \fIfile\fRs are already in \fIarchive\fR. This option is
useful to avoid quadratic behavior when creating a large archive
piece-by-piece.
.RE

.sp
.ne 2
.na
\fB\fB-r\fR\fR
.ad
.RS 6n
Replaces or adds \fIfile\fRs in \fIarchive\fR. If \fIarchive\fR does not exist,
a new archive file is created and a diagnostic message is written to standard
error, unless the \fB-c\fR option is specified. If no \fIfile\fRs are specified
and the \fIarchive\fR exists, the results are undefined. Files that replace
existing files do not change the order of the archive. If the \fB-u\fR option
is used with the \fB-r\fR option, only those files with dates of modification
later than the archive files are replaced. If the \fB-a\fR, \fB-b\fR, or
\fB-i\fR option is used, the \fIposname\fR argument must be present and
specifies that new files are to be placed after (\fB-a\fR) or before (\fB-b\fR
or \fB-i\fR) \fIposname\fR. Otherwise, the new files are placed at the end.
.RE

.sp
.ne 2
.na
\fB\fB-s\fR\fR
.ad
.RS 6n
Forces the regeneration of the archive symbol table even if \fBar\fR is not
invoked with an option that will modify the archive contents. This command is
useful to restore the archive symbol table after the \fBstrip\fR(1) command has
been used on the archive.
.RE

.sp
.ne 2
.na
\fB\fB-t\fR\fR
.ad
.RS 6n
Prints a table of contents of \fIarchive\fR. The files specified by the
\fIfile\fR operands are included in the written list. If no \fIfile\fR operands
are specified, all files in \fIarchive\fR are included in the order of the
archive.
.RE

.sp
.ne 2
.na
\fB\fB-T\fR\fR
.ad
.RS 6n
Allows file name truncation of extracted files whose archive names are longer
than the file system can support. By default, extracting a file with a name
that is too long is an error. In that case, a diagnostic message is written and
the file is not extracted.
```

```
259 .RE

261 .sp
262 .ne 2
263 .na
264 \fB\fB-u\fR\fR
265 .ad
266 .RS 6n
267 Updates older files. When used with the \fB-r\fR option, files within
268 \fIarchive\fR are replaced only if the corresponding \fIfile\fR has a
269 modification time that is at least as new as the modification time of the file
270 within \fIarchive\fR.
271 .RE

273 .sp
274 .ne 2
275 .na
276 \fB\fB-v\fR\fR
277 .ad
278 .RS 6n
279 Gives verbose output. When used with options \fB-d\fR, \fB-r\fR, or \fB-x\fR,
280 the \fB-v\fR option writes a detailed file-by-file description of the archive
281 creation and the constituent \fIfile\fRs, and maintenance activity. When used
282 with \fB-p\fR, \fB-v\fR writes the name of the file to the standard output
283 before writing the file itself to the standard output. When used with \fB-t\fR,
284 \fB-v\fR includes a long listing of information about the files within the
285 archive. When used with \fB-x\fR, \fB-v\fR prints the filename preceding each
286 extraction. When writing to an archive, \fB-v\fR writes a message to the
287 standard error.
288 .RE

290 .sp
291 .ne 2
292 .na
293 \fB\fB-V\fR\fR
294 .ad
295 .RS 6n
296 Prints its version number on standard error.
297 .RE

299 .SS "\fB/usr/xpg4/bin/ar\fR"
300 .sp
301 .LP
302 The following options are supported for \fB/usr/xpg4/bin/ar\fR:
303 .sp
304 .ne 2
305 .na
306 \fB\fB-v\fR\fR
307 .ad
308 .RS 6n
309 Same as the \fB/usr/bin/ar\fR version, except when writing to an archive, no
310 message is written to the standard error.
311 .RE

313 .sp
314 .ne 2
315 .na
316 \fB\fB-x\fR\fR
317 .ad
318 .RS 6n
319 Extracts the files named by the \fIfile\fR operands from \fIarchive\fR. The
320 contents of \fIarchive\fR are not changed. If no \fIfile\fR operands are given,
321 all files in \fIarchive\fR are extracted. If the file name of a file extracted
322 from \fIarchive\fR is longer than that supported in the directory to which it
323 is being extracted, the results are undefined. The modification time of each
324 \fIfile\fR extracted is set to the time \fIfile\fR is extracted from
```



```
325 \fIarchive\fR.
326 .RE

328 .SH OPERANDS
329 .sp
330 .LP
331 The following operands are supported:
332 .sp
333 .ne 2
334 .na
335 \fB\fIarchive\fR\fR
336 .ad
337 .RS 11n
338 A path name of the archive file.
339 .RE

341 .sp
342 .ne 2
343 .na
344 \fB\fIfile\fR\fR
345 .ad
346 .RS 11n
347 A path name. Only the last component is used when comparing against the names
348 of files in the archive. If two or more \fIfile\fR operands have the same last
349 path name component (see \fBbasename\fR(1)), the results are unspecified. The
350 implementation's archive format will not truncate valid file names of files
351 added to or replaced in the archive.
352 .RE

354 .sp
355 .ne 2
356 .na
357 \fB\fIposname\fR\fR
358 .ad
359 .RS 11n
360 The name of a file in the archive file, used for relative positioning. See
361 options \fB-m\fR and \fB-r\fR.
362 .RE

364 .SH ENVIRONMENT VARIABLES
365 .sp
366 .LP
367 See \fBenviron\fR(5) for descriptions of the following environment variables
368 that affect the execution of \fBar\fR: \fBLANG\fR, \fBLC_ALL\fR,
369 \fBLC_CTYPE\fR, \fBLC_MESSAGES\fR, \fBLC_TIME\fR, and \fBNLSPATH\fR.
370 .sp
371 .ne 2
372 .na
373 \fB\fBTMPDIR\fR\fR
374 .ad
375 .RS 10n
376 Determine the pathname that overrides the default directory for temporary
377 files, if any.
378 .RE

380 .sp
381 .ne 2
382 .na
383 \fB\fBTZ\fR\fR
384 .ad
385 .RS 10n
386 Determine the timezone used to calculate date and time strings written by
387 \fBar\fR \fB-tv\fR. If \fBTZ\fR is unset or null, an unspecified default
388 timezone is used.
389 .RE
```

```
391 .SH EXIT STATUS
392 .sp
393 .LP
394 The following exit values are returned:
395 .sp
396 .ne 2
397 .na
398 \fB\fB0\fR\fR
399 .ad
400 .RS 6n
401 Successful completion.
402 .RE

404 .sp
405 .ne 2
406 .na
407 \fB\fB>0\fR\fR
408 .ad
409 .RS 6n
410 An error occurred.
411 .RE

413 .SH ATTRIBUTES
414 .sp
415 .LP
416 See \fBattributes\fR(5) for descriptions of the following attributes:
417 .SS "\fB/usr/bin/ar\fR"
418 .sp

420 .sp
421 .TS
422 box;
423 c | c
424 l | l .
425 ATTRIBUTE TYPE  ATTRIBUTE VALUE
426 _
427 Interface Stability     Committed
428 .TE

430 .SS "\fB/usr/xpg4/bin/ar\fR"
431 .sp

433 .sp
434 .TS
435 box;
436 c | c
437 l | l .
438 ATTRIBUTE TYPE  ATTRIBUTE VALUE
439 _
440 Interface Stability     Committed
441 _
442 Standard        See \fBstandards\fR(5).
443 .TE

445 .SH SEE ALSO
446 .sp
447 .LP
448 \fBbasename\fR(1), \fBcpio\fR(1), \fBld\fR(1), \fBlorder\fR(1), \fBstrip\fR(1),
449 \fBtar\fR(1), \fBar.h\fR(3HEAD), \fBa.out\fR(4), \fBattributes\fR(5),
450 \fBenviron\fR(5), \fBstandards\fR(5)
451 .SH NOTES
452 .sp
453 .LP
454 If the same file is mentioned twice in an argument list, it may be put in the
455 archive twice.
456 .sp
```



```
457 .LP
458 By convention, archives are suffixed with "\fB\&.a\fR".
459 .sp
460 .LP
461 When inserting \fBELF\fR objects into an archive file, \fBar\fR might add
462 "\fB\en\fR" characters to pad these objects to an 8-byte boundary. Such padding
462 "\fB\n\fR" characters to pad these objects to an 8-byte boundary. Such padding
463 improves the efficiency with which \fBld\fR(1) can access the archive. Only
464 \fBELF\fR object files are padded in this way. Other archive members are not
465 altered. When an object with such padding is extracted from an archive, the
466 padding is not included in the resulting output.
```

```
**********************************************************
    5555 Tue Sep 10 18:35:18 2013
new/usr/src/man/man1/elfwrap.1
4023 - Typo in file(1) manpage and various others
**********************************************************
'\" te
.\"  Copyright (c) 2008 by 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 ELFWRAP 1 "Sep 10, 2013"
.TH ELFWRAP 1 "Mar 17, 2008"
.SH NAME
elfwrap \- wrap data in an \fBELF\fR file
.SH SYNOPSIS
.LP
.nf
\fBelfwrap\fR [\fB-64\fR] [\fB-o\fR \fIrelobj-file\fR] [\fB-z\fR target=\fBsparc
\fIdata-file\fR...
.fi

.SH DESCRIPTION
.sp
.LP
The \fBelfwrap\fR utility creates an \fBELF\fR relocatable object file from one
or more data files. The relocatable object encapsulates each data file within
an individual section, together with symbols that can be used to reference the
section. The relocatable object is appropriate for inclusion with a subsequent
link-edit. Users can reference the encapsulated data using the associated
symbols.
.sp
.LP
By default, a 32-bit \fBELF\fR relocatable object is created that is
appropriate for the machine on which \fBelfwrap\fR is executed. The \fB-64\fR
option can be used to create a 64-bit \fBELF\fR relocatable object. The \fB-z
target\fR option can be used to create a relocatable object for a specific
machine type.
.LP
Note -
.sp
.RS 2
Any data encapsulated with \fBelfwrap\fR must be in a format appropriate for
the destination target.
.RE
.sp
.LP
By default, the relocatable object \fBa.wrap.o\fR is created. The \fB-o\fR
option can be used to specify an alternative relocatable object name.
.sp
.LP
The \fBbasename\fR(1) of each data file is used to create various pieces of
\fBELF\fR information.  For example, if the input data file is
\fBISV/isv-data\fR, the following \fBELF\fR information is created within the
relocatable object.
.sp
.ne 2
.na
\fBAn \fBELF\fR section named \fB\&.isv-data\fR\fR
.ad
.sp .6
.RS 4n
This section contains the entire contents of the input data file.
.RE

.sp
.ne 2
```



```
.na
\fBAn \fBELF\fR symbol named \fBisv-data_start\fR\fR
.ad
.sp .6
.RS 4n
This symbol reflects the starting address of the \fB\&.isv-data\fR section.
.RE

.sp
.ne 2
.na
\fBAn \fBELF\fR symbol named \fBisv-data_end\fR\fR
.ad
.sp .6
.RS 4n
This symbol reflects the address of the first location after the
\fB\&.isv-data\fR section.
.RE

.SH OPTIONS
.sp
.LP
The following options are supported:
.sp
.ne 2
.na
\fB\fB-64\fR\fR
.ad
.sp .6
.RS 4n
Create a 64-bit \fBELF\fR relocatable object.
.RE

.sp
.ne 2
.na
\fB\fB-o\fR \fIrelobj-file\fR\fR
.ad
.sp .6
.RS 4n
Produce a relocatable object that is named \fIrelobj-file\fR.
.RE

.sp
.ne 2
.na
\fB\fB-z\fR target=\fBsparc\fR | \fBx86\fR\fR
.ad
.sp .6
.RS 4n
Specifies the machine type for the output relocatable object. Supported targets
are \fBsparc\fR and \fBx86\fR. The 32-bit machine type for the specified target
is used unless the \fB-64\fR option is also present, in which case the
corresponding 64-bit machine type is used. By default, the relocatable object
that is generated is 32-bit for the machine one which \fBelfwrap\fR is
executed.
.RE

.SH EXAMPLES
.sp
.LP
The following example encapsulates the system \fBpasswd\fR file and the system
\fBgroup\fR file within a relocatable object \fBpassgroup.o\fR.
.sp
.in +2
.nf
```

```
example% \fBelfwrap -o passgroup.o /etc/passwd /etc/group\fR
example% \fBelfdump -s passgroup.o | egrep "passwd|group"\fR
  [2]  0x00000000 0x00000000  SECT LOCL  D  0 .passwd
  [3]  0x00000000 0x00000000  SECT LOCL  D  0 .group
  [7]  0x00000000 0x000002f0  OBJT GLOB  D  0 .passwd  passwd_start
  [8]  0x000002f0 0x00000000  OBJT GLOB  D  0 .passwd  passwd_end
  [9]  0x00000000 0x00000121  OBJT GLOB  D  0 .group   group_start
 [10]  0x00000121 0x00000000  OBJT GLOB  D  0 .group   group_end
example% \fBstrings -N.passwd passgroup.o | head -1\fR
root:x:0:0:Super-User:/:/sbin/sh
example% \fBstrings -N.group passgroup.o | head -1\fR
root::0:
.fi
.in -2
.sp

.sp
.LP
This relocatable object can be referenced from the following user code.
.sp
.in +2
.nf
example% \fBcat main.c\fR
#include        <stdio.h>

extern char     passwd_start, passwd_end;

void main()
{
    char    *pstart = &passwd_start, *pend = &passwd_end;
    char    *str, *lstr;

    for (lstr = str = pstart; str < pend; str++) {
        if ((*str == '\en') && (str != (pend - 1)))  {
        if ((*str == '\n') && (str != (pend - 1)))  {
            (void) printf("%.*s", (++str - lstr), lstr);
            lstr = str;
        }
    }
}
```

*unchanged_portion_omitted_*

```
**********************************************************
    8891 Tue Sep 10 18:35:18 2013
new/usr/src/man/man1/file.1
4023 - Typo in file(1) manpage and various others
**********************************************************
'\" te
.\" Copyright 1989 AT&T Copyright (c) 1992, X/Open Company Limited All Rights Re
.\" Sun Microsystems, Inc. gratefully acknowledges The Open Group for permission
.\" http://www.opengroup.org/bookstore/.
.\" The Institute of Electrical and Electronics Engineers and The Open Group, ha
.\"  This notice shall appear on any product containing this material.
.\" 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 FILE 1 "Sep 10, 2013"
.TH FILE 1 "May 15, 2006"
.SH NAME
file \- determine file type
.SH SYNOPSIS
.LP
.nf
\fB/usr/bin/file\fR [\fB-dh\fR] [\fB-m\fR \fImfile\fR] [\fB-M\fR \fIMfile\fR] [\
.fi

.LP
.nf
\fB/usr/bin/file\fR [\fB-dh\fR] [\fB-m\fR \fImfile\fR] [\fB-M\fR \fIMfile\fR] \f
.fi

.LP
.nf
\fB/usr/bin/file\fR \fB-i\fR [\fB-h\fR] [\fB-f\fR \fIffile\fR] \fIfile\fR...
.fi

.LP
.nf
\fB/usr/bin/file\fR \fB-i\fR [\fB-h\fR] \fB-f\fR \fIffile\fR
.fi

.LP
.nf
\fB/usr/bin/file\fR \fB-c\fR [\fB-d\fR] [\fB-m\fR \fImfile\fR] [\fB-M\fR \fIMfil
.fi

.LP
.nf
\fB/usr/xpg4/bin/file\fR [\fB-dh\fR] [\fB-m\fR \fImfile\fR] [\fB-M\fR \fIMfile\f
.fi

.LP
.nf
\fB/usr/xpg4/bin/file\fR [\fB-dh\fR] [\fB-m\fR \fImfile\fR] [\fB-M\fR \fIMfile\f
.fi

.LP
.nf
\fB/usr/xpg4/bin/file\fR \fB-i\fR [\fB-h\fR] [\fB-f\fR \fIffile\fR] \fIfile\fR..
.fi

.LP
.nf
\fB/usr/xpg4/bin/file\fR \fB-i\fR [\fB-h\fR] \fB-f\fR \fIffile\fR
.fi

.LP
.nf
```



```
\fB/usr/xpg4/bin/file\fR \fB-c\fR [\fB-d\fR] [\fB-m\fR \fImfile\fR] [\fB-M\fR \f
.fi

.SH DESCRIPTION
.sp
.LP
The \fBfile\fR utility performs a series of tests on each file supplied by
\fIfile\fR and, optionally, on each file listed in \fIffile\fR in an attempt to
classify it. If the file is not a regular file, its file type is identified.
The file types directory, \fBFIFO\fR, block special, and character special are
identified as such. If the file is a regular file and the file is zero-length,
it is identified as an empty file.
.sp
.LP
If \fIfile\fR appears to be a text file, \fBfile\fR examines the first 512
bytes and tries to determine its programming language. If \fIfile\fR is a
symbolic link, by default the link is followed and \fBfile\fR tests the file to
which the symbolic link refers.
.sp
.LP
If \fIfile\fR is a relocatable object, executable, or shared object, \fBfile\fR
prints out information about the file's execution requirements. This
information includes the machine class, byte-ordering, static/dynamic linkage,
and any software or hardware capability requirements. If \fIfile\fR is a
runtime linking configuration file, \fBfile\fR prints information about the
target platform, including the machine class and byte-ordering.
.sp
.LP
By default, \fBfile\fR will try to use the localized magic file
\fB/usr/lib/locale/\fIlocale\fR/LC_MESSAGES/magic\fR, if it exists, to identify
files that have a magic number. For example, in the Japanese locale, \fBfile\fR
will try to use \fB/usr/lib/locale/ja/LC_MESSAGES/magic\fR. If a localized
magic file does not exist, \fBfile\fR will utilize \fB/etc/magic\fR. A magic
number is a numeric or string constant that indicates the file type. See
\fBmagic\fR(4) for an explanation of the format of \fB/etc/magic\fR.
.sp
.LP
If \fIfile\fR does not exist, cannot be read, or its file status could not be
determined, it is not considered an error that affects the exit status. The
output will indicate that the file was processed, but that its type could not
be determined.
.SH OPTIONS
.sp
.LP
The following options are supported:
.sp
.ne 2
.na
\fB\fB-c\fR\fR
.ad
.RS 12n
Checks the magic file for format errors. For reasons of efficiency, this
validation is normally not carried out.
.RE

.sp
.ne 2
.na
\fB\fB-d\fR\fR
.ad
.RS 12n
Applies any position-sensitive and context-sensitive default system tests to
the file.
.RE

.sp
```

```
.ne 2
.na
\fB\fB-f\fR \fIffile\fR\fR
.ad
.RS 12n
\fIffile\fR contains a list of the files to be examined.
.RE

.sp
.ne 2
.na
\fB\fB-h\fR\fR
.ad
.RS 12n
When a symbolic link is encountered, this option identifies the file as a
symbolic link. If \fB-h\fR is not specified and \fIfile\fR is a symbolic link
that refers to a non-existent file, the \fBfile\fR utility identifies the file
as a symbolic link, as if \fB-h\fR had been specified.
.RE

.sp
.ne 2
.na
\fB\fB-i\fR\fR
.ad
.RS 12n
If a file is a regular file, this option does not attempt to classify the type
of file further, but identifies the file as a "regular file".
.RE

.sp
.ne 2
.na
\fB\fB-m\fR \fImfile\fR\fR
.ad
.RS 12n
.sp
.ne 2
.na
\fB\fB/usr/bin/file\fR\fR
.ad
.RS 22n
Uses \fImfile\fR as an alternate magic file, instead of \fB/etc/magic\fR.
.RE

.sp
.ne 2
.na
\fB\fB/usr/xpg4/bin/file\fR\fR
.ad
.RS 22n
Specifies the name of a file containing position-sensitive tests that are
applied to a file in order to classify it (see \fBmagic\fR(4)). If the \fB-m\fR
option is specified without specifying the \fB-d\fR option or the \fB-M\fR
option, position-sensitive default system tests are applied after the
position-sensitive tests specified by the \fB-m\fR option.
.RE

.RE

.sp
.ne 2
.na
\fB\fB-M\fR \fIMfile\fR\fR
.ad
.RS 12n
```



```
Specifies the name of a file containing position-sensitive tests that are
applied to a file in order to classify it (see \fBmagic\fR(4)). No
position-sensitive default system tests nor context-sensitive default system
tests are applied unless the \fB-d\fR option is also specified.
.RE

.sp
.LP
If the \fB-M\fR option is specified with the \fB-d\fR option, the \fB-m\fR
option, or both, or if the \fB-m\fR option is specified with the \fB-d\fR
option, the concatenation of the position-sensitive tests specified by these
options is applied in the order specified by the appearance of these options.
.SH OPERANDS
.sp
.LP
The following operands are supported:
.sp
.ne 2
.na
\fB\fIfile\fR\fR
.ad
.RS 8n
A path name of a file to be tested.
.RE

.SH USAGE
.sp
.LP
See \fBlargefile\fR(5) for the description of the behavior of \fBfile\fR when
encountering files greater than or equal to 2 Gbyte ( 2^31 bytes).
.SH EXAMPLES
.LP
\fBExample 1 \fRDetermining if an Argument is a Binary Executable Files
.sp
.LP
The following example determine if an argument is a binary executable file:

.sp
.in +2
.nf
file "$1" | grep \(miFq executable &&
          printf "%s is executable.\en" "$1"
          printf "%s is executable.\n" "$1"
.fi
.in -2
.sp

.SH ENVIRONMENT VARIABLES
.sp
.LP
See \fBenviron\fR(5) for descriptions of the following environment variables
that affect the execution of \fBfile\fR: \fBLANG\fR, \fBLC_ALL\fR,
\fBLC_CTYPE\fR, \fBLC_MESSAGES\fR, and \fBNLSPATH\fR.
.SH EXIT STATUS
.sp
.LP
The following exit values are returned:
.sp
.ne 2
.na
\fB\fB0\fR\fR
.ad
.RS 6n
Successful completion.
.RE
```

```
 258 .sp
 259 .ne 2
 260 .na
 261 \fB\fB>0\fR\fR
 262 .ad
 263 .RS 6n
 264 An error occurred.
 265 .RE

 267 .SH FILES
 268 .sp
 269 .ne 2
 270 .na
 271 \fB\fB/etc/magic\fR\fR
 272 .ad
 273 .RS 14n
 274 \fBfile\fR's magic number file
 275 .RE

 277 .SH ATTRIBUTES
 278 .sp
 279 .LP
 280 See \fBattributes\fR(5) for descriptions of the following attributes:
 281 .sp

 283 .sp
 284 .TS
 285 box;
 286 c | c
 287 l | l .
 288 ATTRIBUTE TYPE  ATTRIBUTE VALUE
 289 _
 290 CSI     Enabled
 291 _
 292 Interface Stability     Standard
 293 .TE

 295 .SH SEE ALSO
 296 .sp
 297 .LP
 298 \fBcrle\fR(1), \fBelfdump\fR(1), \fBls\fR(1), \fBmagic\fR(4),
 299 \fBattributes\fR(5), \fBenviron\fR(5), \fBlargefile\fR(5), \fBstandards\fR(5)
```

```
**********************************************************
  179686 Tue Sep 10 18:35:18 2013
new/usr/src/man/man1/ksh93.1
4023 - Typo in file(1) manpage and various others
**********************************************************
```

```
'\" te
.\" Copyright (c) 1982-2007 AT&T Knowledge Ventures
.\" To view license terms, see http://www.opensource.org/licenses/cpl1.0.txt
.\" Portions Copyright (c) 2009, Sun Microsystems, Inc.
.TH KSH93 1 "Sep 10, 2013"
.TH KSH93 1 "Aug 11, 2009"
.SH NAME
ksh93, rksh93 \- Korn Shell, a standard and restricted command and programming
language
.SH SYNOPSIS
.LP
.nf
\fBksh93\fR [\fB\(+-abcefhikmnoprstuvxBCD\fR] [\fB-R\fR \fIfile\fR] [ \fB\(+-o\f
      [-] [\fIarg\fR ...]
.fi

.LP
.nf
\fBrksh93\fR [\fB\(+-abcefhikmnoprstuvxBCD\fR] [\fB-R\fR \fIfile\fR] [\fB\(+-o\f
      [-] [\fIarg\fR ...]
.fi

.SH DESCRIPTION
.sp
.LP
\fBksh93\fR is a command and programming language that executes commands read
from a terminal or a file. \fBrksh93\fR is a restricted version of the command
interpreter \fBksh93\fR. \fBrksh93\fR is used to set up login names and
execution environments whose capabilities are more controlled than those of the
standard shell.
.sp
.LP
See \fBInvocation\fR for the meaning of arguments to the shell.
.SS "Definitions"
.sp
.LP
A \fImetacharacter\fR is defined as one of the following characters:
.sp
.in +2
.nf
; & ( ) | < > NEWLINE SPACE TAB
.fi
.in -2
.sp

.sp
.LP
A \fIblank\fR is a \fBTAB\fR or a \fBSPACE\fR.
.sp
.LP
An \fIidentifier\fR is a sequence of letters, digits, or underscores starting
with a letter or underscore. Identifiers are used as components of \fIvariable
names\fR.
.sp
.LP
A \fIvname\fR is a sequence of one or more identifiers separated by a period
(\fB\&.\fR) and optionally preceded by a period (\fB\&.\fR). \fIvnames\fR are
used as function and variable names.
.sp
.LP
A \fIword\fR is a sequence of \fIcharacters\fR from the character set defined
```



```
by the current locale, excluding non-quoted \fImetacharacters\fR.
.sp
.LP
A \fIcommand\fR is a sequence of characters in the syntax of the shell
language. The shell reads each command and carries out the desired action
either directly or by invoking separate utilities. A built-in command is a
command that is carried out by the shell itself without creating a separate
process. Some commands are built-in purely for convenience and are not
documented in this manual page. Built-ins that cause side effects in the shell
environment and built-ins that are found before performing a path search (see
\fBExecution\fR) are documented in this manual page. For historical reasons,
some of these built-ins behave differently than other built-ins and are called
special built-ins.
.SS "Commands"
.sp
.LP
A \fIsimple-command\fR is a list of variable assignments (see \fBVariable
Assignments\fR) or a sequence of \fIblank\fR-separated words which can be
preceded by a list of variable assignments. See the \fBEnvironment\fR section
of this manual page.
.sp
.LP
The first word specifies the name of the command to be executed. Except as
specified in this section, the remaining words are passed as arguments to the
invoked command. The command name is passed as argument 0. See \fBexec\fR(2).
The \fIvalue\fR of a simple-command is its exit status. If it terminates
normally, its value is \fB0\fR-\fB255\fR. If it terminates abnormally, its
value is \fB256+\fR\fIsignum\fR. The name of the signal corresponding to the
exit status can be obtained by way of the \fB-l\fR option of the kill built-in
utility.
.sp
.LP
A \fIpipeline\fR is a sequence of one or more commands separated by \fB|\fR.
The standard output of each command but the last is connected by a
\fBpipe\fR(2) to the standard input of the next command. Each command, except
possibly the last, is run as a separate process. The shell waits for the last
command to terminate. The exit status of a pipeline is the exit status of the
last command unless the \fBpipefail\fR option is enabled. Each pipeline can be
preceded by the reserved word\fB!\fR. This causes the exit status of the
pipeline to become \fB0\fR if the exit status of the last command is
\fBnon-zero\fR, and \fB1\fR if the exit status of the last command is \fB0\fR.
.sp
.LP
A \fIlist\fR is a sequence of one or more pipelines separated by \fB;, &, |&,
&&, or |\fR, and optionally terminated by \fB;, &,\fR or \fB|&\fR. Of these
five symbols, \fB;, &\fR, and \fB|&\fR have equal precedence, which is lower
than that of \fB&&\fR and \fB||\fR. The symbols \fB&&\fR and \fB||\fR also have
equal precedence.
.sp
.LP
A semicolon (\fB;\fR) causes sequential execution of the preceding pipeline. An
ampersand (\fB&\fR) causes asynchronous execution of the preceding pipeline,
that is, the shell does \fInot\fR wait for that pipeline to finish. The symbol
\fB|&\fR causes asynchronous execution of the preceding pipeline with a two-way
pipe established to the parent shell. The standard input and output of the
spawned pipeline can be written to and read from by the parent shell by
applying the redirection operators \fB<&\fR and \fB>&\fR with \fBarg p\fR to
commands and by using \fB-p\fR option of the built-in commands \fBread\fR and
\fBprint\fR. The symbol \fB&&\fR (\fB||\fR) causes the \fIlist\fR following it
to be executed only if the preceding pipeline returns a zero (\fBnon-zero\fR)
value. One or more NEWLINEs can appear in a \fIlist\fR instead of a semicolon,
to delimit a command. The first \fIitem\fR of the first \fIpipeline\fR of a
\fIlist\fR that is a simple command not beginning with a redirection, and not
occurring within a \fBwhile\fR, \fBuntil\fR, or \fBif\fR \fIlist\fR , can be
preceded by a semicolon. This semicolon is ignored unless the \fBshowme\fR
option is enabled as described with the \fBset\fR built-in.
```

```
.sp
.LP
A \fICommand\fR is either a simple-command or one of commands in the following
list. Unless otherwise stated, the value returned by a command is that of the
last simple-command executed in the command.
.sp
.ne 2
.na
\fB\fBfor\fR \fIvname\fR \fB[ in\fR \fIword\fR \fB\&... ] ;do\fR \fIlist\fR
\fB;done\fR\fR
.ad
.sp .6
.RS 4n
Each time a \fBfor\fR command is executed, \fIvname\fR is set to the next
\fIword\fR taken from the \fBin\fR \fIword\fR list. If \fBin\fR \fIword ...\fR
is omitted, the \fBfor\fR command executes the \fBdo\fR \fIlist\fR once for
each positional parameter that is set starting from 1. Execution ends when
there are no more words in the list. See \fBParameter Expansion\fR.
.RE

.sp
.ne 2
.na
\fB\fB(( [\fR\fIexpr1\fR\fB] ; [\fR\fIexpr2\fR] ; [\fIexpr3\fR\fB] )) ;do\fR
\fIlist\fR \fB;done\fR\fR
.ad
.sp .6
.RS 4n
The arithmetic expression \fIexpr1\fR is evaluated first. The arithmetic
expression \fIexpr2\fR is repeatedly evaluated until it evaluates to \fBzero\fR
and when \fBnon-zero\fR, \fIlist\fR is executed and the arithmetic expression
\fIexpr3\fR evaluated. If any expression is omitted, then it behaves as if it
evaluated to \fB1\fR. See \fBArithmetic Evaluation\fR.
.RE

.sp
.ne 2
.na
\fB\fBselect\fR \fIvname\fR [ in \fIword\fR \fB\&... ] ;do\fR \fIlist\fR
\fB;done\fR\fR
.ad
.sp .6
.RS 4n
A \fBselect\fR command prints on standard error (file descriptor 2) the set of
\fIwords\fR, each preceded by a number. If \fBin\fR \fIword...\fR is omitted,
the positional parameters starting from \fB1\fR are used instead. See
\fBParameter Expansion\fR. The \fBPS3\fR prompt is printed and a line is read
from the standard input. If this line consists of the number of one of the
listed \fIword\fRs, then the value of the variable \fIvname\fR is set to the
\fIword\fR corresponding to this number. If this line is empty, the selection
list is printed again. Otherwise the value of the variable \fIvname\fR is set
to \fBnull\fR. The contents of the line read from standard input is saved in
the variable \fBREPLY\fR. The \fIlist\fR is executed for each selection until a
break or \fBEOF\fR is encountered. If the \fBREPLY\fR variable is set to
\fBnull\fR by the execution of \fIlist\fR, the selection list is printed before
displaying the \fBPS3\fR prompt for the next selection.
.RE

.sp
.ne 2
.na
\fB\fBcase\fR \fIword\fR \fBin [ [(]\fR \fIpattern\fR \fB[ |\fR \fIpattern\fR
\fB] ... )\fR \fIlist\fR \fB;; ] ... esac\fR\fR
.ad
.sp .6
.RS 4n
```



```
A \fBcase\fR command executes the \fIlist\fR associated with the first
\fIpattern\fR that matches \fIword\fR. The form of the patterns is the same as
that used for file name generation. See \fBFile Name Generation\fR.
.sp
The \fB;;\fR operator causes execution of \fBcase\fR to terminate. If \fB;&\fR
is used in place of \fB;;\fR the next subsequent list, if any, is executed.
.RE

.sp
.ne 2
.na
\fB\fBif\fR \fIlist\fR \fB;then\fR \fIlist\fR \fB[ ;elif\fR \fIlist\fR
\fB;then\fR \fIlist\fR ] \fB\&... [ ;else\fR \fIlist\fR \fB]\fR \fB;fi\fR\fR
.ad
.sp .6
.RS 4n
The \fIlist\fR following \fBif\fR is executed and, if it returns a \fBzero\fR
exit status, the \fIlist\fR following the first \fBthen\fR is executed.
Otherwise, the \fIlist\fR following \fBelif\fR is executed, and, if its value
is \fBzero\fR, the \fIlist\fR following the next \fBthen\fR is executed.
Failing each successive \fBelif\fR \fIlist\fR, the \fBelse\fR \fIlist\fR is
executed. If the \fBif\fR \fIlist\fR has \fBnon-zero\fR exit status and there
is no \fBelse\fR \fIlist\fR, then the \fBif\fR command returns a \fBzero\fR
exit status.
.RE

.sp
.ne 2
.na
\fB\fBwhile\fR \fIlist\fR \fB;do\fR \fIlist\fR \fB;done\fR\fR
.ad
.br
.na
\fBuntil \fIlist\fR \fB;do\fR \fIlist\fR \fB;done\fR\fR
.ad
.sp .6
.RS 4n
A \fBwhile\fR command repeatedly executes the while \fIlist\fR and, if the exit
status of the last command in the list is zero, executes the \fBdo\fR
\fIlist\fR, otherwise the loop terminates. If no commands in the \fBdo\fR
\fIlist\fR are executed, then the \fBwhile\fR command returns a \fBzero\fR exit
status, \fBuntil\fR can be used in place of \fBwhile\fR to negate the loop
termination test.
.RE

.sp
.ne 2
.na
\fB\fB((\fR\fIexpression\fR\fB))\fR\fR
.ad
.sp .6
.RS 4n
The \fIexpression\fR is evaluated using the rules for arithmetic evaluation
described in this manual page. If the value of the arithmetic expression is
\fBnon-zero\fR, the exit status is \fB0\fR. Otherwise the exit status is
\fB1\fR.
.RE

.sp
.ne 2
.na
\fB\fB(\fR\fIlist\fR\fB;)\fR\fR
.ad
.sp .6
.RS 4n
Execute list in a separate environment. If two adjacent open parentheses are
```

```
needed for nesting, a SPACE must be inserted to avoid evaluation as an
arithmetic command as described in this section.
.sp
\fIlist\fR is simply executed. Unlike the metacharacters, \fB(\fR and \fB)\fR,
\fB{\fR and \fB}\fR are \fIreserved words\fR and must occur at the beginning of
a line or after a \fB;\fR to be recognized.
.RE

.sp
.ne 2
.na
\fB\fB[[\fR \fIexpression\fR \fB]]\fR\fR
.ad
.sp .6
.RS 4n
Evaluates \fIexpression\fR and returns a \fBzero\fR exit status when
\fIexpression\fR is true. See \fBConditional Expressions\fR for a description
of \fIexpression\fR.
.RE

.sp
.ne 2
.na
\fB\fBfunction\fR \fIvarname\fR \fB{\fR \fIlist\fR \fB;}\fR\fR
.ad
.br
.na
\fB\fIvarname\fR \fB() {\fR \fIlist\fR \fB;}\fR\fR
.ad
.sp .6
.RS 4n
Define a function which is referenced by \fIvarname\fR. A function whose
\fIvarname\fR contains a \fB\&.\fR is called a discipline function and the
portion of the \fIvarname\fR preceding the last \fB\&.\fR must refer to an
existing variable.
.sp
The body of the function is the \fIlist\fR of commands between \fB{\fR and
\fB}\fR. A function defined with the function \fIvarname\fR syntax can also be
used as an argument to the \fB\&.\fR special built-in command to get the
equivalent behavior as if the \fIvarname\fR\fB()\fR syntax were used to define
it. See \fBFunctions\fR.
.RE

.sp
.ne 2
.na
\fB\fBtime [\fR \fIpipeline\fR \fB]\fR\fR
.ad
.sp .6
.RS 4n
If \fIpipeline\fR is omitted, the user and system time for the current shell
and completed child processes is printed on standard error. Otherwise,
\fIpipeline\fR is executed and the elapsed time as well as the user and system
time are printed on standard error. The \fBTIMEFORMAT\fR variable can be set to
a format string that specifies how the timing information should be displayed.
See \fBShell Variables\fR for a description of the \fBTIMEFORMAT\fR variable.
.RE

.sp
.LP
The following reserved words are recognized as reserved only when they are the
first word of a command and are not quoted:
.br
.in +2
\fBcase\fR
.in -2
```



```
.br
.in +2
\fBdo\fR
.in -2
.br
.in +2
\fBdone\fR
.in -2
.br
.in +2
\fBelse\fR
.in -2
.br
.in +2
\fBelif\fR
.in -2
.br
.in +2
\fBesac\fR
.in -2
.br
.in +2
\fBfor\fR
.in -2
.br
.in +2
\fBfi\fR
.in -2
.br
.in +2
\fBfunction\fR
.in -2
.br
.in +2
\fBif\fR
.in -2
.br
.in +2
\fBselect\fR
.in -2
.br
.in +2
\fBthen\fR
.in -2
.br
.in +2
\fBtime\fR
.in -2
.br
.in +2
\fBuntil\fR
.in -2
.br
.in +2
\fBwhile\fR
.in -2
.br
.in +2
\fB{ }\fR
.in -2
.br
.in +2
\fB[[ ]]\fR
.in -2
.br
.in +2
```

```
391 \fB!\fR
392 .in -2
393 .SS "Variable Assignments"
394 .sp
395 .LP
396 One or more variable assignments can start a simple command or can be arguments
397 to the \fBtypeset\fR, \fBexport\fR, or \fBreadonly\fR special built-in
398 commands. The syntax for an \fIassignment\fR is of the form:
399 .sp
400 .ne 2
401 .na
402 \fB\fIvarname\fR\fB=\fR\fIword\fR\fR
403 .ad
404 .br
405 .na
406 \fB\fIvarname\fR\fB[\fR\fIword\fR\fB]=\fR\fIword\fR\fR
407 .ad
408 .sp .6
409 .RS 4n
410 No space is permitted between \fIvarname\fR and the \fB=\fR or between \fB=\fR
411 and \fIword\fR.
412 .RE

414 .sp
415 .ne 2
416 .na
417 \fB\fIvarname\fR\fB=(\fR\fIassignlist\fR\fB)\fR\fR
418 .ad
419 .sp .6
420 .RS 4n
421 No space is permitted between \fIvarname\fR and the \fB=\fR. An
422 \fIassignlist\fR can be one of the following:
423 .sp
424 .ne 2
425 .na
426 \fB\fIword ...\fR\fR
427 .ad
428 .sp .6
429 .RS 4n
430 Indexed array assignment.
431 .RE

433 .sp
434 .ne 2
435 .na
436 \fB\fB[\fR\fIword\fR\fB]=\fR\fIword ...\fR\fR
437 .ad
438 .sp .6
439 .RS 4n
440 Associative array assignment. If prefixed by \fBtypeset\fR \fB-a\fR, creates an
441 indexed array instead.
442 .RE

444 .sp
445 .ne 2
446 .na
447 \fB\fIassignment ...\fR\fR
448 .ad
449 .sp .6
450 .RS 4n
451 Compound variable assignment. This creates a compound variable \fIvarname\fR
452 with sub-variables of the form \fIvarname.name\fR, where \fIname\fR is the name
453 portion of assignment. The value of \fIvarname\fR contains all the assignment
454 elements. Additional assignments made to sub-variables of \fIvarname\fR are
455 also displayed as part of the value of \fIvarname\fR. If no \fIassignment\fRs
456 are specified, \fIvarname\fR is a compound variable allowing subsequence child
```



```
457 elements to be defined.
458 .RE

460 .sp
461 .ne 2
462 .na
463 \fB\fBtypeset [\fR\fIoptions\fR] \fIassignment\fR \fB\&...\fR\fR
464 .ad
465 .sp .6
466 .RS 4n
467 Nested variable assignment. Multiple assignments can be specified by separating
468 each of them with a \fB;\fR. The previous value is unset before the assignment.
469 .RE

471 In addition, a \fB+=\fR can be used in place of the \fB=\fR to signify adding
472 to or appending to the previous value. When \fB+=\fR is applied to an
473 arithmetic type, \fIword\fR is evaluated as an arithmetic expression and added
474 to the current value. When applied to a string variable, the value defined by
475 \fIword\fR is appended to the value. For compound assignments, the previous
476 value is not unset and the new values are appended to the current ones provided
477 that the types are compatible.
478 .RE

480 .SS "Comments"
481 .sp
482 .LP
483 A word beginning with \fB#\fR causes that word and all the following characters
484 up to a NEWLINE to be commented, or ignored.
485 .SS "Aliasing"
486 .sp
487 .LP
488 The first word of each command is replaced by the text of an alias if an alias
489 for this word has been defined. An alias name consists of any number of
490 characters excluding metacharacters, quoting characters, file expansion
491 characters, parameter expansion characters, command substitution characters,
492 and \fB=\fR. The replacement string can contain any valid shell script
493 including the metacharacters listed in the \fBCommands\fR section. The first
494 word of each command in the replaced text, other than any that are in the
495 process of being replaced, are tested for aliases. If the last character of the
496 alias value is a BLANK then the word following the alias is also checked for
497 alias substitution.
498 .sp
499 .LP
500 Aliases can be used to redefine built-in commands but cannot be used to
501 redefine the reserved words listed in the \fBCommands\fR section. Aliases can
502 be created and listed with the alias command and can be removed with the
503 \fBunalias\fR command.
504 .sp
505 .LP
506 Aliasing is performed when scripts are read, not while they are executed. For
507 an alias to take effect, the \fBalias\fR definition command has to be executed
508 before the command which references the alias is read. The following aliases
509 are compiled into the shell but can be unset or redefined:
510 .sp
511 .in +2
512 .nf
513 autoload='typeset -fu'
514 command='command '
515 fc=hist
516 float='typeset -lE'
517 functions='typeset -f'
518 hash='alias -t --'
519 history='hist -l'
520 integer='typeset -li'
521 nameref='typeset -n'
522 nohup='nohup '
```

```
r='hist -s'
redirect='command exec'
source='command .'
stop='kill -s STOP'
suspend='kill -s STOP $$'
times='{ { time;} 2>&1;}'
type='whence -v'
.fi
.in -2
.sp

.SS "Tilde Substitution"
.sp
.LP
After alias substitution is performed, each word is checked to see if it begins
with an unquoted tilde (\fB~\fR). For tilde substitution, \fIword\fR also
refers to the \fIword\fR portion of parameter expansion. See \fBParameter
Expansion\fR.
.sp
.LP
If it does, the word up to a \fB/\fR is checked to see if it matches a user
name in the password database. If a match is found, the \fB~\fR and the matched
login name are replaced by the login directory of the matched user. If no match
is found, the original text is left unchanged. A \fB~\fR by itself, or in front
of a \fB/\fR, is replaced by \fB$HOME\fR. A \fB~\fR followed by a \fB+\fR or
\fB-\fR is replaced by the value of \fB$PWD\fR and \fB$OLDPWD\fR respectively.
.sp
.LP
In addition, when expanding a \fIvariable assignment\fR, tilde substitution is
attempted when the value of the assignment begins with a \fB~\fR, and when a
\fB~\fR appears after a colon (\fB:\fR). The \fB:\fR also terminates a \fB~\fR
login name.
.SS "Command Substitution"
.sp
.LP
The standard output from a command enclosed in parentheses preceded by a dollar
sign (\fB$\fR) or a pair of grave accents (\fB``\fR) can be used as part or all
of a word. Trailing NEWLINEs are removed. In the second (obsolete) form, the
string between the quotes is processed for special quoting characters before
the command is executed. See \fBQuoting\fR.
.sp
.LP
The command substitution \fB$(cat file)\fR can be replaced by the equivalent
but faster \fB$(<file)\fR. The command substitution \fB$(\fR\fIn\fR\fB<#)\fR
expands to the current byte offset for file descriptor \fIn\fR.
.SS "Arithmetic Substitution"
.sp
.LP
An arithmetic expression enclosed in double parentheses preceded by a dollar
sign ( \fB$((\fR\fIarithmetic_expression\fR\fB))\fR) is replaced by the value
of the arithmetic expression within the double parentheses.
.SS "Process Substitution"
.sp
.LP
Process substitution is only available on versions of the UNIX operating system
that support the \fB/dev/fd\fR directory for naming open files.
.sp
.LP
Each command argument of the form \fB<(\fR\fIlist\fR\fB)\fR or
\fB>(\fR\fIlist\fR\fB)\fR runs process \fIlist\fR asynchronously connected to
some file in \fB/dev/fd\fR. The name of this file becomes the argument to the
command. If the form with \fB>\fR is selected then writing on this file
provides input for \fIlist\fR. If \fB<\fR is used, then the file passed as an
argument contains the output of the \fIlist\fR process.
.sp
.LP
```



```
For example,
.sp
.in +2
.nf
paste <(cut -f1 \fIfile1\fR) <(cut -f3 \fIfile2\fR) | tee \e
     >(\fIprocess1\fR) >(\fIprocess2\fR)
.fi
.in -2
.sp

.sp
.LP
\fBcut\fRs fields 1 and 3 from the files \fIfile1\fR and \fIfile2\fR
respectively, \fBpaste\fRs the results together, and sends it to the processes
\fIprocess1\fR and \fIprocess2\fR. It also displays the results to the standard
output. The file, which is passed as an argument to the command, is a UNIX
\fBpipe\fR(2). Programs that expect to \fBlseek\fR(2) on the file do not work.
.SS "Parameter Expansion"
.sp
.LP
A parameter is a variable, one or more digits, or any of the characters
\fB*\fR, \fB@\fR, \fB#\fR, \fB?\fR, \fB-\fR, \fB$\fR, and \fB!\fR. A variable
is denoted by a \fIvname\fR. To create a variable whose \fIvname\fR contains a
\fB\&.\fR, a variable whose \fBvname\fR consists of everything before the
last . must already exist. A variable has a value and zero or more attributes.
Variables can be assigned values and attributes by using the \fBtypeset\fR
special built-in command. The attributes supported by the shell are described
later with the \fBtypeset\fR special built-in command. Exported variables pass
values and attributes to the environment.
.sp
.LP
The shell supports both indexed and associative arrays. An element of an array
variable is referenced by a subscript. A subscript for an indexed array is
denoted by an arithmetic expression, (see \fBArithmetic Evaluation\fR), between
a \fB[\fR and a \fB]\fR. Use \fBset -A\fR \fIvname value ...\fR to assign
values to an indexed array. The value of all subscripts must be in the range of
\fB0\fR through \fB1,048,575\fR. Indexed arrays do not need to be declared. Any
reference to a variable with a valid subscript is legal and an array is created
if necessary.
.sp
.LP
An associative array is created with the \fB-A\fR option to \fBtypeset\fR. A
subscript for an associative array is denoted by a string enclosed between
\fB[\fR and \fB]\fR.
.sp
.LP
Referencing any array without a subscript is equivalent to referencing the
array with subscript \fB0\fR.
.sp
.LP
The value of a variable can be assigned by:
.sp
.in +2
.nf
\fIvname\fR=\fIvalue\fR [\fIvname\fR=\fIvalue\fR] ...
.fi
.in -2
.sp

.sp
.LP
or
.sp
.in +2
.nf
\fIvname\fR[\fIsubscript\fR]=\fIvalue\fR [\fIvname\fR[\fIsubscript\fR]=\fIvalue\
```

```
.fi
.in -2
.sp

.sp
.LP
No space is allowed before or after the \fB=\fR. A \fInameref\fR is a variable
that is a reference to another variable. A \fInameref\fR is created with the
\fB-n\fR attribute of \fBtypeset\fR. The value of the variable at the time of
the \fBtypeset\fR command becomes the variable that is referenced whenever the
\fInameref\fR variable is used. The name of a \fInameref\fR cannot contain a
dot (\fI\&.\fR). When a variable or function name contains a \fB\&.,\fR and the
portion of the name up to the first \fB\&.\fR matches the name of a
\fInameref\fR, the variable referred to is obtained by replacing the
\fInameref\fR portion with the name of the variable referenced by the
\fInameref\fR. If a \fInameref\fR is used as the index of a \fBfor\fR loop, a
name reference is established for each item in the list. A \fInameref\fR
provides a convenient way to refer to the variable inside a function whose name
is passed as an argument to a function. For example, if the name of a variable
is passed as the first argument to a function, the command
.sp
.in +2
.nf
 typeset -n var=$1
.fi
.in -2
.sp

.sp
.LP
inside the function causes references and assignments to \fIvar\fR to be
references and assignments to the variable whose name has been passed to the
function. If either of the floating point attributes, \fB-E\fR, or \fB-F\fR, or
the integer attribute, \fB-i\fR, is set for \fIvname\fR, then the \fIvalue\fR
is subject to arithmetic evaluation as described in this manual page.
Positional parameters, parameters denoted by a number, can be assigned values
with the \fBset\fR special built-in command. Parameter \fB$0\fR is set from
argument zero when the shell is invoked. The character \fB$\fR is used to
introduce substitutable parameters.
.sp
.ne 2
.na
\fB\fB${\fR\fIparameter\fR\fB}\fR\fR
.ad
.sp .6
.RS 4n
The shell reads all the characters from \fB${\fR to the matching \fB}\fR as
part of the same word even if it contains braces or metacharacters. The value,
if any, of the parameter is substituted. The braces are required when
\fIparameter\fR is followed by a letter, digit, or underscore that is not to be
interpreted as part of its name, when the variable name contains a \fB\&.\fR,
or when a variable is subscripted. If \fIparameter\fR is one or more digits
then it is a positional parameter. A positional parameter of more than one
digit must be enclosed in braces. If \fIparameter\fR is \fB*\fR or \fB@\fR,
then all the positional parameters, starting with \fB$1\fR, are substituted and
separated by a field separator character. If an array \fIvname\fR with
subscript \fB*\fR or \fB@\fR is used, then the value for each of the elements
is substituted, separated by the first character of the value of \fBIFS\fR.
.RE

.sp
.ne 2
.na
\fB\fB${#\fR\fIparameter\fR\fB}\fR\fR
.ad
.sp .6
```



```
.RS 4n
If \fIparameter\fR is \fB*\fR or \fB@\fR, the number of positional parameters
is substituted. Otherwise, the length of the value of the \fIparameter\fR is
substituted.
.RE

.sp
.ne 2
.na
\fB\fB${#\fR\fIvname\fR\fB[*]}\fR\fR
.ad
.br
.na
\fB\fB${#\fR\fIvname\fR\fB[@]}\fR\fR
.ad
.sp .6
.RS 4n
The number of elements in the array \fIvname\fR is substituted.
.RE

.sp
.ne 2
.na
\fB\fB${!\fR\fIvname\fR\fB}\fR\fR
.ad
.sp .6
.RS 4n
Expands to the name of the variable referred to by \fIvname\fR. This is
\fIvname\fR except when \fIvname\fR is a name reference.
.RE

.sp
.ne 2
.na
\fB\fB${!\fR\fIvname\fR\fB[\fR\fIsubscript\fR\fB]}\fR\fR
.ad
.sp .6
.RS 4n
Expands to name of the subscript unless \fIsubscript\fR is \fB*\fR or \fB@\fR.
When \fIsubscript\fR is \fB*\fR, the list of array subscripts for \fIvname\fR
is generated. For a variable that is not an array, the value is \fB0\fR if the
variable is set. Otherwise it is \fBnull\fR. When \fIsubscript\fR is \fB@\fR,
it is the same as \fB${\fR\fIvname\fR\fB[\fR\fI*\fR\fB]}\fR, except that when
used in double quotes, each array subscript yields a separate argument.
.RE

.sp
.ne 2
.na
\fB\fB${!\fR\fIprefix\fR\fB*}\fR\fR
.ad
.sp .6
.RS 4n
Expands to the names of the variables whose names begin with \fIprefix\fR.
.RE

.sp
.ne 2
.na
\fB\fB${\fR\fIparameter\fR\fB:-\fR\fIword\fR\fB}\fR\fR
.ad
.sp .6
.RS 4n
If \fIparameter\fR is set and is non-null then substitute its value. Otherwise
substitute \fIword\fR.
.sp
```

```
\fIword\fR is not evaluated unless it is to be used as the substituted string.
.sp
In the following example, \fBpwd\fR is executed only if \fBd\fR is not set or
is NULL:
.sp
.in +2
.nf
print ${d:-$(pwd)}
.fi
.in -2
.sp

If the colon (\fB:\fR ) is omitted from the expression, the shell only checks
whether \fIparameter\fR is set or not.
.RE

.sp
.ne 2
.na
\fB\fB${\fR\fIparameter\fR\fB:=\fR\fIword\fR\fB}\fR\fR
.ad
.sp .6
.RS 4n
If \fIparameter\fR is not set or is \fBnull\fR, set it to \fIword\fR. The value
of the parameter is then substituted. Positional parameters cannot be assigned
to in this way.
.sp
\fIword\fR is not evaluated unless it is to be used as the substituted string.
.sp
In the following example, \fBpwd\fR is executed only if \fBd\fR is not set or
is NULL:
.sp
.in +2
.nf
print ${d:-$(pwd)}
.fi
.in -2
.sp

If the colon (\fB:\fR) is omitted from the expression, the shell only checks
whether \fIparameter\fR is set or not.
.RE

.sp
.ne 2
.na
\fB\fB${\fR\fIparameter\fR\fB:?\fR\fIword\fR\fB}\fR\fR
.ad
.sp .6
.RS 4n
If \fIparameter\fR is set and is non-null, substitute its value. Otherwise,
print \fIword\fR and exit from the shell , if the shell is not interactive. If
\fIword\fR is omitted then a standard message is printed.
.sp
\fIword\fR is not evaluated unless it is to be used as the substituted string.
.sp
In the following example, \fBpwd\fR is executed only if \fBd\fR is not set or
is NULL:
.sp
.in +2
.nf
print ${d:-$(pwd)}
.fi
.in -2
.sp
```



```
If the colon (\fB:\fR ) is omitted from the expression, the shell only checks
whether \fIparameter\fR is set or not.
.RE

.sp
.ne 2
.na
\fB\fB${\fR\fIparameter\fR\fB:+\fR\fIword\fR\fB}\fR\fR
.ad
.sp .6
.RS 4n
If \fIparameter\fR is set and is non-null, substitute \fIword\fR. Otherwise
substitute nothing.
.sp
\fIword\fR is not evaluated unless it is to be used as the substituted string.
.sp
In the following example, \fBpwd\fR is executed only if \fBd\fR is not set or
is NULL:
.sp
.in +2
.nf
print ${d:-$(pwd)}
.fi
.in -2
.sp

If the colon (\fB:\fR) is omitted from the expression, the shell only checks
whether \fIparameter\fR is set or not.
.RE

.sp
.ne 2
.na
\fB\fB${\fR\fIparameter\fR\fB:\fR\fIoffset\fR:\fIlength\fR}\fR
.ad
.br
.na
\fB\fB${\fR\fIparameter\fR\fB:\fR\fIoffset\fR\fB}\fR\fR
.ad
.sp .6
.RS 4n
Expands to the portion of the value of \fIparameter\fR starting at the
character (counting from \fB0\fR) determined by expanding offset as an
arithmetic expression and consisting of the number of characters determined by
the arithmetic expression defined by \fIlength\fR.
.sp
In the second form, the remainder of the value is used. A negative offset
counts backwards from the end of \fIparameter\fR.
.sp
One or more BLANKs is required in front of a minus sign to prevent the shell
from interpreting the operator as \fB:-\fR. If parameter is \fB*\fR or \fB@\fR,
or is an array name indexed by \fB*\fR or \fB@\fR, then \fIoffset\fR and
\fIlength\fR refer to the array index and number of elements respectively. A
negative \fIoffset\fR is taken relative to one greater than the highest
subscript for indexed arrays. The order for associative arrays is unspecified.
.RE

.sp
.ne 2
.na
\fB\fB${\fR\fIparameter\fR\fB#\fR\fIpattern\fR\fB}\fR\fR
.ad
.br
.na
\fB\fB${\fR\fIparameter\fR\fB##\fR\fIpattern\fR\fB}\fR\fR
.ad
```

```
 919 .sp .6
 920 .RS 4n
 921 If the shell \fIpattern\fR matches the beginning of the value of
 922 \fIparameter\fR, then the value of this expansion is the value of the
 923 \fIparameter\fR with the matched portion deleted. Otherwise the value of this
 924 \fIparameter\fR is substituted. In the first form the smallest matching
 925 \fIpattern\fR is deleted and in the second form the largest matching
 926 \fIpattern\fR is deleted. When \fIparameter\fR is \fB@\fR, \fB*\fR, or an array
 927 variable with subscript \fB@\fR or \fB*\fR, the substring operation is applied
 928 to each element in turn.
 929 .RE

 931 .sp
 932 .ne 2
 933 .na
 934 \fB\fB${\fR\fIparameter\fR\fB%\fR\fIpattern\fR\fB}\fR\fR
 935 .ad
 936 .br
 937 .na
 938 \fB\fB${\fR\fIparameter\fR\fB%%\fR\fIpattern\fR\fB}\fR\fR
 939 .ad
 940 .sp .6
 941 .RS 4n
 942 If the shell \fIpattern\fR matches the end of the value of \fIparameter\fR,
 943 then the value of this expansion is the value of the parameter with the matched
 944 part deleted. Otherwise substitute the value of \fIparameter\fR. In the first
 945 form the smallest matching pattern is deleted, and in the second form the
 946 largest matching pattern is deleted. When parameter is \fB@\fR, \fB*\fR, or an
 947 array variable with subscript \fB@\fR or \fB*\fR, the substring operation is
 948 applied to each element in turn.
 949 .RE

 951 .sp
 952 .ne 2
 953 .na
 954 \fB\fB${\fR\fIparameter\fR\fB/\fR\fIpattern\fR\fB/\fR\fIstring\fR\fB}\fR\fR
 955 .ad
 956 .br
 957 .na
 958 \fB\fB${\fR\fIparameter\fR\fB//\fR\fIpattern\fR\fB/\fR\fIstring\fR\fB}\fR\fR
 959 .ad
 960 .br
 961 .na
 962 \fB\fB${\fR\fIparameter\fR\fB/#\fR\fIpattern\fR\fB/\fR\fIstring\fR\fB}\fR\fR
 963 .ad
 964 .br
 965 .na
 966 \fB\fB${\fR\fIparameter\fR\fB/%\fR\fIpattern\fR\fB/\fR\fIstring\fR\fB}\fR\fR
 967 .ad
 968 .sp .6
 969 .RS 4n
 970 Expands \fIparameter\fR and replaces the longest match of \fIpattern\fR with
 971 the specified \fIstring\fR. Each occurrence of \fB\e\fR\fIn\fR in \fIstring\fR
 972 is replaced by the portion of \fIparameter\fR that matches the \fIn\fR\fBth\fR
 973 sub-pattern.
 974 .sp
 975 When \fIstring\fR is null, the \fIpattern\fR is deleted and the \fB/\fR in
 976 front of string can be omitted. When \fIparameter\fR is \fB@\fR, \fB*\fR, or an
 977 array variable with subscript \fB@\fR or \fB*\fR, the substitution operation is
 978 applied to each element in turn. In this case, the \fIstring\fR portion of
 979 \fIword\fR is re-evaluated for each element.
 980 .sp
 981 In the first form, only the first occurrence of \fIpattern\fR is replaced.
 982 .sp
 983 In the second form, each match for \fIpattern\fR is replaced by the specified
 984 \fIstring\fR.
```



```
 985 .sp
 986 The third form restricts the pattern match to the beginning of the
 987 \fIstring\fR.
 988 .sp
 989 The fourth form restricts the pattern match to the end of the \fIstring\fR.
 990 .RE

 992 .sp
 993 .LP
 994 The following parameters are automatically set by the shell:
 995 .sp
 996 .ne 2
 997 .na
 998 \fB\fB#\fR\fR
 999 .ad
1000 .RS 19n
1001 The number of positional parameters in decimal.
1002 .RE

1004 .sp
1005 .ne 2
1006 .na
1007 \fB\fB-\fR\fR
1008 .ad
1009 .RS 19n
1010 Options supplied to the shell on invocation or by the \fBset\fR command.
1011 .RE

1013 .sp
1014 .ne 2
1015 .na
1016 \fB\fB?\fR\fR
1017 .ad
1018 .RS 19n
1019 The decimal value returned by the last executed command.
1020 .RE

1022 .sp
1023 .ne 2
1024 .na
1025 \fB\fB$\fR\fR
1026 .ad
1027 .RS 19n
1028 The process number of this shell.
1029 .RE

1031 .sp
1032 .ne 2
1033 .na
1034 \fB\fB_\fR\fR
1035 .ad
1036 .RS 19n
1037 Initially, the value of \fB_\fR is the absolute pathname of the shell or script
1038 being executed as passed in the environment. It is subsequently assigned the
1039 last argument of the previous command.
1040 .sp
1041 This parameter is not set for commands which are asynchronous. This parameter
1042 is also used to hold the name of the matching \fBMAIL\fR file when checking for
1043 mail.
1044 .RE

1046 .sp
1047 .ne 2
1048 .na
1049 \fB\fB!\fR\fR
1050 .ad
```

```
.RS 19n
The process number of the last background command invoked or the most recent
job put in the background with the \fBbg\fR built-in command.
.RE

.sp
.ne 2
.na
\fB\fB\&.sh.command\fR\fR
.ad
.RS 19n
When processing a \fBDEBUG\fR trap, this variable contains the current command
line that is about to run.
.RE

.sp
.ne 2
.na
\fB\fB\&.sh.edchar\fR\fR
.ad
.RS 19n
This variable contains the value of the keyboard character (or sequence of
characters if the first character is an ESC, \fBASCII 033\fR) that has been
entered when processing a \fBKEYBD\fR trap. If the value is changed as part of
the trap action, then the new value replaces the key (or key sequence) that
caused the trap. See the \fBKey Bindings\fR section of this manual page.
.RE

.sp
.ne 2
.na
\fB\fB\&.sh.edcol\fR\fR
.ad
.RS 19n
The character position of the cursor at the time of the most recent \fBKEYBD\fR
trap.
.RE

.sp
.ne 2
.na
\fB\fB\&.sh.edmode\fR\fR
.ad
.RS 19n
The value is set to ESC when processing a \fBKEYBD\fR trap while in \fBvi\fR
insert mode. Otherwise, \fB\&.sh.edmode\fR is null when processing a
\fBKEYBD\fR trap. See the \fBvi Editing Mode\fR section of this manual page.
.RE

.sp
.ne 2
.na
\fB\fB\&.sh.edtext\fR\fR
.ad
.RS 19n
The characters in the input buffer at the time of the most recent \fBKEYBD\fR
trap. The value is null when not processing a \fBKEYBD\fR trap.
.RE

.sp
.ne 2
.na
\fB\fB\&.sh.file\fR\fR
.ad
.RS 19n
The pathname of the file than contains the current command.
```



```
.RE

.sp
.ne 2
.na
\fB\fB\&.sh.fun\fR\fR
.ad
.RS 19n
The name of the current function that is being executed.
.RE

.sp
.ne 2
.na
\fB\fB\&.sh.match\fR\fR
.ad
.RS 19n
An indexed array which stores the most recent match and sub-pattern matches
after conditional pattern matches that match and after variables expansions
using the operators \fB#\fR, \fB%\fR, or \fB/\fR. The \fB0\fRth element stores
the complete match and the \fIi\fRth element stores the \fIi\fRth sub-match.
The \fB\&.sh.match\fR variable is unset when the variable that has expanded is
assigned a new value.
.RE

.sp
.ne 2
.na
\fB\fB\&.sh.name\fR\fR
.ad
.RS 19n
Set to the name of the variable at the time that a discipline function is
invoked.
.RE

.sp
.ne 2
.na
\fB\fB\&.sh.subscript\fR\fR
.ad
.RS 19n
Set to the name subscript of the variable at the time that a discipline
function is invoked.
.RE

.sp
.ne 2
.na
\fB\fB\&.sh.subshell\fR\fR
.ad
.RS 19n
The current depth for sub-shells and command substitution.
.RE

.sp
.ne 2
.na
\fB\fB\&.sh.value\fR\fR
.ad
.RS 19n
Set to the value of the variable at the time that the set or append discipline
function is invoked.
.RE

.sp
.ne 2
```

```
.na
\fB\fB\&.sh.version\fR\fR
.ad
.RS 19n
Set to a value that identifies the version of this shell.
.RE

.sp
.ne 2
.na
\fB\fBLINENO\fR\fR
.ad
.RS 19n
The current line number within the script or function being executed.
.RE

.sp
.ne 2
.na
\fB\fBOLDPWD\fR\fR
.ad
.RS 19n
The previous working directory set by the \fBcd\fR command.
.RE

.sp
.ne 2
.na
\fB\fBOPTARG\fR\fR
.ad
.RS 19n
The value of the last option argument processed by the \fBgetopts\fR built-in
command.
.RE

.sp
.ne 2
.na
\fB\fBOPTIND\fR\fR
.ad
.RS 19n
The index of the last option argument processed by the \fBgetopts\fR built-in
command.
.RE

.sp
.ne 2
.na
\fB\fBPPID\fR\fR
.ad
.RS 19n
The process number of the parent of the shell.
.RE

.sp
.ne 2
.na
\fB\fBPWD\fR\fR
.ad
.RS 19n
The present working directory set by the \fBcd\fR command.
.RE

.sp
.ne 2
.na
```



```
\fB\fBRANDOM\fR\fR
.ad
.RS 19n
Each time this variable is referenced, a random integer, uniformly distributed
between \fB0\fR and \fB32767\fR, is generated. The sequence of random numbers
can be initialized by assigning a numeric value to \fBRANDOM\fR.
.RE

.sp
.ne 2
.na
\fB\fBREPLY\fR\fR
.ad
.RS 19n
This variable is set by the \fBselect\fR statement and by the \fBread\fR
built-in command when no arguments are supplied.
.RE

.sp
.ne 2
.na
\fB\fBSECONDS\fR\fR
.ad
.RS 19n
Each time this variable is referenced, the number of seconds since shell
invocation is returned. If this variable is assigned a value, then the value
returned upon reference is the value that was assigned plus the number of
seconds since the assignment.
.RE

.sp
.LP
The following variables are used by the shell:
.sp
.ne 2
.na
\fB\fBCDPATH\fR\fR
.ad
.RS 14n
Defines the search path for the \fBcd\fR command.
.RE

.sp
.ne 2
.na
\fB\fBCOLUMNS\fR\fR
.ad
.RS 14n
Defines the width of the edit window for the shell edit modes and for printing
select lists.
.RE

.sp
.ne 2
.na
\fB\fBEDITOR\fR\fR
.ad
.RS 14n
If the \fBVISUAL\fR variable is not set, the value of this variable is checked
for the patterns as described with \fBVISUAL\fR and the corresponding editing
option is turned on.
.sp
See the \fBset\fR command in the \fBSpecial Command\fR section of this manual
page.
.RE
```

```
.sp
.ne 2
.na
\fB\fBENV\fR\fR
.ad
.RS 14n
Performs parameter expansion, command substitution, and arithmetic substitution
on the value to generate the pathname of the script that is executed when the
shell is invoked. This file is typically used for alias and function
definitions. The default value is \fB$HOME/.kshrc\fR.
.sp
See the \fBInvocation\fR section of this manual page.
.sp
\fBENV\fR is not set by the shell.
.RE

.sp
.ne 2
.na
\fB\fBFCEDIT\fR\fR
.ad
.RS 14n
Obsolete name for the default editor name for the \fBhist\fR command.
\fBFCEDIT\fR is not used when \fBHISTEDIT\fR is set.
.sp
The shell specifies a default value to \fBFCEDIT\fR.
.RE

.sp
.ne 2
.na
\fB\fBFIGNORE\fR\fR
.ad
.RS 14n
A pattern that defines the set of file names that is ignored when performing
file name matching.
.RE

.sp
.ne 2
.na
\fB\fBFPATH\fR\fR
.ad
.RS 14n
The search path for function definitions. The directories in this path are
searched for a file with the same name as the function or command when a
function with the \fB-u\fR attribute is referenced and when a command is not
found. If an executable file with the name of that command is found, then it is
read and executed in the current environment. Unlike \fBPATH\fR, the current
directory must be represented explicitly by dot (\fB\&.\fR) rather than by
adjacent colon (\fB:\fR) characters or a beginning or ending colon (\fB:\fR).
.RE

.sp
.ne 2
.na
\fB\fBHISTCMD\fR\fR
.ad
.RS 14n
The number of the current command in the history file.
.RE

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



```
.ad
.RS 14n
The name for the default editor name for the \fBhist\fR command.
.RE

.sp
.ne 2
.na
\fB\fBHISTFILE\fR\fR
.ad
.RS 14n
If this variable is set when the shell is invoked, the value is the pathname of
the file that is used to store the command history. See the \fBCommand
Re-entry\fR section of this manual page.
.RE

.sp
.ne 2
.na
\fB\fBHISTSIZE\fR\fR
.ad
.RS 14n
If this variable is set when the shell is invoked, then the number of
previously entered commands that are accessible by this shell is greater than
or equal to this number. The default is \fB512\fR.
.RE

.sp
.ne 2
.na
\fB\fBHOME\fR\fR
.ad
.RS 14n
The default argument (home directory) for the \fBcd\fR command.
.sp
\fBHOME\fR is not set by the shell. \fBHOME\fR is set by \fBlogin\fR(1).
.RE

.sp
.ne 2
.na
\fB\fBIFS\fR\fR
.ad
.RS 14n
Internal field separators, normally SPACE, TAB, and NEWLINE that are used to
separate the results of command substitution or parameter expansion and to
separate fields with the built-in command read. The first character of the
\fBIFS\fR variable is used to separate arguments for the \fB"$*"\fR
substitution. See the \fBQuoting\fR section of this manual page.
.sp
Each single occurrence of an \fBIFS\fR character in the string to be split,
that is not in the \fBissspace\fR character class, and any adjacent characters
in \fBIFS\fR that are in the \fBissspace\fR character class, delimit a field.
One or more characters in IFS that belong to the \fBissspace\fR character
class, delimit a field. In addition, if the same \fBissspace\fR character
appears consecutively inside \fBIFS\fR, this character is treated as if it were
not in the \fBissspace\fR class, so that if \fBIFS\fR consists of two tab
characters, then two adjacent tab characters delimit a null field.
.sp
The shell specifies a default value to \fBIFS\fR.
.RE

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

```
.ad
.RS 14n
This variable determines the locale category for any category not specifically
selected with a variable starting with \fBLC_\fR or \fBLANG\fR.
.RE

.sp
.ne 2
.na
\fB\fBLC_ALL\fR\fR
.ad
.RS 14n
This variable overrides the value of the \fBLANG\fR variable and any other
\fBLC_\fR variable.
.RE

.sp
.ne 2
.na
\fB\fBLC_COLLATE\fR\fR
.ad
.RS 14n
This variable determines the locale category for character collation
information.
.RE

.sp
.ne 2
.na
\fB\fBLC_CTYPE\fR\fR
.ad
.RS 14n
This variable determines the locale category for character handling functions.
It determines the character classes for pattern matching. See the \fBFile Name
Generation\fR section of this manual page.
.RE

.sp
.ne 2
.na
\fB\fBLC_NUMERIC\fR\fR
.ad
.RS 14n
This variable determines the locale category for the decimal point character.
.RE

.sp
.ne 2
.na
\fB\fBLINES\fR\fR
.ad
.RS 14n
If this variable is set, the value is used to determine the column length for
printing select lists. Select lists prints vertically until about two-thirds of
\fBLINES\fR lines are filled.
.RE

.sp
.ne 2
.na
\fB\fBMAIL\fR\fR
.ad
.RS 14n
If this variable is set to the name of a mail file \fBand\fR the \fBMAILPATH\fR
variable is not set, then the shell informs the user of arrival of mail in the
specified file.
```



```
.sp
\fBMAIL\fR is not set by the shell. On some systems, \fBMAIL\fR is set by
\fBlogin\fR(1).
.RE

.sp
.ne 2
.na
\fB\fBMAILCHECK\fR\fR
.ad
.RS 14n
Specifies how often in seconds the shell checks for changes in the modification
time of any of the files specified by the \fBMAILPATH\fR or \fBMAIL\fR
variables. The default value is \fB600\fR seconds. When the time has elapsed
the shell checks before issuing the next prompt.
.sp
The shell specifies a default value to \fBMAILCHECK\fR.
.RE

.sp
.ne 2
.na
\fB\fBMAILPATH\fR\fR
.ad
.RS 14n
A colon ( \fB:\fR ) separated list of file names. If this variable is set, then
the shell informs the user of any modifications to the specified files that
have occurred within the last \fBMAILCHECK\fR seconds. Each file name can be
followed by a \fB?\fR and a message that is printed. The message undergoes
parameter expansion, command substitution, and arithmetic substitution with the
variable \fB$_\fR defined as the name of the file that has changed. The default
message is \fByou have mail in $_\fR.
.RE

.sp
.ne 2
.na
\fB\fBPATH\fR\fR
.ad
.RS 14n
The search path for commands. Except in \fB\&.profile\fR, users cannot change
\fBPATH\fR if executing under \fBrksh93\fR. See the \fBExecution\fR section of
this manual page.
.sp
The shell specifies a default value to \fBPATH\fR.
.RE

.sp
.ne 2
.na
\fB\fBPS1\fR\fR
.ad
.RS 14n
The value of this variable is expanded for parameter expansion, command
substitution, and arithmetic substitution to define the primary prompt string
which by default is \fB$\fR. The character \fB!\fR in the primary prompt string
is replaced by the command number. Two successive occurrences of \fB!\fR
produces a single \fB!\fR when the prompt string is printed. See the \fBCommand
Re-entry\fR section of this manual page.
.sp
The shell specifies a default value to \fBPS1\fR.
.RE

.sp
.ne 2
.na
```

```
1579 \fB\fBPS2\fR\fR
1580 .ad
1581 .RS 14n
1582 Secondary prompt string, by default, \fB>\fR.
1583 .sp
1584 The shell specifies a default value to \fBPS2\fR.
1585 .RE

1587 .sp
1588 .ne 2
1589 .na
1590 \fB\fBPS3\fR\fR
1591 .ad
1592 .RS 14n
1593 Selection prompt string used within a select loop, by default \fB#?\fR.
1594 .sp
1595 The shell specifies a default value to \fBPS3\fR.
1596 .RE

1598 .sp
1599 .ne 2
1600 .na
1601 \fB\fBPS4\fR\fR
1602 .ad
1603 .RS 14n
1604 The value of this variable is expanded for parameter evaluation, command
1605 substitution, and arithmetic substitution and precedes each line of an
1606 execution trace. By default, \fBPS4\fR is \fB+\fR. When \fBPS4\fR is unset, the
1607 execution trace prompt is also \fB+\fR .
1608 .sp
1609 The shell specifies a default value to \fBPS4\fR.
1610 .RE

1612 .sp
1613 .ne 2
1614 .na
1615 \fB\fBSHELL\fR\fR
1616 .ad
1617 .RS 14n
1618 The pathname of the shell is kept in the environment. At invocation, if the
1619 basename of this variable is \fBrsh\fR, \fBrksh\fR, \fBrksh93\fR, or
1620 \fBkrsh\fR, the shell becomes restricted.
1621 .sp
1622 \fBSHELL\fR is not set by the shell. On some systems, \fBSHELL\fR is set by
1623 \fBlogin\fR(1).
1624 .RE

1626 .sp
1627 .ne 2
1628 .na
1629 \fB\fBTIMEFORMAT\fR\fR
1630 .ad
1631 .RS 14n
1632 The value of this parameter is used as a format string specifying how the
1633 timing information for pipelines prefixed with the \fBtime\fR reserved word
1634 should be displayed. The \fB%\fR character introduces a format sequence that is
1635 expanded to a time value or other information.
1636 .sp
1637 The format sequences and their meanings are as follows.
1638 .sp
1639 .ne 2
1640 .na
1641 \fB\fB%%\fR\fR
1642 .ad
1643 .sp .6
1644 .RS 4n
```



```
1645 A literal \fB%\fR.
1646 .RE

1648 .sp
1649 .ne 2
1650 .na
1651 \fB\fB%[\fIp\fR][l]R\fR\fR
1652 .ad
1653 .sp .6
1654 .RS 4n
1655 The elapsed time in seconds.
1656 .RE

1658 .sp
1659 .ne 2
1660 .na
1661 \fB\fB%[\fIp\fR][l]U\fR\fR
1662 .ad
1663 .sp .6
1664 .RS 4n
1665 The number of CPU seconds spent in user mode.
1666 .RE

1668 .sp
1669 .ne 2
1670 .na
1671 \fB\fB%[\fIp\fR][l]S\fR\fR
1672 .ad
1673 .sp .6
1674 .RS 4n
1675 The number of CPU seconds spent in system mode.
1676 .RE

1678 .sp
1679 .ne 2
1680 .na
1681 \fB\fB%P\fR\fR
1682 .ad
1683 .sp .6
1684 .RS 4n
1685 The CPU percentage, computed as \fB(U + S) / R\fR.
1686 .RE

1688 The braces denote optional portions. The optional \fIp\fR is a digit specifying
1689 the \fIprecision\fR, the number of fractional digits after a decimal point. A
1690 value of \fB0\fR causes no decimal point or fraction to be output. At most
1691 three places after the decimal point can be displayed. Values of \fIp\fR
1692 greater than \fB3\fR are treated as \fB3\fR. If \fIp\fR is not specified, the
1693 value \fB3\fR is used.
1694 .sp
1695 The optional \fBl\fR specifies a longer format, including hours if greater than
1696 zero, minutes, and seconds of the form \fIHHhMMmSS.FFs\fR. The value of \fIp\fR
1697 determines whether or not the fraction is included.
1698 .sp
1699 All other characters are output without change and a trailing NEWLINE is added.
1700 If unset, the default value, \fB$'\enreal\t%2lR\nuser\t%2lU\nsys%2lS'\fR, is
1700 If unset, the default value, \fB$'\nreal\t%2lR\nuser\t%2lU\nsys%2lS'\fR, is
1701 used. If the value is null, no timing information is displayed.
1702 .RE

1704 .sp
1705 .ne 2
1706 .na
1707 \fB\fBTMOUT\fR\fR
1708 .ad
1709 .RS 14n
```

If set to a value greater than zero, \fBTMOUT\fR is the default time-out value for the \fBread\fR built-in command. The \fBselect\fR compound command terminates after \fBTMOUT\fR seconds when input is from a terminal. Otherwise, the shell terminates if a line is not entered within the prescribed number of seconds while reading from a terminal. The shell can be compiled with a maximum bound for this value which cannot be exceeded.
.sp
The shell specifies a default value to \fBTMOUT\fR.
.RE

.sp
.ne 2
.na
\fB\fBVISUAL\fR\fR
.ad
.RS 14n
If the value of this variable matches the pattern \fB*[Vv][Ii]*\fR, then the \fBvi\fR option is turned on. See \fBSpecial Commands\fR. If the value matches the pattern \fB*gmacs*\fR , the \fBgmacs\fR option is turned on. If the value matches the pattern \fB*macs*\fR, then the \fBemacs\fR option is turned on. The value of \fBVISUAL\fR overrides the value of \fBEDITOR\fR.
.RE

.SS "Field Splitting"
.sp
.LP
After parameter expansion and command substitution, the results of substitutions are scanned for the field separator characters (those found in \fBIFS\fR) and split into distinct fields where such characters are found. Explicit null fields (\fB""\fR or \fB\&''\fR) are retained. Implicit null fields, those resulting from parameters that have no values or command substitutions with no output, are removed.
.sp
.LP
If the \fBbraceexpand\fR (\fB-B\fR) option is set, each of the fields resulting from \fBIFS\fR are checked to see if they contain one or more of the brace patterns. Valid brace patterns: \fB{*\fR,\fB*}\fR,
\fB{\fR\fIl1\fR\fB\&..\fR\fIl2\fR\fB}\fR ,
\fB{\fR\fIn1\fR\fB\&..\fR\fIn2\fR\fB}\fR,
\fB{\fR\fIn1\fR\fB\&..\fR\fIn2\fR\fB%\fR\fIfmt\fR\fB}
{\fR\fIn1\fR\fB\&..\fR\fIn2\fR \fB\&..\fR\fIn3\fR\fB}\fR, or
\fB{\fR\fIn1\fR\fB\&..\fR\fIn2\fR \fB\&..\fR\fIn3\fR\fB%\fR\fIfmt\fR\fB}\fR ,
where \fB*\fR represents any character, \fIl1\fR,\fIl2\fR are letters and \fIn1\fR,\fIn2\fR,\fIn3\fR are signed numbers and \fIfmt\fR is a format specified as used by \fBprintf\fR. In each case, fields are created by prepending the characters before the \fB{\fR and appending the characters after the } to each of the strings generated by the characters between the \fB{\fR and \fB}\fR. The resulting fields are checked to see if they have any brace patterns.
.sp
.LP
In the first form, a field is created for each string between \fB{\fR and \fB,,\fR between \fB,\fR and \fB,,\fR and between , and \fB}\fR. The string represented by \fB*\fR can contain embedded matching { and } without quoting. Otherwise, each \fB{\fR and \fB}\fR with \fB*\fR must be quoted.
.sp
.LP
In the second form, \fIl1\fR and \fIl2\fR must both be either upper case or both be lower case characters in the C locale. In this case a field is created for each character from \fIl1\fR through \fIl2\fR.
.sp
.LP
In the remaining forms, a field is created for each number starting at \fIn1\fR. This continues until it reaches \fIn2\fR and increments \fIn1\fR by \fIn3\fR. The cases where \fIn3\fR is not specified behave as if \fIn3\fR were 1 if \fIn1\fR\fB<=\fR\fIn2\fR, and \fB-1\fR otherwise. In forms which specify

\fB%\fR\fIfmt\fR, any format flags, widths and precisions can be specified and \fIfmt\fR can end in any of the specifiers \fBcdiouxX\fR. For example, \fB{a,z}{1..5..3%02d}{b..c}x\fR expands to the 8 fields, \fBa01bx, a01cx, a04bx, a04cx, z01bx, z01cx, z04bx,\fR and \fBz4cx\fR.
.SS "File Name Generation"
.sp
.LP
Following splitting, each field is scanned for the characters \fB*\fR, \fB?\fR, \fB(\fR, and \fB[\fR, unless the \fB-f\fR option has been set. If one of these characters appears, then the word is regarded as a pattern.
.sp
.LP
Each file name component that contains any pattern character is replaced with a lexicographically sorted set of names that matches the pattern from that directory. If no file name is found that matches the pattern, then that component of the file name is left unchanged unless the pattern is prefixed with \fB~(N)\fR in which case it is removed. If \fBFIGNORE\fR is set, then each file name component that matches the pattern defined by the value of \fBFIGNORE\fR is ignored when generating the matching file names. The names \fB\&.\fR and \fB\&..\fR are also ignored. If \fBFIGNORE\fR is not set, the character \fB\&.\fR at the start of each file name component is ignored unless the first character of the pattern corresponding to this component is the character \fB\&.\fR itself. For other uses of pattern matching the \fB/\fR and \fB\&.\fR are not specially treated.
.sp
.ne 2
.na
\fB\fB*\fR\fR
.ad
.RS 11n
Match any string, including the null string. When used for file name expansion, if the \fBglobstar\fR option is on, two adjacent \fB*\fRs by themselves match all files and zero or more directories and subdirectories. If the two adjacent \fB*\fRs are followed by a \fB/\fR, only directories and subdirectories match.
.RE

.sp
.ne 2
.na
\fB\fB?\fR\fR
.ad
.RS 11n
Matches any single character.
.RE

.sp
.ne 2
.na
\fB[\fB\&...\fR]\fR
.ad
.RS 11n
Match any one of the enclosed characters. A pair of characters separated by \fB-\fR matches any character lexically between the pair, inclusive. If the first character following the opening \fB[\fR is a \fB!\fR, any character not enclosed is matched. A \fB-\fR can be included in the character set by putting it as the first or last character. Within \fB[\fR and \fB]\fR, character classes can be specified with the syntax \fB[:\fR\fIclass\fR\fB:]\fR where \fIclass\fR is one of the following classes defined in the \fBANSI-C\fR standard:
.sp
.in +2
.nf
\fIalnum alpha blank cntrl digit graph
lower print punct space upper
word xdigit\fR
.fi

```
.in -2
.sp

\fIword\fR is equivalent to \fIalnum\fR plus the character \fB_\fR. Within
\fB[\fR and \fB]\fR, an equivalence class can be specified with the syntax
\fB[=\fR\fIc\fR\fB=]\fR which matches all characters with the same primary
collation weight (as defined by the current locale) as the character \fIc\fR.
Within \fB[\fR and \fB]\fR, [\fI\&.symbol.\fR] matches the collating symbol
\fIsymbol\fR.
.RE

.sp
.LP
A \fIpattern-list\fR is a list of one or more patterns separated from each
other with an \fB&\fR or \fB|\fR. An \fB&\fR signifies that all patterns must
be matched whereas \fB|\fR requires that only one pattern be matched. Composite
patterns can be formed with one or more of the following sub-patterns:
.sp
.ne 2
.na
\fB\fB?(\fR\fIpattern-list\fR\fB)\fR\fR
.ad
.RS 22n
Optionally matches any one of the specified patterns.
.RE

.sp
.ne 2
.na
\fB\fB*(\fR\fIpattern-list\fR\fB)\fR\fR
.ad
.RS 22n
Matches zero or more occurrences of the specified patterns.
.RE

.sp
.ne 2
.na
\fB\fB+(\fR\fIpattern-list\fR\fB)\fR\fR
.ad
.RS 22n
Matches one or more occurrences of the specified patterns.
.RE

.sp
.ne 2
.na
\fB\fB{\fR\fIn\fR\fB(\fR\fIpattern-list\fR\fB)\fR\fR
.ad
.RS 22n
Matches \fIn\fR occurrences of the specified patterns.
.RE

.sp
.ne 2
.na
\fB\fB{\fR\fIm\fR\fB,\fR\fIn\fR\fB(\fR\fIpattern-list\fR\fB)\fR\fR
.ad
.RS 22n
Matches from \fIm\fR to \fIn\fR occurrences of the specified patterns. If
\fIm\fR is omitted, \fB0\fR is used. If \fIn\fR is omitted at least \fIm\fR
occurrences are matched.
.RE

.sp
.ne 2
```



```
.na
\fB\fB@(\fR\fIpattern-list\fR\fB)\fR\fR
.ad
.RS 22n
Matches exactly one of the specified patterns.
.RE

.sp
.ne 2
.na
\fB\fB!(\fR\fIpattern-list\fR\fB)\fR\fR
.ad
.RS 22n
Matches anything except one of the specified patterns.
.RE

.sp
.LP
By default, each pattern, or sub-pattern matches the longest string possible
consistent with generating the longest overall match. If more than one match is
possible, the one starting closest to the beginning of the string is chosen.
However, for each of the compound patterns a \fB-\fR can be inserted in front
of the \fB(\fR to cause the shortest match to the specified \fIpattern-list\fR
to be used.
.sp
.LP
When \fIpattern-list\fR is contained within parentheses, the backslash
character \fB\e\fR is treated specially even when inside a character class. All
\fBANSI-C\fR character escapes are recognized and match the specified
character. In addition the following escape sequences are recognized:
.sp
.ne 2
.na
\fB\fB\ed\fR\fR
.ad
.RS 7n
Matches any character in the digit class.
.RE

.sp
.ne 2
.na
\fB\fB\eD\fR\fR
.ad
.RS 7n
Matches any character not in the digit class.
.RE

.sp
.ne 2
.na
\fB\fB\es\fR\fR
.ad
.RS 7n
Matches any character in the space class.
.RE

.sp
.ne 2
.na
\fB\fB\eS\fR\fR
.ad
.RS 7n
Matches any character not in the space class.
.RE
```

```
.sp
.ne 2
.na
\fB\fB\ew\fR\fR
.ad
.RS 7n
Matches any character in the word class.
.RE

.sp
.ne 2
.na
\fB\fB\eW\fR\fR
.ad
.RS 7n
Matches any character not in the word class.
.RE

.sp
.LP
A pattern of the form \fB%(\fR\fIpattern-pairs\fR\fB)\fR is a sub-pattern that
can be used to match nested character expressions. Each \fIpattern-pair\fR is a
two character sequence which cannot contain \fB&\fR or \fB|\fR. The first
\fIpattern-pair\fR specifies the starting and ending characters for the match.
Each subsequent \fIpattern-pair\fR represents the beginning and ending
characters of a nested group that is skipped over when counting starting and
ending character matches. The behavior is unspecified when the first character
of a \fIpattern-pair\fR is alphanumeric except for the following:
.sp
.ne 2
.na
\fB\fBD\fR\fR
.ad
.RS 5n
Causes the ending character to terminate the search for this pattern without
finding a match.
.RE

.sp
.ne 2
.na
\fB\fBE\fR\fR
.ad
.RS 5n
Causes the ending character to be interpreted as an escape character.
.RE

.sp
.ne 2
.na
\fB\fBL\fR\fR
.ad
.RS 5n
Causes the ending character to be interpreted as a quote character causing all
characters to be ignored when looking for a match.
.RE

.sp
.ne 2
.na
\fB\fBQ\fR\fR
.ad
.RS 5n
Causes the ending character to be interpreted as a quote character causing all
characters other than any escape character to be ignored when looking for a
match.
```



```
.RE

.sp
.LP
\fB%({}Q"E\e)\fR, matches characters starting at \fB{\fR until the matching
\fB}\fR is found not counting any \fB{\fR or \fB}\fR that is inside a double
quoted string or preceded by the escape character \fB\e\fR\&. Without the
\fB{}\fR this pattern matches any C language string.
.sp
.LP
Each sub-pattern in a composite pattern is numbered, starting at \fB1\fR, by
the location of the \fB(\fR within the pattern. The sequence \fB\e\fR\fIn\fR,
where \fIn\fR is a single digit and \fB\e\fR\fIn\fR comes after the \fIn\fRth.
sub-pattern, matches the same string as the sub-pattern itself.
.sp
.LP
A pattern can contain sub-patterns of the form
\fB~(\fR\fIoptions\fR\fB:\fR\fIpattern-list\fR\fB)\fR, where either
\fIoptions\fR or \fB:\fR\fIpattern-list\fR can be omitted. Unlike the other
compound patterns, these sub-patterns are not counted in the numbered
sub-patterns. If \fIoptions\fR is present, it can consist of one or more of the
following:
.sp
.ne 2
.na
\fB\fB+\fR\fR
.ad
.RS 5n
Enable the following options. This is the default.
.RE

.sp
.ne 2
.na
\fB\fB-\fR\fR
.ad
.RS 5n
Disable the following options.
.RE

.sp
.ne 2
.na
\fB\fBE\fR\fR
.ad
.RS 5n
The remainder of the pattern uses extended regular expression syntax like the
\fBegrep\fR(1) command.
.RE

.sp
.ne 2
.na
\fB\fBF\fR\fR
.ad
.RS 5n
The remainder of the pattern uses \fBfgrep\fR(1) expression syntax.
.RE

.sp
.ne 2
.na
\fB\fBg\fR\fR
.ad
.RS 5n
File the longest match (greedy).
```

```
.sp
This is the default.
.RE

.sp
.ne 2
.na
\fB\fBG\fR\fR
.ad
.RS 5n
The remainder of the pattern uses basic regular expression syntax like the
\fBgrep\fR(1) command.
.RE

.sp
.ne 2
.na
\fB\fBi\fR\fR
.ad
.RS 5n
Treat the match as case insensitive.
.RE

.sp
.ne 2
.na
\fB\fBK\fR\fR
.ad
.RS 5n
The remainder of the pattern uses shell pattern syntax.
.sp
This is the default.
.RE

.sp
.ne 2
.na
\fB\fBl\fR\fR
.ad
.RS 5n
Left anchor the pattern.
.sp
This is the default for \fBK\fR style patterns.
.RE

.sp
.ne 2
.na
\fB\fBN\fR\fR
.ad
.RS 5n
This is ignored. However, when it is the first letter and is used with file
name generation, and no matches occur, the file pattern expands to the empty
string.
.RE

.sp
.ne 2
.na
\fB\fBr\fR\fR
.ad
.RS 5n
Right anchor the pattern.
.sp
This is the default for \fBK\fR style patterns.
.RE
```



```
.sp
.LP
If both \fIoptions\fR and \fB:\fR\fIpattern-list\fR are specified, then the
options apply only to \fIpattern-list\fR. Otherwise, these options remain in
effect until they are disabled by a subsequent \fB~(...)\fR or at the end of
the sub-pattern containing \fB~(...)\fR.
.SS "Quoting"
.sp
.LP
Each of the metacharacters listed in the \fBDefinitions\fR has a special
meaning to the shell.
.sp
.ne 2
.na
\fB\fBg\fR\fR
.ad
.RS 5n
File the longest match (greedy). This is the default.
.RE

.sp
.ne 2
.na
\fB\fBi\fR\fR
.ad
.RS 5n
Treat the match as case insensitive.
.RE

.sp
.LP
If both \fIoptions\fR and \fB:\fR\fIpattern-list\fR are specified, then the
options apply only to \fIpattern-list\fR. Otherwise, the options remain in
effect until they are disabled by a subsequent \fB~(...)\fR or at the end of
the sub-pattern containing \fB~(...)\fR.
.sp
.LP
Each of the metacharacters listed in the \fBDefinitions\fR section of this
manual page has a special meaning to the shell and causes termination of a word
unless quoted. A character can be quoted, that is, made to stand for itself, by
preceding it with a backslash (\fB\e\fR). The pair \fB\e\fRNEWLINE is removed.
All characters enclosed between a pair of single quote marks (\fB\&''\fR) that
is not preceded by a \fB$\fR are quoted. A single quote cannot appear within
the single quotes. A single quoted string preceded by an unquoted \fB$\fR is
processed as an \fBANSI-C\fR string except for the following:
.sp
.ne 2
.na
\fB\fB\e0\fR\fR
.ad
.RS 19n
Causes the remainder of the string to be ignored.
.RE

.sp
.ne 2
.na
\fB\fB\ec\fR\fIx\fR\fR
.ad
.RS 19n
Expands to the character CTRL-x.
.RE

.sp
.ne 2
```

```
.na
\fB\fB\eC\fR[\fB\&.\fR\fIname\fR\fB\&.\fR]\fR
.ad
.RS 19n
Expands to the collating element \fIname\fR.
.RE

.sp
.ne 2
.na
\fB\fB\ee\fR\fR
.ad
.RS 19n
Equivalent to the escape character (\fBASCII\fR 033),
.RE

.sp
.ne 2
.na
\fB\fB\eE\fR\fR
.ad
.RS 19n
Equivalent to the escape character (\fBASCII\fR 033),
.RE

.sp
.LP
Inside double quote marks (\fB""\fR), parameter and command substitution occur
and \fB\e\fR quotes the characters \fB\e\fR, \fB‘\fR, \fB"\fR, and \fB$\fR. A
\fB$\fR in front of a double quoted string is ignored in the \fBC\fR or
\fBPOSIX\fR locale, and might cause the string to be replaced by a locale
specific string otherwise. The meaning of \fB$*\fR and \fB$@\fR is identical
when not quoted or when used as a variable assignment value or as a file name.
However, when used as a command argument, \fB"$*"\fR is equivalent to
\fB"$1\fId\fR$2\fId\fR..."\fR, where \fId\fR is the first character of the IFS
variable, whereas \fB"$@"\fR is equivalent to \fB"$1" "$2" ....\fR Inside grave
quote marks (\fB‘‘\fR), \fB\\fR quotes the characters \fB\e\fR, \fB‘\fR, and
\fB$\fR. If the grave quotes occur within double quotes, then \fB\e\fR also
quotes the character \fB"\fR.
.sp
.LP
The special meaning of reserved words or aliases can be removed by quoting any
character of the reserved word. The recognition of function names or built-in
command names cannot be altered by quoting them.
.SS "Arithmetic Evaluation"
.sp
.LP
The shell performs arithmetic evaluation for arithmetic substitution, to
evaluate an arithmetic command, to evaluate an indexed array subscript, and to
evaluate arguments to the built-in commands \fBshift\fR and \fBlet\fR.
Arithmetic evaluation is also performed on argument operands of the built-in
command printf that correspond to numeric format specifiers in the format
operand. See \fBprintf\fR(1). Evaluations are performed using double precision
floating point arithmetic or long double precision floating point for systems
that provide this data type. Floating point constants follow the \fBANSI-C\fR
programming language floating point conventions. Integer constants follow the
\fBANSI-C\fR programming language integer constant conventions although only
single byte character constants are recognized and character casts are not
recognized. Constants can be of the form \fB[\fR\fIbase#\fR\fB]\fR\fIn\fR where
\fIbase\fR is a decimal number between two and sixty-four representing the
arithmetic base and \fIn\fR is a number in that base. The digits greater than
\fB9\fR are represented by the lower case letters, the upper case letters,
\fB@\fR, and \fB_\fR respectively. For bases less than or equal to \fB36\fR,
upper and lower case characters can be used interchangeably.
.sp
.LP
```



```
An arithmetic expression uses the same syntax, precedence, and associativity of
expression as the C language. All the C language operators that apply to
floating point quantities can be used. In addition, the operator \fB**\fR can
be used for exponentiation. It has higher precedence than multiplication and is
left associative. When the value of an arithmetic variable or subexpression can
be represented as a long integer, all C language integer arithmetic operations
can be performed. Variables can be referenced by name within an arithmetic
expression without using the parameter expansion syntax. When a variable is
referenced, its value is evaluated as an arithmetic expression.
.sp
.LP
Any of the following math library functions that are in the C math library can
be used within an arithmetic expression:
.sp
.in +2
.nf
abs acos acosh asin asinh atan atan2 atanh cbrt
copysign cos cosh erf erfc exp exp2 expm1 fabs
fdim finite floor fma fmax fmod hypot ilogb
int isinf isnan lgamma log log2 logb
nearbyint nextafter nexttoward pow remainder
rint round sin sinh sqrt tan tanh tgamma trunc
.fi
.in -2
.sp

.sp
.LP
An internal representation of a \fIvariable\fR as a double precision floating
point can be specified with the \fB-E [\fR\fIn\fR\fB]\fR or \fB-F
[\fR\fIn\fR\fB]\fR option of the \fBtypeset\fR special built-in command. The
\fB-E\fR option causes the expansion of the value to be represented using
scientific notation when it is expanded. The optional option argument \fIn\fR
defines the number of significant figures. The \fB-F\fR option causes the
expansion to be represented as a floating decimal number when it is expanded.
The optional option argument \fIn\fR defines the number of places after the
decimal point in this case.
.sp
.LP
An internal integer representation of a \fIvariable\fR can be specified with
the \fB-i\fR \fB[\fR\fIn\fR\fB]\fR option of the \fBtypeset\fR special built-in
command. The optional option argument \fIn\fR specifies an arithmetic base to
be used when expanding the variable. If you do not specify an arithmetic base,
base 10 is used.
.sp
.LP
Arithmetic evaluation is performed on the value of each assignment to a
variable with the \fB-E\fR, \fB-F\fR, or \fB-i\fR option. Assigning a floating
point number to a variable whose type is an integer causes the fractional part
to be truncated.
.SS "Prompting"
.sp
.LP
When used interactively, the shell prompts with the value of \fBPS1\fR after
expanding it for parameter expansion, command substitution, and arithmetic
substitution, before reading a command. In addition, each single \fB!\fR in the
prompt is replaced by the command number. A \fB!!\fR is required to place
\fB!\fR in the prompt. If at any time a NEWLINE is typed and further input is
needed to complete a command, then the secondary prompt, that is, the value of
\fBPS2\fR, is issued.
.SS "Conditional Expressions"
.sp
.LP
A \fBconditional expression\fR is used with the \fB[[\fR compound command to
test attributes of files and to compare strings. Field splitting and file name
generation are not performed on the words between \fB[[\fR and \fB]]\fR.
```

```
.sp
.LP
Each expression can be constructed from one or more of the following unary or
binary expressions:
.sp
.ne 2
.na
\fB\fB-a\fR \fIfile\fR\fR
.ad
.RS 21n
True, if \fIfile\fR exists.
.sp
This option is the same as \fB-e\fR. This option is obsolete.
.RE

.sp
.ne 2
.na
\fB\fB-b\fR \fIfile\fR\fR
.ad
.RS 21n
True, if \fIfile\fR exists and is a block special file.
.RE

.sp
.ne 2
.na
\fB\fB-c\fR \fIfile\fR\fR
.ad
.RS 21n
True, if \fIfile\fR exists and is a character special file.
.RE

.sp
.ne 2
.na
\fB\fB-d\fR \fIfile\fR\fR
.ad
.RS 21n
True, if \fIfile\fR exists and is a directory.
.RE

.sp
.ne 2
.na
\fB\fB-e\fR \fIfile\fR\fR
.ad
.RS 21n
True, if \fIfile\fR exists.
.RE

.sp
.ne 2
.na
\fB\fB-f\fR \fIfile\fR\fR
.ad
.RS 21n
True, if \fIfile\fR exists and is an ordinary file.
.RE

.sp
.ne 2
.na
\fB\fB-g\fR \fIfile\fR\fR
.ad
.RS 21n
```



```
True, if \fIfile\fR exists and it has its \fBsetgid\fR bit set.
.RE

.sp
.ne 2
.na
\fB\fB-G\fR \fIfile\fR\fR
.ad
.RS 21n
True, if \fIfile\fR exists and its group matches the effective group id of this
process.
.RE

.sp
.ne 2
.na
\fB\fB-h\fR \fIfile\fR\fR
.ad
.RS 21n
True, if \fIfile\fR exists and is a symbolic link.
.RE

.sp
.ne 2
.na
\fB\fB-k\fR \fIfile\fR\fR
.ad
.RS 21n
True, if \fIfile\fR exists and it has its sticky bit set.
.RE

.sp
.ne 2
.na
\fB\fB-L\fR \fIfile\fR\fR
.ad
.RS 21n
True, if \fIfile\fR exists and is a symbolic link.
.RE

.sp
.ne 2
.na
\fB\fB-n\fR \fIstring\fR\fR
.ad
.RS 21n
True, if length of \fIstring\fR is \fBnon-zero\fR.
.RE

.sp
.ne 2
.na
\fB\fB-N\fR \fIfile\fR\fR
.ad
.RS 21n
True, if \fIfile\fR exists and the modification time is greater than the last
access time.
.RE

.sp
.ne 2
.na
\fB\fB-o\fR \fIoption\fR\fR
.ad
.RS 21n
True, if option named \fIoption\fR is on.
```

```
.RE

.sp
.ne 2
.na
\fB\fB-o\fR \fI?option\fR\fR
.ad
.RS 21n
True, if option named \fIoption\fR is a valid option name.
.RE

.sp
.ne 2
.na
\fB\fB-O\fR \fIfile\fR\fR
.ad
.RS 21n
True, if \fIfile\fR exists and is owned by the effective user id of this
process.
.RE

.sp
.ne 2
.na
\fB\fB-p\fR \fIfile\fR\fR
.ad
.RS 21n
True, if \fIfile\fR exists and is a \fBFIFO\fR special file or a pipe.
.RE

.sp
.ne 2
.na
\fB\fB-r\fR \fIfile\fR\fR
.ad
.RS 21n
True, if \fIfile\fR exists and is readable by current process.
.RE

.sp
.ne 2
.na
\fB\fB-s\fR \fIfile\fR\fR
.ad
.RS 21n
True, if \fIfile\fR exists and has size greater than zero.
.RE

.sp
.ne 2
.na
\fB\fB-S\fR \fIfile\fR\fR
.ad
.RS 21n
True, if \fIfile\fR exists and is a socket.
.RE

.sp
.ne 2
.na
\fB\fB-t\fR \fIfildes\fR\fR
.ad
.RS 21n
True, if file descriptor number \fIfildes\fR is open and associated with a
terminal device.
.RE
```



```
.sp
.ne 2
.na
\fB\fB-u\fR \fIfile\fR\fR
.ad
.RS 21n
True, if \fIfile\fR exists and it has its \fBsetuid\fR bit set.
.RE

.sp
.ne 2
.na
\fB\fB-w\fR \fIfile\fR\fR
.ad
.RS 21n
True, if \fIfile\fR exists and is writable by current process.
.RE

.sp
.ne 2
.na
\fB\fB-x\fR \fIfile\fR\fR
.ad
.RS 21n
True, if \fIfile\fR exists and is executable by current process. If \fIfile\fR
exists and is a directory, then true if the current process has permission to
search in the directory.
.RE

.sp
.ne 2
.na
\fB\fB-z\fR \fIstring\fR\fR
.ad
.RS 21n
True, if length of \fIstring\fR is zero.
.RE

.sp
.ne 2
.na
\fB\fIfile1\fR \fB-ef\fR \fIfile2\fR\fR
.ad
.RS 21n
True, if \fIfile1\fR and \fIfile2\fR exist and refer to the same file.
.RE

.sp
.ne 2
.na
\fB\fIfile1\fR \fB-nt\fR \fIfile2\fR\fR
.ad
.RS 21n
True, if \fIfile1\fR exists and \fIfile2\fR does not, or \fIfile1\fR is newer
than \fIfile2\fR.
.RE

.sp
.ne 2
.na
\fB\fIfile1\fR \fB-ot\fR \fIfile2\fR\fR
.ad
.RS 21n
True, if \fIfile2\fR exists and \fIfile1\fR does not, or \fIfile1\fR is older
than \fIfile2\fR.
```

```
.RE

.sp
.ne 2
.na
\fB\fIstring\fR\fR
.ad
.RS 21n
True, if \fIstring\fR is not null.
.RE

.sp
.ne 2
.na
\fB\fIstring\fR \fB==\fR \fIpattern\fR\fR
.ad
.RS 21n
True, if \fIstring\fR matches \fIpattern\fR. Any part of \fIpattern\fR can be
quoted to cause it to be matched as a string. With a successful match to
\fIpattern\fR, the \fB\&.sh.match\fR array variable contains the match and
sub-pattern matches.
.RE

.sp
.ne 2
.na
\fB\fIstring\fR \fB=\fR \fIpattern\fR\fR
.ad
.RS 21n
Same as \fB==\fR, but is obsolete.
.RE

.sp
.ne 2
.na
\fB\fIstring\fR \fB!=\fR \fIpattern\fR\fR
.ad
.RS 21n
True, if \fIstring\fR does not match \fIpattern\fR. When the \fIstring\fR
matches the \fIpattern\fR the \fB\&.sh.match\fR array variable contains the
match and sub-pattern matches.
.RE

.sp
.ne 2
.na
\fB\fIstring\fR \fB=~\fR \fIere\fR\fR
.ad
.RS 21n
True if \fIstring\fR matches the pattern \fB~(E)\fR\fIere\fR where \fIere\fR is
an extended regular expression.
.RE

.sp
.ne 2
.na
\fB\fIstring1\fR \fB<\fR \fIstring2\fR\fR
.ad
.RS 21n
True, if \fIstring1\fR comes before \fIstring2\fR based on \fBASCII\fR value of
their characters.
.RE

.sp
.ne 2
.na
```



```
\fB\fIstring1\fR \fB>\fR \fIstring2\fR\fR
.ad
.RS 21n
True, if \fIstring1\fR comes after \fIstring2\fR based on \fBASCII\fR value of
their characters.
.RE

.sp
.LP
In each of the following expressions, if \fIfile\fR is of the form
\fB/dev/fd/\fR\fIn\fR, where \fIn\fR is an integer, the test is applied to the
open file whose descriptor number is \fIn\fR. The following obsolete arithmetic
comparisons are supported:
.sp
.ne 2
.na
\fB\fIexp1\fR \fB-eq\fR \fIexp2\fR\fR
.ad
.RS 17n
True, if \fIexp1\fR is equal to \fIexp2\fR.
.RE

.sp
.ne 2
.na
\fB\fIexp1\fR \fB-ge\fR \fIexp2\fR\fR
.ad
.RS 17n
True, if \fIexp1\fR is greater than or equal to \fIexp2\fR.
.RE

.sp
.ne 2
.na
\fB\fIexp1\fR \fB-gt\fR \fIexp2\fR\fR
.ad
.RS 17n
True, if \fIexp1\fR is greater than \fIexp2\fR.
.RE

.sp
.ne 2
.na
\fB\fIexp1\fR \fB-le\fR \fIexp2\fR\fR
.ad
.RS 17n
True, if \fIexp1\fR is less than or equal to \fIexp2\fR.
.RE

.sp
.ne 2
.na
\fB\fIexp1\fR \fB-lt\fR \fIexp2\fR\fR
.ad
.RS 17n
True, if \fIexp1\fR is less than \fIexp2\fR.
.RE

.sp
.ne 2
.na
\fB\fIexp1\fR \fB-ne\fR \fIexp2\fR\fR
.ad
.RS 17n
True, if \fIexp1\fR is not equal to \fIexp2\fR.
.RE
```

```
.sp
.LP
A compound expression can be constructed from these primitives by using any of
the following, listed in decreasing order of precedence:
.sp
.ne 2
.na
\fB\fB(\fR\fIexpression\fR\fB)\fR\fR
.ad
.RS 30n
True, if \fIexpression\fR is true. Used to group expressions.
.RE

.sp
.ne 2
.na
\fB\fB!\fR \fIexpression\fR\fR
.ad
.RS 30n
True, if \fIexpression\fR is false.
.RE

.sp
.ne 2
.na
\fB\fIexpression1\fR \fB&&\fR \fIexpression2\fR\fR
.ad
.RS 30n
True, if \fIexpression1\fR and \fIexpression2\fR are both true.
.RE

.sp
.ne 2
.na
\fB\fIexpression1\fR \fB||\fR \fIexpression2\fR\fR
.ad
.RS 30n
True, if either \fIexpression1\fR or \fIexpression2\fR is true.
.RE

.SS "Input and Output"
.sp
.LP
Before a command is executed, its input and output can be redirected using a
special notation interpreted by the shell. The following can appear anywhere in
a simple command or can precede or follow a command and are \fBnot\fR passed on
to the invoked command. Command substitution, parameter expansion, and
arithmetic substitution occur before \fIword\fR or \fIdigit\fR is used except
as noted in this section. File name generation occurs only if the shell is
interactive and the pattern matches a single file. Field splitting is not
performed.
.sp
.LP
In each of the following redirections, if \fIfile\fR is of the form
\fB/dev/sctp/\fR\fIhost\fR\fB/\fR\fIport\fR,
\fB/dev/tcp/\fR\fIhost\fR\fB/\fR\fIport\fR, or
\fB/dev/udp/\fR\fIhost\fR\fB/\fR\fIport\fR, where \fIhost\fR is a hostname or
host address, and \fIport\fR is a service specified by name or an integer port
number, then the redirection attempts to make a \fBtcp\fR, \fBsctp\fR or
\fBudp\fR connection to the corresponding socket.
.sp
.LP
No intervening space is allowed between the characters of redirection
operators.
.sp
```



```
.ne 2
.na
\fB\fB<\fR\fIword\fR\fR
.ad
.RS 14n
Use file \fIword\fR as standard input (file descriptor 0).
.RE

.sp
.ne 2
.na
\fB\fB>\fR\fIword\fR\fR
.ad
.RS 14n
Use file \fIword\fR as standard output (file descriptor 1). If the file does
not exist then it is created. If the file exists, and the \fBnoclobber\fR
option is on, this causes an error. Otherwise, it is truncated to zero length.
.RE

.sp
.ne 2
.na
\fB\fB>|\fR\fIword\fR\fR
.ad
.RS 14n
Same as \fB>\fR, except that it overrides the \fBnoclobber\fR option.
.RE

.sp
.ne 2
.na
\fB\fB>>\fR\fIword\fR\fR
.ad
.RS 14n
Use file \fIword\fR as standard output. If the file exists, then output is
appended to it (by first seeking to the end-of-file). Otherwise, the file is
created.
.RE

.sp
.ne 2
.na
\fB\fB<>\fR\fIword\fR\fR
.ad
.RS 14n
Open file \fIword\fR for reading and writing as standard input.
.RE

.sp
.ne 2
.na
\fB\fB<<\fR\fB[-]\fR\fIword\fR\fR
.ad
.RS 14n
The shell input is read up to a line that is the same as \fIword\fR after any
quoting has been removed, or to an end-of-file. No parameter substitution,
command substitution, arithmetic substitution or file name generation is
performed on \fIword\fR. The resulting document, called a \fBhere-document\fR,
becomes the standard input. If any character of \fIword\fR is quoted, then no
interpretation is placed upon the characters of the document. Otherwise,
parameter expansion, command substitution, and arithmetic substitution occur,
\fB\e\fRNEWLINE is ignored, and \fB\e\fR must be used to quote the characters
\fB\e\fR, \fB$\fR, \fB`\fR\&. If \fB-\fR is appended to \fB<<\fR, then all
leading tabs are stripped from \fIword\fR and from the document. If \fB#\fR is
appended to \fB<<\fR, then leading SPACEs and TABs are stripped off the first
line of the document and up to an equivalent indentation is stripped from the
```

```
2898 remaining lines and from \fIword\fR. A tab stop is assumed to occur at every 8
2899 columns for the purposes of determining the indentation.
2900 .RE

2902 .sp
2903 .ne 2
2904 .na
2905 \fB\fB<<<\fR\fIword\fR\fR
2906 .ad
2907 .RS 14n
2908 A short form of here document in which \fIword\fR becomes the contents of the
2909 here-document after any parameter expansion, command substitution, and
2910 arithmetic substitution occur.
2911 .RE

2913 .sp
2914 .ne 2
2915 .na
2916 \fB\fB<&\fR\fIdigit\fR\fR
2917 .ad
2918 .RS 14n
2919 The standard input is duplicated from file descriptor \fIdigit\fR, and
2920 similarly for the standard output using \fB>&\fR\fIdigit\fR. See \fBdup\fR(2).
2921 .RE

2923 .sp
2924 .ne 2
2925 .na
2926 \fB\fB<&\fR\fIdigit\fR\fB-\fR\fR
2927 .ad
2928 .RS 14n
2929 The file descriptor specified by \fIdigit\fR is moved to standard input.
2930 Similarly for the standard output using \fB>&\fR\fIdigit\fR\fB-\fR.
2931 .RE

2933 .sp
2934 .ne 2
2935 .na
2936 \fB\fB<&-\fR\fR
2937 .ad
2938 .RS 14n
2939 The standard input is closed. Similarly for the standard output using
2940 \fB>&-\fR.
2941 .RE

2943 .sp
2944 .ne 2
2945 .na
2946 \fB\fB<&p\fR\fR
2947 .ad
2948 .RS 14n
2949 The input from the co-process is moved to standard input.
2950 .RE

2952 .sp
2953 .ne 2
2954 .na
2955 \fB\fB>&p\fR\fR
2956 .ad
2957 .RS 14n
2958 The output to the co-process is moved to standard output.
2959 .RE

2961 .sp
2962 .ne 2
2963 .na
```



```
2964 \fB\fB<#((\fR\fIexpr\fR\fB))\fR\fR
2965 .ad
2966 .RS 14n
2967 Evaluate arithmetic expression \fIexpr\fR and position file descriptor 0 to the
2968 resulting value bytes from the start of the file. The variables \fBCUR\fR and
2969 \fBEOF\fR evaluate to the current offset and end-of-file offset respectively
2970 when evaluating \fIexpr\fR.
2971 .RE

2973 .sp
2974 .ne 2
2975 .na
2976 \fB\fB>#((\fR\fIexpr\fR\fB))\fR\fR
2977 .ad
2978 .RS 14n
2979 The same as \fB<#\fR except applies to file descriptor 1.
2980 .RE

2982 .sp
2983 .ne 2
2984 .na
2985 \fB\fB<#\fR\fIpattern\fR\fR
2986 .ad
2987 .RS 14n
2988 Seek forward to the beginning of the next line containing pattern.
2989 .RE

2991 .sp
2992 .ne 2
2993 .na
2994 \fB\fB<##\fR\fIpattern\fR\fR
2995 .ad
2996 .RS 14n
2997 The same as \fB<#\fR, except that the portion of the file that is skipped is
2998 copied to standard output.
2999 .RE

3001 .sp
3002 .LP
3003 If one of the redirection operators is preceded by a digit, with no intervening
3004 space, then the file descriptor number referred to is that specified by the
3005 digit (instead of the default 0 or 1). If one of the redirection operators
3006 other than \fB>&-\fR and the \fB>#\fR and \fB<#\fR forms, is preceded by
3007 \fB{\fR\fIvarname\fR\fB}\fR with no intervening space, then a file descriptor
3008 number \fB> 10\fR is selected by the shell and stored in the variable
3009 \fIvarname\fR. If \fB>&-\fR or the any of the \fB>#\fR and \fB<#\fR forms is
3010 preceded by \fB{\fR\fIvarname\fR\fB}\fR the value of \fIvarname\fR defines the
3011 file descriptor to close or position. For example:
3012 .sp
3013 .in +2
3014 .nf
3015 \&... 2>&1
3016 .fi
3017 .in -2
3018 .sp

3020 .sp
3021 .LP
3022 means file descriptor 2 is to be opened for writing as a duplicate of file
3023 descriptor 1 and
3024 .sp
3025 .in +2
3026 .nf
3027 exec [\fIn\fR]<\fIfile\fR
3028 .fi
3029 .in -2
```

```
3030 .sp

3032 .sp
3033 .LP
3034 means open \fIfile\fR for reading and store the file descriptor number in
3035 variable \fIn\fR. The order in which redirections are specified is significant.
3036 The shell evaluates each redirection in terms of the (\fIfile_descriptor\fR,
3037 \fIfile\fR) association at the time of evaluation. For example:
3038 .sp
3039 .in +2
3040 .nf
3041 \&... 1>\fIfname\fR 2>&1
3042 .fi
3043 .in -2
3044 .sp

3046 .sp
3047 .LP
3048 first associates file descriptor 1 with file \fIfname\fR. It then associates
3049 file descriptor 2 with the file associated with file descriptor 1, that is,
3050 \fIfname\fR. If the order of redirections were reversed, file descriptor 2
3051 would be associated with the terminal (assuming file descriptor 1 had been) and
3052 then file descriptor 1 would be associated with file \fIfname\fR. If a command
3053 is followed by \fB&\fR and job control is not active, the default standard
3054 input for the command is the empty file \fB/dev/null\fR. Otherwise, the
3055 environment for the execution of a command contains the file descriptors of the
3056 invoking shell as modified by input and output specifications.
3057 .SS "Environment"
3058 .sp
3059 .LP
3060 The \fIenvironment\fR is a list of name-value pairs that is passed to an
3061 executed program in the same way as a normal argument list. See
3062 \fBenviron\fR(5).
3063 .sp
3064 .LP
3065 The names must be \fIidentifiers\fR and the values are character strings. The
3066 shell interacts with the environment in several ways. On invocation, the shell
3067 scans the environment and creates a variable for each name found, giving it the
3068 corresponding value and attributes and marking it \fBexport\fR. Executed
3069 commands inherit the environment. If the user modifies the values of these
3070 variables or creates new ones, using the \fBexport\fR or \fBtypeset\fR \fB-x\fR
3071 commands, they become part of the environment. The environment seen by any
3072 executed command is thus composed of any name-value pairs originally inherited
3073 by the shell, whose values can be modified by the current shell, plus any
3074 additions which must be noted in \fBexport\fR or \fBtypeset\fR \fB-x\fR
3075 commands. The environment for any simple-command or function can be augmented
3076 by prefixing it with one or more variable assignments. A variable assignment
3077 argument is a word of the form \fIidentifier\fR\fB=\fR\fIvalue\fR. Thus:
3078 .sp
3079 .in +2
3080 .nf
3081 TERM=450 cmd args
3082 .fi
3083 .in -2
3084 .sp

3086 .sp
3087 .LP
3088 and
3089 .sp
3090 .in +2
3091 .nf
3092 (export TERM; TERM=450; cmd args)
3093 .fi
3094 .in -2
3095 .sp
```



```
3097 .sp
3098 .LP
3099 are equivalent (as far as the execution of \fIcmd\fR is concerned except for
3100 special built-in commands listed in the \fBBuilt-Ins\fR section, those that are
3101 preceded with a dagger. If the obsolete \fB-k\fR option is set, all variable
3102 assignment arguments are placed in the environment, even if they occur after
3103 the command name.
3104 .sp
3105 .LP
3106 The following example first prints \fBa=b c\fR and then \fBc\fR:
3107 .sp
3108 .in +2
3109 .nf
3110 echo a=b c
3111 set -k
3112 echo a=b c
3113 .fi
3114 .in -2
3115 .sp

3117 .sp
3118 .LP
3119 This feature is intended for use with scripts written for early versions of the
3120 shell and its use in new scripts is strongly discouraged.
3121 .SS "Functions"
3122 .sp
3123 .LP
3124 For historical reasons, there are two ways to define functions, the
3125 \fBname()\fR syntax and the \fBfunction\fR \fBname\fR syntax. These are
3126 described in the \fBCommands\fR section of this manual page.
3127 .sp
3128 .LP
3129 Shell functions are read in and stored internally. Alias names are resolved
3130 when the function is read. Functions are executed like commands with the
3131 arguments passed as positional parameters. See the \fBExecution\fR section of
3132 this manual page for details.
3133 .sp
3134 .LP
3135 Functions defined by the \fBfunction\fR \fBname\fR syntax and called by name
3136 execute in the same process as the caller and share all files and present
3137 working directory with the caller. Traps caught by the caller are reset to
3138 their default action inside the function. A trap condition that is not caught
3139 or ignored by the function causes the function to terminate and the condition
3140 to be passed on to the caller. A trap on \fBEXIT\fR set inside a function is
3141 executed in the environment of the caller after the function completes.
3142 Ordinarily, variables are shared between the calling program and the function.
3143 However, the \fBtypeset\fR special built-in command used within a function
3144 defines local variables whose scope includes the current function. They can be
3145 passed to functions that they call in the variable assignment list that
3146 precedes the call or as arguments passed as name references. Errors within
3147 functions return control to the caller.
3148 .sp
3149 .LP
3150 Functions defined with the \fBname()\fR syntax and functions defined with the
3151 \fBfunction\fR \fBname\fR syntax that are invoked with the \fB\&.\fR special
3152 built-in are executed in the caller's environment and share all variables and
3153 traps with the caller. Errors within these function executions cause the script
3154 that contains them to abort.
3155 .sp
3156 .LP
3157 The special built-in command \fBreturn\fR is used to return from function
3158 calls.
3159 .sp
3160 .LP
3161 Function names can be listed with the \fB-f\fR or \fB+f\fR option of the
```

\fBtypeset\fR special built-in command. The text of functions, when available, is also listed with \fB-f\fR. Functions can be undefined with the \fB-f\fR option of the \fBunset\fR special built-in command.

.LP

Ordinarily, functions are unset when the shell executes a shell script. Functions that need to be defined across separate invocations of the shell should be placed in a directory and the \fBFPATH\fR variable should contain the name of this directory. They can also be specified in the \fBENV\fR file.

.SS "Discipline Functions"

.LP

Each variable can have zero or more discipline functions associated with it. The shell initially understands the discipline names \fBget\fR, \fBset\fR, \fBappend\fR, and \fBunset\fR but on most systems others can be added at run time via the C programming interface extension provided by the \fBbuiltin\fR built-in utility. If the \fBget\fR discipline is defined for a variable, it is invoked whenever the specified variable is referenced. If the variable \fB\&.sh.value\fR is assigned a value inside the discipline function, the referenced variable is evaluated to this value instead. If the \fBset\fR discipline is defined for a variable, it is invoked whenever the specified variable is assigned a value. If the \fBappend\fR discipline is defined for a variable, it is invoked whenever a value is appended to the specified variable. The variable \fB\&.sh.value\fR is specified the value of the variable before invoking the discipline, and the variable is assigned the value of \fB\&.sh.value\fR after the discipline completes. If .\fBsh.value\fR is \fBunset\fR inside the discipline, then that value is unchanged. If the \fBunset\fR discipline is defined for a variable, it is invoked whenever the specified variable is unset. The variable is not unset unless it is unset explicitly from within this discipline function.

.sp

.LP

The variable \fB\&.sh.name\fR contains the name of the variable for which the discipline function is called, \fB\&.sh.subscript\fR is the subscript of the variable, and \fB\&.sh.value\fR contains the value being assigned inside the \fBset\fR discipline function. For the \fBset\fR discipline, changing \fB\&.sh.value\fR changes the value that gets assigned.

.SS "Jobs"

.sp

.LP

If the monitor option of the \fBset\fR command is turned on, an interactive shell associates a job with each pipeline. It keeps a table of current jobs, printed by the \fBjobs\fR command, and assigns them small integer numbers. When a job is started asynchronously with \fB&\fR, the shell prints a line which looks like:

.sp

.in +2

.nf

[1] 1234

.fi

.in -2

.sp

.sp

.LP

indicating that the job which was started asynchronously was job number 1 and had one (top-level) process, whose process id was \fB1234\fR.

.sp

.LP

If you are running a job and wish to stop it, CTRL-z sends a \fBSTOP\fR signal to the current job. The shell normally displays a message that the job has been stopped, and displays another prompt. You can then manipulate the state of this job, putting it in the background with the \fBbg\fR command, or run some other commands and then eventually bring the job back into the foreground with the foreground command \fBfg\fR. A CTRL-z takes effect immediately and is like an interrupt in that pending output and unread input are discarded when it is

typed.

.sp

.LP

A job being run in the background stops if it tries to read from the terminal. Background jobs are normally allowed to produce output, but this can be disabled by giving the command \fBsttytostop\fR. If you set this \fBtty\fR option, then background jobs stop when they try to produce output like they do when they try to read input.

.sp

.LP

There are several ways to refer to jobs in the shell. A job can be referred to by the process id of any process of the job or by one of the following:

.sp

.ne 2

.na

\fB\fB%\fR\fInumber\fR\fR

.ad

.RS 12n

The job with the specified number.

.RE

.sp

.ne 2

.na

\fB\fB%\fR\fIstring\fR\fR

.ad

.RS 12n

Any job whose command line begins with \fIstring\fR.

.RE

.sp

.ne 2

.na

\fB\fB%?\fR\fIstring\fR\fR

.ad

.RS 12n

Any job whose command line contains \fIstring\fR.

.RE

.sp

.ne 2

.na

\fB\fB%%\fR\fR

.ad

.RS 12n

Current job.

.RE

.sp

.ne 2

.na

\fB\fB%+\fR\fR

.ad

.RS 12n

Equivalent to \fB%%\fR.

.RE

.sp

.ne 2

.na

\fB\fB%-\fR\fR

.ad

.RS 12n

Previous job.

.RE

```
.sp
.LP
The shell learns immediately whenever a process changes state. It normally
informs you whenever a job becomes blocked so that no further progress is
possible, but only just before it prints a prompt. This is done so that it does
not otherwise disturb your work. The notify option of the \fBset\fR command
causes the shell to print these job change messages as soon as they occur.
.sp
.LP
When the \fBmonitor\fR option is on, each background job that completes
triggers any trap set for \fBCHLD\fR.
.sp
.LP
When you try to leave the shell while jobs are running or stopped, you are
warned that \fBYou have stopped(running) jobs.\fR You can use the \fBjobs\fR
command to see what they are. If you immediately try to exit again, the shell
does not warn you a second time, and the stopped jobs are terminated. When a
login shell receives a \fBHUP\fR signal, it sends a \fBHUP\fR signal to each
job that has not been disowned with the \fBdisown\fR built-in command.
.SS "Signals"
.sp
.LP
The \fBINT\fR and \fBQUIT\fR signals for an invoked command are ignored if the
command is followed by \fB&\fR and the \fBmonitor\fR option is not active.
Otherwise, signals have the values inherited by the shell from its parent. See
the \fBtrap\fR built-in command.
.SS "Execution"
.sp
.LP
Each time a command is read, the substitutions are carried out. If the command
name matches one of the ones in the \fBSpecial Built-in Commands\fR section of
this manual page, it is executed within the current shell process. Next, the
command name is checked to see if it matches a user defined function. If it
does, the positional parameters are saved and then reset to the arguments of
the function call. A function is also executed in the current shell process.
When the function completes or issues a return, the positional parameter list
is restored. For functions defined with the \fBfunction\fR \fBname\fR syntax,
any trap set on \fBEXIT\fR within the function is executed. The exit value of a
function is the value of the last command executed. If a command name is not a
special built-in command or a user defined function, but it is one of the
built-in commands, it is executed in the current shell process.
.sp
.LP
The shell variable \fBPATH\fR defines the search path for the directory
containing the command. Alternative directory names are separated by a colon
(\fB:\fR). The default path is \fB/bin:/usr/bin:\fR, specifying \fB/bin\fR,
\fB/usr/bin\fR, and the current directory in that order. The current directory
can be specified by two or more adjacent colons, or by a colon at the beginning
or end of the path list. If the command name contains a slash (\fB/\fR), the
search path is not used. Otherwise, each directory in the path is searched for
an executable file of the specified name that is not a directory. If found, and
if the shell determines that there is a built-in version of a command
corresponding to a specified pathname, this built-in is invoked in the current
process. If found, and this directory is also contained in the value of the
\fBFPATH\fR variable, then this file is loaded into the current shell
environment as if it were the argument to the . command except that only preset
aliases are expanded, and a function of the specified name is executed as
described in this manual page. If not found, and the file \fB\&.paths\fR is
found, and this file contains a line of the form \fBFPATH=\fR\fIpath\fR where
\fIpath\fR is an existing directory, and this directory contains a file of the
specified name, then this file is loaded into the current shell environment as
if it were the argument to the \fB\&. special\fR built-in command and a
function of the specified name is executed. Otherwise, if found, a process is
created and an attempt is made to execute the command using \fBexec\fR(2).
.sp
.LP
```



```
When an executable is found, the directory where it is found in is searched for
a file named \fB\&.paths\fR. If this file is found and it contains a line of
the form \fBBUILTIN_LIB=\fR\fIvalue\fR, the library named by \fIvalue\fR is
searched for as if it were an option argument to \fBbuiltin -f\fR, and if it
contains a built-in of the specified name this is executed instead of a command
by this name. Otherwise, if this file is found and it contains a line of the
form \fIname\fR\fB=\fR\fIvalue\fR in the first or second line, then the
environment variable \fIname\fR is modified by prepending the directory
specified by \fIvalue\fR to the directory list. If \fIvalue\fR is not an
absolute directory, then it specifies a directory relative to the directory
that the executable was found. If the environment variable \fIname\fR does not
already exist it is added to the environment list for the specified command.
.sp
.LP
If the file has execute permission but is not an \fBa.out\fR file, it is
assumed to be a file containing shell commands. A separate shell is spawned to
read it. All non-exported variables are removed in this case. If the shell
command file doesn't have read permission, and/or if the \fBsetuid\fR and
\fBsetgid\fR bits are set on the file, then the shell executes an agent whose
job it is to set up the permissions and execute the shell with the shell
command file passed down as an open file. A parenthesized command is executed
in a sub-shell without removing non-exported variables.
.SS "Command Re-entry"
.sp
.LP
The text of the last \fBHISTSIZE\fR (default 512) commands entered from a
terminal device is saved in a history file. The file \fB$HOME/.sh_history\fR is
used if the \fBHISTFILE\fR variable is not set or if the file it names is not
writable. A shell can access the commands of all interactive shells which use
the same named \fBHISTFILE\fR. The built-in command \fBhist\fR is used to list
or edit a portion of this file. The portion of the file to be edited or listed
can be selected by number or by giving the first character or characters of the
command. A single command or range of commands can be specified. If you do not
specify an editor program as an argument to \fBhist\fR then the value of the
variable \fBHISTEDIT\fR is used. If \fBHISTEDIT\fR is unset, the obsolete
variable \fBFCEDIT\fR is used. If \fBFCEDIT\fR is not defined, then
\fB/bin/ed\fR is used. The edited commands are printed and executed again upon
leaving the editor unless you quit without writing. The \fB-s\fR option (and in
obsolete versions, the editor name \fB-\fR) is used to skip the editing phase
and to re-execute the command. In this case a substitution parameter of the
form \fIold\fR\fB=\fR\fInew\fRcan be used to modify the command before
execution. For example, with the preset alias \fBr\fR, which is aliased to
\fB\&'hist -s'\fR, typing \fB'r bad=good c'\fR re-executes the most recent
command which starts with the letter \fBc\fR, replacing the first occurrence of
the string bad with the string good.
.SS "Inline Editing Options"
.sp
.LP
Normally, each command line entered from a terminal device is simply typed
followed by a NEWLINE (RETURN or LINE FEED). If either the \fBemacs\fR,
\fBgmacs\fR, or \fBvi\fR option is active, the user can edit the command line.
To be in either of these edit modes set the corresponding option. An editing
option is automatically selected each time the \fBVISUAL\fR or \fBEDITOR\fR
variable is assigned a value ending in either of these option names.
.sp
.LP
The editing features require that the user's terminal accept RETURN as carriage
return without line feed and that a SPACE must overwrite the current character
on the screen.
.sp
.LP
Unless the \fBmultiline\fR option is on, the editing modes implement a concept
where the user is looking through a window at the current line. The window
width is the value of \fBCOLUMNS\fR if it is defined, otherwise \fB80\fR. If
the window width is too small to display the prompt and leave at least 8
columns to enter input, the prompt is truncated from the left. If the line is
```

longer than the window width minus two, a mark is displayed at the end of the window to notify the user. As the cursor moves and reaches the window boundaries the window is centered about the cursor. The mark is a \fB>\fR (\fB<, *\fR) if the line extends on the right , left, or both sides of the window.
.sp
.LP
The search commands in each edit mode provide access to the history file. Only strings are matched, not patterns, although a leading \fB^\fR in the string restricts the match to begin at the first character in the line.
.sp
.LP
Each of the edit modes has an operation to list the files or commands that match a partially entered word. When applied to the first word on the line, or the first word after a \fB;\fR, \fB|\fR, \fB&\fR, or \fB(\fR, and the word does not begin with \fB~\fR or contain a \fB/\fR, the list of aliases, functions, and executable commands defined by the \fBPATH\fR variable that could match the partial word is displayed. Otherwise, the list of files that match the specified word is displayed. If the partially entered word does not contain any file expansion characters, a \fB*\fR is appended before generating these lists. After displaying the generated list, the input line is redrawn. These operations are called command name listing and file name listing, respectively. There are additional operations, referred to as command name completion and file name completion, which compute the list of matching commands or files, but instead of printing the list, replace the current word with a complete or partial match. For file name completion, if the match is unique, a \fB/\fR is appended if the file is a directory and a space is appended if the file is not a directory. Otherwise, the longest common prefix for all the matching files replaces the word. For command name completion, only the portion of the file names after the last \fB/\fR are used to find the longest command prefix. If only a single name matches this prefix, then the word is replaced with the command name followed by a space. When using a \fBTAB\fR for completion that does not yield a unique match, a subsequent TAB provides a numbered list of matching alternatives. A specific selection can be made by entering the selection number followed by a TAB.
.SS "Key Bindings"
.sp
.LP
The \fBKEYBD\fR trap can be used to intercept keys as they are typed and change the characters that are actually seen by the shell. This trap is executed after each character (or sequence of characters when the first character is ESC) is entered while reading from a terminal.
.sp
.LP
The variable \fB\&.sh.edchar\fR contains the character or character sequence which generated the trap. Changing the value of \fB\&.sh.edchar\fR in the trap action causes the shell to behave as if the new value were entered from the keyboard rather than the original value. The variable \fB\&.sh.edcol\fR is set to the input column number of the cursor at the time of the input. The variable \fB\&.sh.edmode\fR is set to \fBESC\fR when in \fBvi\fR insert mode and is null otherwise. By prepending \fB${.sh.editmode}\fR to a value assigned to \fB\&.sh.edchar\fR it causes the shell to change to control mode if it is not already in this mode.
.sp
.LP
This trap is not invoked for characters entered as arguments to editing directives, or while reading input for a character search.
.SS "\fBemacs\fR Editing Mode"
.sp
.LP
This mode is entered by enabling either the \fBemacs\fR or \fBgmacs\fR option. The only difference between these two modes is the way they handle \fB^T\fR. To edit, the user moves the cursor to the point needing correction and then inserts or deletes characters or words as needed. All the editing commands are control characters or escape sequences. The notation for control characters is caret (\fB^\fR) followed by the character.

.sp
.LP
For example, \fB^F\fR is the notation for CTRL/F. This is entered by depressing \fBf\fR while holding down the CTRL (control) key. The SHIFT key is not depressed. (The notation \fB^?\fR indicates the DEL (delete) key.)
.sp
.LP
The notation for escape sequences is \fBM-\fR followed by a character. For example, \fBM-f\fR (pronounced \fBMeta f\fR) is entered by depressing ESC (\fBASCII 033\fR) followed by \fBf\fR. \fBM-F\fR is the notation for ESC followed by \fBF\fR.
.sp
.LP
All edit commands operate from any place on the line, not just at the beginning. The RETURN or the LINE FEED key is not entered after edit commands except when noted.
.sp
.ne 2
.na
\fB\fB^F\fR\fR
.ad
.RS 13n
Move the cursor forward (right) one character.
.RE

.sp
.ne 2
.na
\fB\fBM-[C\fR\fR
.ad
.RS 13n
Move the cursor forward (right) one character.
.RE

.sp
.ne 2
.na
\fB\fBM-f\fR\fR
.ad
.RS 13n
Move the cursor forward one word. The \fBemacs\fR editor’s idea of a word is a string of characters consisting of only letters, digits and underscores.
.RE

.sp
.ne 2
.na
\fB\fB^B\fR\fR
.ad
.RS 13n
Move the cursor backward (left) one character.
.RE

.sp
.ne 2
.na
\fB\fBM-[D\fR\fR
.ad
.RS 13n
Move the cursor backward (left) one character.
.RE

.sp
.ne 2
.na
\fB\fBM-b\fR\fR

```
.ad
.RS 13n
Move the cursor backward one word.
.RE

.sp
.ne 2
.na
\fB\fB^A\fR\fR
.ad
.RS 13n
Move the cursor to the beginning of the line.
.RE

.sp
.ne 2
.na
\fB\fBM-[H\fR\fR
.ad
.RS 13n
Move the cursor to the beginning of the line.
.RE

.sp
.ne 2
.na
\fB\fB^E\fR\fR
.ad
.RS 13n
Move the cursor to the end of the line.
.RE

.sp
.ne 2
.na
\fB\fBM-[Y\fR\fR
.ad
.RS 13n
Move the cursor to the end of line.
.RE

.sp
.ne 2
.na
\fB\fB^]\fR\fIchar\fR\fR
.ad
.RS 13n
Move the cursor forward to the character \fIchar\fR on the current line.
.RE

.sp
.ne 2
.na
\fB\fBM-^]\fR\fIchar\fR\fR
.ad
.RS 13n
Move the cursor backwards to the character \fIchar\fR on the current line.
.RE

.sp
.ne 2
.na
\fB\fB^X^X\fR\fR
.ad
.RS 13n
Interchange the cursor and the mark.
```



```
.RE

.sp
.ne 2
.na
\fB\fIerase\fR\fR
.ad
.RS 13n
Delete the previous character. The user-defined erase character is defined by
the \fBstty\fR(1) command, and is usually \fB^H\fR or \fB#\fR.
.RE

.sp
.ne 2
.na
\fB\fIlnext\fR\fR
.ad
.RS 13n
Removes the next character’s editing features. The user-defined literal next
character is defined by the \fBstty\fR(1) command, or is \fB^V\fR if not
defined.
.RE

.sp
.ne 2
.na
\fB\fB^D\fR\fR
.ad
.RS 13n
Delete the current character.
.RE

.sp
.ne 2
.na
\fB\fBM-d\fR\fR
.ad
.RS 13n
Delete the current word.
.RE

.sp
.ne 2
.na
\fB\fBM-^H\fR\fR
.ad
.RS 13n
MetaBACKSPACE. Delete the previous word.
.RE

.sp
.ne 2
.na
\fB\fBM-h\fR\fR
.ad
.RS 13n
Delete the previous word.
.RE

.sp
.ne 2
.na
\fB\fBM-^?\fR\fR
.ad
.RS 13n
MetaDEL. Delete the previous word. If your interrupt character is \fB^?\fR
```

(DEL, the default), this command does not work.
.RE

.sp
.ne 2
.na
\fB\fB^T\fR\fR
.ad
.RS 13n
Transpose the current character with the previous character, and advance the cursor in \fBemacs\fR mode. Transpose two previous characters in \fBgmacs\fR mode.
.RE

.sp
.ne 2
.na
\fB\fB^C\fR\fR
.ad
.RS 13n
Capitalize the current character.
.RE

.sp
.ne 2
.na
\fB\fBM-c\fR\fR
.ad
.RS 13n
Capitalize the current word.
.RE

.sp
.ne 2
.na
\fB\fBM-l\fR\fR
.ad
.RS 13n
Change the current word to lower case.
.RE

.sp
.ne 2
.na
\fB\fB^K\fR\fR
.ad
.RS 13n
Delete from the cursor to the end of the line. If preceded by a numerical parameter whose value is less than the current cursor position, delete from specified position up to the cursor. If preceded by a numerical parameter whose value is greater than the current cursor position, then delete from cursor up to specified cursor position.
.RE

.sp
.ne 2
.na
\fB\fB^W\fR\fR
.ad
.RS 13n
Kill from the cursor to the mark.
.RE

.sp
.ne 2
.na

\fB\fBM-p\fR\fR
.ad
.RS 13n
Push the region from the cursor to the mark on the stack.
.RE

.sp
.ne 2
.na
\fB\fIkill\fR\fR
.ad
.RS 13n
Kill the entire current line. The user-defined kill character is defined by the \fBstty\fR(1) command, usually a \fB^G\fR or \fB@\fR. If two kill characters are entered in succession, all kill characters from then on cause a line feed. This is useful when using paper terminals.
.RE

.sp
.ne 2
.na
\fB\fB^Y\fR\fR
.ad
.RS 13n
Restore the last item removed from line. Yank the item back to the line.
.RE

.sp
.ne 2
.na
\fB\fB^L\fR\fR
.ad
.RS 13n
Line feed and print the current line.
.RE

.sp
.ne 2
.na
\fB\fBM-^L\fR\fR
.ad
.RS 13n
Clear the screen.
.RE

.sp
.ne 2
.na
\fB\fB^@\fR\fR
.ad
.RS 13n
Null character. Set mark.
.RE

.sp
.ne 2
.na
\fB\fBM-\fR\fIspace\fR\fR
.ad
.RS 13n
MetaSPACE. Set the mark.
.RE

.sp
.ne 2
.na

```
\fB\fB^J\fR\fR
.ad
.RS 13n
New line. Execute the current line.
.RE

.sp
.ne 2
.na
\fB\fB^M\fR\fR
.ad
.RS 13n
Return. Execute the current line.
.RE

.sp
.ne 2
.na
\fB\fBEOF\fR\fR
.ad
.RS 13n
End-of-file character, normally \fB^D\fR, is processed as an \fBend-of-file\fR
only if the current line is null.
.RE

.sp
.ne 2
.na
\fB\fB^P\fR\fR
.ad
.RS 13n
Fetch the previous command. Each time \fB^P\fR is entered the previous command
back in time is accessed. Moves back one line when it is not on the first line
of a multi-line command.
.RE

.sp
.ne 2
.na
\fB\fBM-[A\fR\fR
.ad
.RS 13n
Equivalent to \fB^P\fR.
.RE

.sp
.ne 2
.na
\fB\fBM-<\fR\fR
.ad
.RS 13n
Fetch the least recent (oldest) history line.
.RE

.sp
.ne 2
.na
\fB\fBM->\fR\fR
.ad
.RS 13n
Fetch the most recent (youngest) history line.
.RE

.sp
.ne 2
.na
```



```
\fB\fB^N\fR\fR
.ad
.RS 13n
Fetch the next command line. Each time \fB^N\fR is entered the next command
line forward in time is accessed.
.RE

.sp
.ne 2
.na
\fB\fBM-[B\fR\fR
.ad
.RS 13n
Equivalent to \fB^N\fR.
.RE

.sp
.ne 2
.na
\fB\fB^R\fR\fIstring\fR\fR
.ad
.RS 13n
Reverse search history for a previous command line containing \fIstring\fR. If
a parameter of zero is specified, the search is forward. \fIstring\fR is
terminated by a RETURN or NEWLINE. If string is preceded by a \fB^\fR, the
matched line must begin with \fIstring\fR. If \fIstring\fR is omitted, then the
next command line containing the most recent \fIstring\fR is accessed. In this
case a parameter of zero reverses the direction of the search.
.RE

.sp
.ne 2
.na
\fB\fB^O\fR\fR
.ad
.RS 13n
Operate. Execute the current line and fetch the next line relative to current
line from the history file.
.RE

.sp
.ne 2
.na
\fB\fBM-\fR\fIdigits\fR\fR
.ad
.RS 13n
Escape. Define numeric parameter. The digits are taken as a parameter to the
next command. The commands that accept a parameter are: \fB^F\fR, \fB^B\fR,
\fBERASE\fR, \fB^C\fR, \fB^D\fR, \fB^K\fR, \fB^R\fR, \fB^P\fR, \fB^N\fR,
\fB^]\fR, \fBM-.\fR, \fBM-\fR, \fBM-^]\fR, \fBM-_\fR, \fBM-=\fR, \fBM-b\fR,
\fBM-c\fR, \fBM-d\fR, \fBM-f\fR, \fBM-h\fR, \fBM-l\fR, and \fBM-^H\fR.
.RE

.sp
.ne 2
.na
\fB\fBM-\fR\fIletter\fR\fR
.ad
.RS 13n
Soft-key. Search the alias list for an alias by the name \fIletter\fR. If an
alias of \fIletter\fR is defined, insert its value on the input queue.
\fIletter\fR must not be one of the metafunctions in this section.
.RE

.sp
.ne 2
```

```
.na
\fB\fBM-[\fR\fIletter\fR\fR
.ad
.RS 13n
Soft key. Search the alias list for an alias by the name \fIletter\fR. If an
alias of this name is defined, insert its value on the input queue. This can be
used to program function keys on many terminals.
.RE

.sp
.ne 2
.na
\fB\fBM-.\fR\fR
.ad
.RS 13n
The last word of the previous command is inserted on the line. If preceded by a
numeric parameter, the value of this parameter determines which word to insert
rather than the last word.
.RE

.sp
.ne 2
.na
\fB\fBM-_\fR\fR
.ad
.RS 13n
Same as \fBM-.\fR.
.RE

.sp
.ne 2
.na
\fB\fBM-*\fR\fR
.ad
.RS 13n
Attempt filename generation on the current word. As asterisk is appended if the
word does not match any file or contain any special pattern characters.
.RE

.sp
.ne 2
.na
\fB\fBM-\fRESC\fR
.ad
.RS 13n
Command or file name completion as described in this manual page.
.RE

.sp
.ne 2
.na
\fB\fB^I\fRTAB\fR
.ad
.RS 13n
Attempts command or file name completion as described in this manual page. If a
partial completion occurs, repeating this behaves as if \fBM-=\fR were entered.
If no match is found or entered after SPACE, a TAB is inserted.
.RE

.sp
.ne 2
.na
\fB\fBM-=\fR\fR
.ad
.RS 13n
If not preceded by a numeric parameter, generates the list of matching commands
```



```
or file names as described in this manual page. Otherwise, the word under the
cursor is replaced by the item corresponding to the value of the numeric
parameter from the most recently generated command or file list. If the cursor
is not on a word, the word is inserted instead.
.RE

.sp
.ne 2
.na
\fB\fB^U\fR\fR
.ad
.RS 13n
Multiply parameter of next command by \fB4\fR.
.RE

.sp
.ne 2
.na
\fB\fB\e\fR\fR
.ad
.RS 13n
Escape the next character. Editing characters, the user's erase, kill and
interrupt (normally \fB^?\fR) characters can be entered in a command line or in
a search string if preceded by a \fB\e\fR\&. The \fB\e\fR removes the next
character's editing features, if any.
.RE

.sp
.ne 2
.na
\fB\fBM-^V\fR\fR
.ad
.RS 13n
Display the version of the shell.
.RE

.sp
.ne 2
.na
\fB\fBM-#\fR\fR
.ad
.RS 13n
If the line does not begin with a \fB#\fR, a \fB#\fR is inserted at the
beginning of the line and after each NEWLINE, and the line is entered. This
causes a comment to be inserted in the history file. If the line begins with a
\fB#\fR, the \fB#\fR is deleted and one \fB#\fR after each NEWLINE is also
deleted.
.RE

.SS "\fBvi\fR Editing Mode"
.sp
.LP
There are two typing modes. Initially, when you enter a command you are in the
input mode. To edit, the user enters control mode by typing ESC (033) and moves
the cursor to the point needing correction and then inserts or deletes
characters or words as needed. Most control commands accept an optional repeat
\fIcount\fR prior to the command.
.sp
.LP
When in vi mode on most systems, canonical processing is initially enabled and
the command is echoed again if the speed is 1200 baud or greater and it
contains any control characters or less than one second has elapsed since the
prompt was printed. The ESC character terminates canonical processing for the
remainder of the command and the user can then modify the command line. This
scheme has the advantages of canonical processing with the type-ahead echoing
of raw mode.
```

```
4086 .sp
4087 .LP
4088 If the option \fBviraw\fR is also set, the terminal is always have canonical
4089 processing disabled. This mode is implicit for systems that do not support two
4090 alternate end of line delimiters, and might be helpful for certain terminals.
4091 .SS "Input Edit Commands"
4092 .sp
4093 .LP
4094 By default the editor is in input mode.
4095 .sp
4096 .LP
4097 The following input edit commands are supported:
4098 .sp
4099 .ne 2
4100 .na
4101 \fBERASE\fR
4102 .ad
4103 .RS 10n
4104 User defined erase character as defined by the \fBstty\fR command, usually
4105 \fB^H\fR or \fB#\fR. Delete previous character.
4106 .RE

4108 .sp
4109 .ne 2
4110 .na
4111 \fB\fB^W\fR\fR
4112 .ad
4113 .RS 10n
4114 Delete the previous blank separated word. On some systems the \fBviraw\fR
4115 option might be required for this to work.
4116 .RE

4118 .sp
4119 .ne 2
4120 .na
4121 \fBEOF\fR
4122 .ad
4123 .RS 10n
4124 As the first character of the line causes the shell to terminate unless the
4125 \fBignoreeof\fR option is set. Otherwise this character is ignored.
4126 .RE

4128 .sp
4129 .ne 2
4130 .na
4131 \fB\fIlnext\fR\fR
4132 .ad
4133 .RS 10n
4134 User defined literal next character as defined by the \fBstty\fR(1) or \fB^V\fR
4135 if not defined. Removes the next character's editing features, if any. On some
4136 systems the \fBviraw\fR option might be required for this to work.
4137 .RE

4139 .sp
4140 .ne 2
4141 .na
4142 \fB\fB\e\fR\fR
4143 .ad
4144 .RS 10n
4145 Escape the next ERASE or KILL character.
4146 .RE

4148 .sp
4149 .ne 2
4150 .na
4151 \fB\fB^I\fR TAB\fR
```



```
4152 .ad
4153 .RS 10n
4154 Attempts command or file name completion as described in this manual page and
4155 returns to input mode. If a partial completion occurs, repeating this behaves
4156 as if \fB=\fR were entered from control mode. If no match is found or entered
4157 after SPACE, a TAB is inserted.
4158 .RE

4160 .SS "Motion Edit Commands"
4161 .sp
4162 .LP
4163 The motion edit commands move the cursor.
4164 .sp
4165 .LP
4166 The following motion edit commands are supported:
4167 .sp
4168 .ne 2
4169 .na
4170 \fB\fB[\fR\fIcount\fR\fB]l\fR\fR
4171 .ad
4172 .RS 13n
4173 Move the cursor forward (right) one character.
4174 .RE

4176 .sp
4177 .ne 2
4178 .na
4179 \fB\fB[\fR\fIcount\fR\fB][C\fR\fR
4180 .ad
4181 .RS 13n
4182 Move the cursor forward (right) one character.
4183 .RE

4185 .sp
4186 .ne 2
4187 .na
4188 \fB\fB[\fR\fIcount\fR\fB]w\fR\fR
4189 .ad
4190 .RS 13n
4191 Move the cursor forward one alphanumeric word.
4192 .RE

4194 .sp
4195 .ne 2
4196 .na
4197 \fB\fB[\fR\fIcount\fR\fB]W\fR\fR
4198 .ad
4199 .RS 13n
4200 Move the cursor to the beginning of the next word that follows a blank.
4201 .RE

4203 .sp
4204 .ne 2
4205 .na
4206 \fB\fB[\fR\fIcount\fR\fB]e\fR\fR
4207 .ad
4208 .RS 13n
4209 Move the cursor to the end of the word.
4210 .RE

4212 .sp
4213 .ne 2
4214 .na
4215 \fB\fB[\fR\fIcount\fR\fB]E\fR\fR
4216 .ad
4217 .RS 13n
```

```
4218 Move the cursor to the end of the current blank delimited word.
4219 .RE

4221 .sp
4222 .ne 2
4223 .na
4224 \fB\fB[\fR\fIcount\fR\fB]h\fR\fR
4225 .ad
4226 .RS 13n
4227 Move the cursor backward (left) one character.
4228 .RE

4230 .sp
4231 .ne 2
4232 .na
4233 \fB\fB[\fR\fIcount\fR\fB][D\fR\fR
4234 .ad
4235 .RS 13n
4236 Move the cursor backward (left) one character.
4237 .RE

4239 .sp
4240 .ne 2
4241 .na
4242 \fB\fB[\fR\fIcount\fR\fB]b\fR\fR
4243 .ad
4244 .RS 13n
4245 Move the cursor backward one word.
4246 .RE

4248 .sp
4249 .ne 2
4250 .na
4251 \fB\fB[\fR\fIcount\fR\fB]B\fR\fR
4252 .ad
4253 .RS 13n
4254 Move the cursor to the preceding blank separated word.
4255 .RE

4257 .sp
4258 .ne 2
4259 .na
4260 \fB\fB[\fR\fIcount\fR\fB]|\fR\fR
4261 .ad
4262 .RS 13n
4263 Move the cursor to column \fIcount\fR.
4264 .RE

4266 .sp
4267 .ne 2
4268 .na
4269 \fB\fB[\fR\fIcount\fR\fB]f\fR\fIc\fR\fR
4270 .ad
4271 .RS 13n
4272 Find the next character \fIc\fR in the current line.
4273 .RE

4275 .sp
4276 .ne 2
4277 .na
4278 \fB\fB[\fR\fIcount\fR\fB]F\fR\fIc\fR\fR
4279 .ad
4280 .RS 13n
4281 Find the previous character \fIc\fR in the current line.
4282 .RE
```



```
4284 .sp
4285 .ne 2
4286 .na
4287 \fB\fB[\fR\fIcount\fR\fB]t\fR\fIC\fR\fR
4288 .ad
4289 .RS 13n
4290 Equivalent to \fBf\fR followed by \fBh\fR.
4291 .RE

4293 .sp
4294 .ne 2
4295 .na
4296 \fB\fB[\fR\fIcount\fR\fB]T\fR\fIc\fR\fR
4297 .ad
4298 .RS 13n
4299 Equivalent to \fBF\fR followed by \fBl\fR.
4300 .RE

4302 .sp
4303 .ne 2
4304 .na
4305 \fB\fB[\fR\fIcount\fR\fB];\fR\fR
4306 .ad
4307 .RS 13n
4308 Repeat \fIcount\fR times the last single character find command: \fBf\fR,
4309 \fBF\fR, \fBt,\fR or \fBT\fR.
4310 .RE

4312 .sp
4313 .ne 2
4314 .na
4315 \fB\fB[\fR\fIcount\fR\fB],\fR\fR
4316 .ad
4317 .RS 13n
4318 Reverse the last single character find command \fIcount\fR times.
4319 .RE

4321 .sp
4322 .ne 2
4323 .na
4324 \fB\fB0\fR\fR
4325 .ad
4326 .RS 13n
4327 Move the cursor to the start of line.
4328 .RE

4330 .sp
4331 .ne 2
4332 .na
4333 \fB\fB^\fR\fR
4334 .ad
4335 .RS 13n
4336 Move the cursor to start of line.
4337 .RE

4339 .sp
4340 .ne 2
4341 .na
4342 \fB\fB[H\fR\fR
4343 .ad
4344 .RS 13n
4345 Move the cursor to the first non-blank character in the line.
4346 .RE

4348 .sp
4349 .ne 2
```

```
.na
\fB\fB$\fR\fR
.ad
.RS 13n
Move the cursor to the end of the line.
.RE

.sp
.ne 2
.na
\fB\fB[Y\fR\fR
.ad
.RS 13n
Move the cursor to the end of the line.
.RE

.sp
.ne 2
.na
\fB\fB%\fR\fR
.ad
.RS 13n
Moves to balancing \fB(\fR, \fB)\fR, \fB{\fR, \fB}\fR, \fB[\fR, or \fB]\fR. If
cursor is not on one of the characters described in this section, the remainder
of the line is searched for the first occurrence of one of the characters
first.
.RE

.SS "Search Edit Commands"
.sp
.LP
The search edit commands access your command history.
.sp
.LP
The following search edit commands are supported:
.sp
.ne 2
.na
\fB\fB[\fR\fIcount\fR\fB]k\fR\fR
.ad
.RS 13n
Fetch the previous command. Each time \fBk\fR is entered, the previous command
back in time is accessed.
.RE

.sp
.ne 2
.na
\fB\fB[\fR\fIcount\fR\fB]-\fR\fR
.ad
.RS 13n
Fetch the previous command. Each time \fBk\fR is entered, the previous command
back in time is accessed.
.sp
Equivalent to \fBk\fR.
.RE

.sp
.ne 2
.na
\fB\fB[\fR\fIcount\fR\fB][A\fR\fR
.ad
.RS 13n
Fetch the previous command. Each time \fBk\fR is entered, the previous command
back in time is accessed.
.sp
```



```
Equivalent to \fBk\fR.
.RE

.sp
.ne 2
.na
\fB\fB[\fR\fIcount\fR\fB]j\fR\fR
.ad
.RS 13n
Fetch the next command. Each time \fBj\fR is entered, the next command forward
in time is accessed.
.RE

.sp
.ne 2
.na
\fB\fB[\fR\fIcount\fR\fB]+\fR\fR
.ad
.RS 13n
Fetch the next command. Each time \fBj\fR is entered, the next command forward
in time is accessed.
.sp
Equivalent to \fBj\fR.
.RE

.sp
.ne 2
.na
\fB\fB[\fR\fIcount\fR\fB][B\fR\fR
.ad
.RS 13n
Fetch the next command. Each time \fBj\fR is entered, the next command forward
in time is accessed.
.sp
Equivalent to \fBj\fR.
.RE

.sp
.ne 2
.na
\fB\fB[\fR\fIcount\fR\fB]G\fR\fR
.ad
.RS 13n
Fetch command number \fIcount\fR. The default is the least recent history
command.
.RE

.sp
.ne 2
.na
\fB\fB/\fR\fIstring\fR\fR
.ad
.RS 13n
Search backward through history for a previous command containing \fIstring\fR.
\fIstring\fR is terminated by a RETURN or NEWLINE. If string is preceded by a
\fB^\fR, the matched line must begin with \fIstring\fR. If \fIstring\fR is
null, the previous string is used.
.RE

.sp
.ne 2
.na
\fB\fB?\fR\fIstring\fR\fR
.ad
.RS 13n
Search forward through history for a previous command containing \fIstring\fR.
```

```
4482 \fIstring\fR is terminated by a RETURN or NEWLINE. If string is preceded by a
4483 \fB^\fR, the matched line must begin with \fIstring\fR. If \fIstring\fR is
4484 null, the previous string is used.
4485 .sp
4486 Same as \fI/\fR except that search is in the forward direction.
4487 .RE

4489 .sp
4490 .ne 2
4491 .na
4492 \fB\fBn\fR\fR
4493 .ad
4494 .RS 13n
4495 Search in the backwards direction for the next match of the last pattern to
4496 \fI/\fR or \fI?\fR commands.
4497 .RE

4499 .sp
4500 .ne 2
4501 .na
4502 \fB\fBN\fR\fR
4503 .ad
4504 .RS 13n
4505 Search in the forward direction for next match of the last pattern to \fI/\fR
4506 or \fI?\fR.
4507 .RE

4509 .SS "Text Modification Edit Commands"
4510 .sp
4511 .LP
4512 The following commands modify the line:
4513 .sp
4514 .ne 2
4515 .na
4516 \fB\fBa\fR\fR
4517 .ad
4518 .RS 19n
4519 Enter input mode and enter text after the current character.
4520 .RE

4522 .sp
4523 .ne 2
4524 .na
4525 \fB\fBA\fR\fR
4526 .ad
4527 .RS 19n
4528 Append text to the end of the line. Equivalent to \fB$a\fR.
4529 .RE

4531 .sp
4532 .ne 2
4533 .na
4534 \fB\fB[\fR\fIcount\fR\fB]c\fR\fImotion\fR\fR
4535 .ad
4536 .br
4537 .na
4538 \fB\fBc[\fR\fIcount\fR\fB]\fR\fImotion\fR\fR
4539 .ad
4540 .RS 19n
4541 Delete current character through the character that \fImotion\fR would move the
4542 cursor to and enter input mode. If \fImotion\fR is \fBc\fR, the entire line is
4543 deleted and input mode entered.
4544 .RE

4546 .sp
4547 .ne 2
```



```
4548 .na
4549 \fB\fBC\fR\fR
4550 .ad
4551 .RS 19n
4552 Delete the current character through the end of line and enter input mode.
4553 Equivalent to \fBc$\fR.
4554 .RE

4556 .sp
4557 .ne 2
4558 .na
4559 \fB\fBS\fR\fR
4560 .ad
4561 .RS 19n
4562 Equivalent to \fBcc\fR.
4563 .RE

4565 .sp
4566 .ne 2
4567 .na
4568 \fB\fB[\fR\fIcount\fR\fB]s\fR\fR
4569 .ad
4570 .RS 19n
4571 Replace characters under the cursor in input mode.
4572 .RE

4574 .sp
4575 .ne 2
4576 .na
4577 \fB\fBD[\fR\fIcount\fR\fB]d\fR\fImotion\fR\fR
4578 .ad
4579 .RS 19n
4580 Delete the current character through the end of line. Equivalent to d$.
4581 .RE

4583 .sp
4584 .ne 2
4585 .na
4586 \fB\fBd[\fR\fIcount\fR\fB]\fR\fImotion\fR\fR
4587 .ad
4588 .RS 19n
4589 Delete current character through the character that \fImotion\fR would move to.
4590 If \fImotion\fR is d , the entire line is deleted.
4591 .RE

4593 .sp
4594 .ne 2
4595 .na
4596 \fB\fBi\fR\fR
4597 .ad
4598 .RS 19n
4599 Enter input mode and insert text before the current character.
4600 .RE

4602 .sp
4603 .ne 2
4604 .na
4605 \fB\fBI\fR\fR
4606 .ad
4607 .RS 19n
4608 Insert text before the beginning of the line. Equivalent to \fB0i\fR.
4609 .RE

4611 .sp
4612 .ne 2
4613 .na
```

```
4614 \fB\fB[\fR\fIcount\fR\fB]P\fR\fR
4615 .ad
4616 .RS 19n
4617 Place the previous text modification before the cursor.
4618 .RE

4620 .sp
4621 .ne 2
4622 .na
4623 \fB\fB[\fR\fIcount\fR\fB]p\fR\fR
4624 .ad
4625 .RS 19n
4626 Place the previous text modification after the cursor.
4627 .RE

4629 .sp
4630 .ne 2
4631 .na
4632 \fB\fBR\fR\fR
4633 .ad
4634 .RS 19n
4635 Enter input mode and replace characters on the screen with characters you type
4636 overlay fashion.
4637 .RE

4639 .sp
4640 .ne 2
4641 .na
4642 \fB\fB[\fR\fIcount\fR\fB]r\fR\fIc\fR\fR
4643 .ad
4644 .RS 19n
4645 Replace the \fIcount\fR characters starting at the current cursor position with
4646 \fIc\fR, and advance the cursor.
4647 .RE

4649 .sp
4650 .ne 2
4651 .na
4652 \fB\fB[\fR\fIcount\fR\fB]x\fR\fR
4653 .ad
4654 .RS 19n
4655 Delete current character.
4656 .RE

4658 .sp
4659 .ne 2
4660 .na
4661 \fB\fB[\fIcount\fR]X\fR\fR
4662 .ad
4663 .RS 19n
4664 Delete preceding character.
4665 .RE

4667 .sp
4668 .ne 2
4669 .na
4670 \fB\fB[\fIcount\fR].\fR\fR
4671 .ad
4672 .RS 19n
4673 Repeat the previous text modification command.
4674 .RE

4676 .sp
4677 .ne 2
4678 .na
4679 \fB\fB[\fIcount\fR]~\fR\fR
```



```
4680 .ad
4681 .RS 19n
4682 Invert the case of the \fIcount\fR characters starting at the current cursor
4683 position and advance the cursor.
4684 .RE

4686 .sp
4687 .ne 2
4688 .na
4689 \fB\fB[\fIcount\fR]_\fR\fR
4690 .ad
4691 .RS 19n
4692 Causes the \fIcount\fR word of the previous command to be appended and input
4693 mode entered. The last word is used if \fIcount\fR is omitted.
4694 .RE

4696 .sp
4697 .ne 2
4698 .na
4699 \fB\fB*\fR\fR
4700 .ad
4701 .RS 19n
4702 Causes an \fB*\fR to be appended to the current word and file name generation
4703 attempted. If no match is found, it rings the bell. Otherwise, the word is
4704 replaced by the matching pattern and input mode is entered.
4705 .RE

4707 .sp
4708 .ne 2
4709 .na
4710 \fB\fB\e\fR\fR
4711 .ad
4712 .RS 19n
4713 Command or file name completion as described in this manual page.
4714 .RE

4716 .SS "Other Edit Commands"
4717 .sp
4718 .LP
4719 The following miscellaneous edit commands are supported:
4720 .sp
4721 .ne 2
4722 .na
4723 \fB\fB[\fR\fIcount\fR\fB]y\fR\fImotion\fR\fR
4724 .ad
4725 .br
4726 .na
4727 \fB\fBy[\fR\fIcount\fR\fB]\fR\fImotion\fR\fR
4728 .ad
4729 .RS 18n
4730 Yank the current character through the character to which \fImotion\fR would
4731 move the cursor. Put the yanked characters in the delete buffer. The text and
4732 cursor position are unchanged.
4733 .RE

4735 .sp
4736 .ne 2
4737 .na
4738 \fB\fByy\fR\fR
4739 .ad
4740 .RS 18n
4741 Yank the current line.
4742 .RE

4744 .sp
4745 .ne 2
```

```
.na
\fB\fBY\fR\fR
.ad
.RS 18n
Yank the current line from the current cursor location to the end of the line.
Equivalent to \fBy$\fR.
.RE

.sp
.ne 2
.na
\fB\fBu\fR\fR
.ad
.RS 18n
Undo the last text modifying command.
.RE

.sp
.ne 2
.na
\fB\fBU\fR\fR
.ad
.RS 18n
Undo all the text modifying commands performed on current line.
.RE

.sp
.ne 2
.na
\fB\fB[\fR\fIcount\fR\fB]V\fR\fR
.ad
.RS 18n
Return the command :
.sp
.in +2
.nf
hist -e ${VISUAL:-${EDITOR:-vi}} \fIcount\fR
.fi
.in -2
.sp

in the input buffer. If \fIcount\fR is omitted, the current line is used.
.RE

.sp
.ne 2
.na
\fB\fB^L\fR\fR
.ad
.RS 18n
Line feed and print the current line. This command only works in control mode.
.RE

.sp
.ne 2
.na
\fB\fB^J\fR\fR
.ad
.RS 18n
New line. Execute the current line, regardless of mode.
.RE

.sp
.ne 2
.na
\fB\fB^M\fR\fR
```



```
.ad
.RS 18n
Return. Execute the current line, regardless of mode.
.RE

.sp
.ne 2
.na
\fB\fB#\fR\fR
.ad
.RS 18n
If the first character of the command is a \fB#\fR , delete this \fB#\fR and
each \fB#\fR that follows a NEWLINE.
.sp
Otherwise, send the line after inserting a \fB#\fR in front of each line in the
command.
.sp
This is command is useful for causing the current line to be inserted in the
history as a comment and un-commenting previously commented commands in the
history file.
.RE

.sp
.ne 2
.na
\fB\fB[\fR\fIcount\fR\fB]=\fR\fR
.ad
.RS 18n
If \fIcount\fR is not specified, generate the list of matching commands or file
names as described in this manual page.
.sp
Otherwise, replace the word at the current cursor location with the \fIcount\fR
item from the most recently generated command or file list. If the cursor is
not on a word, it is inserted after the current cursor location.
.RE

.sp
.ne 2
.na
\fB\fB@\fR\fIletter\fR\fR
.ad
.RS 18n
Search your alias list for an alias by the name \fIletter\fR. If an alias of
this name is defined, insert its value on the input queue for processing.
.RE

.sp
.ne 2
.na
\fB\fB^V\fR\fR
.ad
.RS 18n
Display version of the shell.
.RE

.SS "Built-in Commands"
.sp
.LP
The following simple-commands are executed in the shell process. Input and
output redirection is permitted. Unless otherwise indicated, the output is
written on file descriptor \fB1\fR and the exit status, when there is no syntax
error, is \fB0\fR. Except for \fB:\fR, \fBtrue\fR, \fBfalse\fR, \fBecho\fR,
\fBnewgrp\fR, and \fBlogin\fR, all built-in commands accept \fB--\fR to
indicate the end of options. They also interpret the option \fB--man\fR as a
request to display the manual page onto standard error and \fB-?\fR as a help
request which prints a usage message on standard error.
```

```
4878 .sp
4879 .LP
4880 Commands that are preceded by one or two \fB++\fR symbols are special built-in
4881 commands and are treated specially in the following ways:
4882 .RS +4
4883 .TP
4884 1.
4885 Variable assignment lists preceding the command remain in effect when the
4886 command completes.
4887 .RE
4888 .RS +4
4889 .TP
4890 2.
4891 I/O redirections are processed after variable assignments.
4892 .RE
4893 .RS +4
4894 .TP
4895 3.
4896 Errors cause a script that contains them to abort.
4897 .RE
4898 .RS +4
4899 .TP
4900 4.
4901 They are not valid function names.
4902 .RE
4903 .RS +4
4904 .TP
4905 5.
4906 Words following a command preceded by \fB++\fR that are in the format of a
4907 variable assignment are expanded with the same rules as a variable assignment.
4908 This means that tilde substitution is performed after the \fB=\fR sign and
4909 field splitting and file name generation are not performed.
4910 .RE
4911 .sp
4912 .ne 2
4913 .na
4914 \fB\fB+ : [\fR\fIarg ...\fR\fB]\fR\fR
4915 .ad
4916 .sp .6
4917 .RS 4n
4918 The command only expands parameters.
4919 .RE

4921 .sp
4922 .ne 2
4923 .na
4924 \fB\fB+ .\fR \fIname\fR \fB[\fR\fIarg ...\fR\fB]\fR\fR
4925 .ad
4926 .sp .6
4927 .RS 4n
4928 If \fIname\fR is a function defined with the \fBfunction\fR \fBname\fR reserved
4929 word syntax, the function is executed in the current environment (as if it had
4930 been defined with the \fIname()\fR syntax.) Otherwise if \fIname\fR refers to a
4931 file, the file is read in its entirety and the commands are executed in the
4932 current shell environment. The search path specified by PATH is used to find
4933 the directory containing the file. If any arguments \fIarg\fR are specified,
4934 they become the positional parameters while processing the . command and the
4935 original positional parameters are restored upon completion. Otherwise the
4936 positional parameters are unchanged. The exit status is the exit status of the
4937 last command executed.
4938 .RE

4940 .sp
4941 .ne 2
4942 .na
4943 \fB\fB++ alias [\fR\fB-ptx\fR\fB] [\fR\fIname\fR\fB[
```



```
4944 =\fR\fIvalue\fR\fB]] ...\fR\fR
4945 .ad
4946 .sp .6
4947 .RS 4n
4948 \fBalias\fR with no arguments prints the list of aliases in the form
4949 \fIname\fR\fB=\fR\fIvalue\fR on standard output. The \fB-p\fR option causes the
4950 word alias to be inserted before each one. When one or more arguments are
4951 specified, an \fIalias\fR is defined for each \fIname\fR whose \fIvalue\fR is
4952 specified. A trailing space in \fIvalue\fR causes the next word to be checked
4953 for alias substitution. The obsolete \fB-t\fR option is used to set and list
4954 tracked aliases. The value of a tracked alias is the full pathname
4955 corresponding to the specified \fIname\fR. The value becomes undefined when the
4956 value of \fBPATH\fR is reset but the alias remains tracked. Without the
4957 \fB-t\fR option, for each \fIname\fR in the argument list for which no
4958 \fIvalue\fR is specified, the name and value of the alias is printed. The
4959 obsolete -x option has no effect. The exit status is \fBnon-zero\fR if a
4960 \fIname\fR is specified, but no value, and no alias has been defined for the
4961 \fIname\fR.
4962 .RE

4964 .sp
4965 .ne 2
4966 .na
4967 \fB\fBbg [\fR \fIjob\fR\fB\&...]\fR\fR
4968 .ad
4969 .sp .6
4970 .RS 4n
4971 This command is only on systems that support job control. Puts each specified
4972 \fIjob\fR into the background. The current job is put in the background if
4973 \fIjob\fR is not specified. See the \fBJobs\fR section of this manual page for
4974 a description of the format of \fIjob\fR.
4975 .RE

4977 .sp
4978 .ne 2
4979 .na
4980 \fB\fB+ break [\fR\fIn\fR\fB]\fR\fR
4981 .ad
4982 .sp .6
4983 .RS 4n
4984 Exit from the enclosing \fBfor\fR, \fBwhile\fR, \fBuntil\fR, or \fBselect\fR
4985 loop, if any. If \fIn\fR is specified, then break \fIn\fR levels.
4986 .RE

4988 .sp
4989 .ne 2
4990 .na
4991 \fB\fBbuiltin [\fR\fB-ds\fR \fB] [\fR\fB-f\fR \fIfile\fR\fB]
4992 [\fR\fIname ...\fR\fB]\fR\fR
4993 .ad
4994 .sp .6
4995 .RS 4n
4996 If \fIname\fR is not specified, and no \fB-f\fR option is specified, the
4997 built-ins are printed on standard output. The \fB-s\fR option prints only the
4998 special built-ins. Otherwise, each \fIname\fR represents the pathname whose
4999 basename is the name of the built-in. The entry point function name is
5000 determined by prepending \fIb\fR to the built-in name. The ISO C/C++ prototype
5001 is \fBb\fR\fImycommand(int argc, char *argv[], void *context)\fR for the
5002 built-in command \fImycommand\fR where \fIargv\fR is an array of \fIargc\fR
5003 elements and \fIcontext\fR is an optional pointer to a \fBShell_t\fR structure
5004 as described in \fB<ast/shell.h>\fR Special built-ins cannot be bound to a
5005 pathname or deleted. The \fB-d\fR option deletes each of the specified
5006 built-ins. On systems that support dynamic loading, the \fB-f\fR option names a
5007 shared library containing the code for built-ins. The shared library prefix
5008 and/or suffix, which depend on the system, can be omitted. Once a library is
5009 loaded, its symbols become available for subsequent invocations of
```

\fBbuiltin\fR. Multiple libraries can be specified with separate invocations of the \fBbuiltin\fR command. Libraries are searched in the reverse order in which they are specified. When a library is loaded, it looks for a function in the library whose name is \fBlib_init()\fR and invokes this function with an argument of \fB0\fR.
.RE

.sp
.ne 2
.na
\fB\fBcd\fR \fB[\fR\fB-LP\fR\fB] [\fR\fIarg\fR\fB]\fR\fR
.ad
.br
.na
\fB\fBcd\fR \fB[\fR\fB-LP\fR\fB]\fR \fIold\fR \fInew\fR\fR
.ad
.sp .6
.RS 4n
This command has two forms.
.sp
In the first form it changes the current directory to \fIarg\fR. If \fIarg\fR is a \fB-\fR, the directory is changed to the previous directory. The shell variable \fBHOME\fR is the default \fIarg\fR. The variable \fBPWD\fR is set to the current directory. The shell variable \fBCDPATH\fR defines the search path for the directory containing \fIarg\fR. Alternative directory names are separated by a colon (\fB:\fR). The default path is \fBNULL\fR (specifying the current directory). The current directory is specified by a null path name, which can appear immediately after the equal sign or between the colon delimiters anywhere else in the path list. If \fIarg\fR begins with a \fB/\fR, the search path is not used. Otherwise, each directory in the path is searched for \fIarg\fR.
.sp
The second form of \fBcd\fR substitutes the string \fInew\fR for the string \fIold\fR in the current directory name, \fBPWD\fR, and tries to change to this new directory. By default, symbolic link names are treated literally when finding the directory name. This is equivalent to the \fB-L\fR option. The \fB-P\fR option causes symbolic links to be resolved when determining the directory. The last instance of \fB-L\fR or \fB-P\fR on the command line determines which method is used. The \fBcd\fR command cannot be executed by \fBrksh93\fR.
.RE

.sp
.ne 2
.na
\fB\fBcommand\fR \fB[\fR\fB-pvVx\fR\fB]\fR \fIname\fR
\fB[\fR\fIarg ...\fR\fB]\fR\fR
.ad
.sp .6
.RS 4n
Without the \fB-v\fR or \fB-V\fR options, executes \fIname\fR with the arguments specified by \fIarg\fR.
.sp
The \fB-p\fR option causes a default path to be searched rather than the one defined by the value of \fBPATH\fR. Functions are not searched when finding \fIname\fR. In addition, if \fIname\fR refers to a special built-in, none of the special properties associated with the leading daggers are honored. For example, the predefined alias \fBredirect='command exec'\fR prevents a script from terminating when an invalid redirection is specified.
.sp
With the \fB-x\fR option, if command execution would result in a failure because there are too many arguments, \fBerrno E2BIG\fR, the shell invokes command \fIname\fR multiple times with a subset of the arguments on each invocation. Arguments that occur prior to the first word that expands to multiple arguments and after the last word that expands to multiple arguments are passed on each invocation. The exit status is the maximum invocation exit

status.
.sp
With the \fB-v\fR option, \fBcommand\fR is equivalent to the built-in \fBwhence\fR command described in this section. The \fB-V\fR option causes \fBcommand\fR to act like \fBwhence -v\fR.
.RE

.sp
.ne 2
.na
\fB\fB+continue\fR \fB[\fR\fIn\fR\fB]\fR\fR
.ad
.sp .6
.RS 4n
Resumes the next iteration of the enclosing \fBfor\fR, \fBwhile\fR, \fBuntil\fR, or \fBselect\fR loop. If \fIn\fR is specified, then resume at the \fIn\fRth enclosing loop.
.RE

.sp
.ne 2
.na
\fB\fBdisown\fR \fB[\fR\fIjob...\fR\fB]\fR\fR
.ad
.sp .6
.RS 4n
Causes the shell not to send a \fBHUP\fR signal to each specified \fIjob\fR, or all active jobs if \fIjob\fR is omitted, when a login shell terminates.
.RE

.sp
.ne 2
.na
\fB\fBecho\fR \fB[\fR\fIarg ...\fR\fB]\fR\fR
.ad
.sp .6
.RS 4n
When the first \fIarg\fR does not begin with a \fB-\fR, and none of the arguments contain a backslash (\fB\e\fR), prints each of its arguments separated by a SPACE and terminated by a NEWLINE. Otherwise, the behavior of \fBecho\fR is system dependent and \fBprint\fR or \fBprintf\fR described in this section should be used. See \fBecho\fR(1) for usage and description.
.RE

.sp
.ne 2
.na
\fB\fB+eval\fR \fB[\fR\fIarg ...\fR\fB]\fR\fR
.ad
.sp .6
.RS 4n
The arguments are read as input to the shell and the resulting commands are executed.
.RE

.sp
.ne 2
.na
\fB\fB+exec\fR [\fB-c\fR] [\fB-a\fR \fIname ...\fR]
\fB[\fR\fIarg ...\fR\fB]\fR\fR
.ad
.sp .6
.RS 4n
If \fIarg\fR is specified, the command specified by the arguments is executed in place of this shell without creating a new process. The \fB-c\fR option causes the environment to be cleared before applying variable assignments

associated with the exec invocation. The \fB-a\fR option causes \fIname\fR rather than the first \fIarg\fR, to become \fBargv[0]\fR for the new process. Input and output arguments can appear and affect the current process. If \fIarg\fR is not specified, the effect of this command is to modify file descriptors as prescribed by the input/output redirection list. In this case, any file descriptor numbers greater than \fB2\fR that are opened with this mechanism are closed when invoking another program.
.RE

.sp
.ne 2
.na
\fB\fB+exit\fR \fB[\fR\fIn\fR\fB]\fR\fR
.ad
.sp .6
.RS 4n
Causes the shell to exit with the exit status specified by \fIn\fR. The value is the least significant 8 bits of the specified status. If \fIn\fR is omitted, then the exit status is that of the last command executed. An end-of-file also causes the shell to exit except for a shell which has the \fBignoreeof\fR option turned on. See \fBset\fR.
.RE

.sp
.ne 2
.na
\fB\fB++export\fR \fB[\fR\fB-p\fR\fB]\fR
\fB[\fR\fIname\fR\fB[=\fR\fIvalue\fR\fB]] ...\fR\fR
.ad
.sp .6
.RS 4n
If \fIname\fR is not specified, the names and values of each variable with the export attribute are printed with the values quoted in a manner that allows them to be re-entered. The \fB-p\fR option causes the word export to be inserted before each one. Otherwise, the specified \fIname\fRs are marked for automatic export to the environment of subsequently-executed commands.
.RE

.sp
.ne 2
.na
\fB\fBfalse\fR\fR
.ad
.sp .6
.RS 4n
Does nothing, and exits \fB1\fR. Used with \fBuntil\fR for infinite loops.
.RE

.sp
.ne 2
.na
\fB\fBfg\fR \fB[\fR\fIjob ...\fR\fB]\fR\fR
.ad
.sp .6
.RS 4n
This command is only on systems that support job control. Each \fIjob\fR specified is brought to the foreground and waited for in the specified order. Otherwise, the current job is brought into the foreground. See \fBJobs\fR for a description of the format of \fIjob\fR.
.RE

.sp
.ne 2
.na
\fB\fBgetconf\fR \fB[\fR\fIname\fR \fB[\fR\fIpathname\fR\fB]]\fR\fR
.ad

.sp .6
.RS 4n
Prints the current value of the configuration parameter specified by \fIname\fR. The configuration parameters are defined by the IEEE POSIX 1003.1 and IEEE POSIX 1003.2 standards. See \fBpathconf\fR(2) and \fBsysconf\fR(3C).
.sp
The \fIpathname\fR argument is required for parameters whose value depends on the location in the file system. If no arguments are specified, \fBgetconf\fR prints the names and values of the current configuration parameters. The pathname \fB/\fR is used for each of the parameters that requires \fIpathname\fR.
.RE

.sp
.ne 2
.na
\fB\fBgetopts\fR \fB[\fR \fB-a\fR \fIname\fR\fB]\fR \fIoptstring\fR \fIvname\fR \fB[\fR\fIarg ...\fR\fB]\fR\fR
.ad
.sp .6
.RS 4n
Checks \fIarg\fR for legal options. If \fIarg\fR is omitted, the positional parameters are used. An option argument begins with a \fB+\fR or a \fB-\fR. An option that does not begin with \fB+\fR or \fB-\fR or the argument \fB--\fR ends the options. Options beginning with \fB+\fR are only recognized when \fIoptstring\fR begins with a \fB+\fR. \fIoptstring\fR contains the letters that \fBgetopts\fR recognizes. If a letter is followed by a \fB:\fR, that option is expected to have an argument. The options can be separated from the argument by blanks. The option \fB-?\fRcauses \fBgetopts\fR to generate a usage message on standard error. The \fB-a\fR option can be used to specify the name to use for the usage message, which defaults to $0. \fBgetopts\fR places the next option letter it finds inside variable \fIvname\fR each time it is invoked. The option letter is prepended with a \fB+\fR when \fIarg\fR begins with a \fB+\fR. The index of the next \fIarg\fR is stored in \fBOPTIND\fR. The option argument, if any, gets stored in \fBOPTARG\fR. A leading : in \fIoptstring\fR causes \fBgetopts\fR to store the letter of an invalid option in \fBOPTARG\fR, and to set \fIvname\fR to \fB?\fR for an unknown option and to: when a required option argument is missing. Otherwise, \fBgetopts\fR prints an error message. The exit status is \fBnon-zero\fR when there are no more options. There is no way to specify any of the options \fB:\fR, \fB+\fR, \fB-\fR, \fB?\fR, \fB[\fR, and \fB]\fR. The option \fB#\fR can only be specified as the first option.
.RE

.sp
.ne 2
.na
\fB\fBhist\fR \fB[\fR \fB-e\fR \fIename\fR\fB]\fR\fB[\fR\fB-nlr\fR\fB]\fR
\fB[\fR \fIfirst\fR\fB[\fR\fIlast\fR \fB]]\fR\fR
.ad
.br
.na
\fB\fR
.ad
.br
.na
\fB\fBhist\fR \fB-s\fR \fB[\fR \fIold\fR\fB=\fR\fInew\fR \fB] [\fR
\fIcommand\fR\fB]\fR\fR
.ad
.sp .6
.RS 4n
In the first form, a range of commands from \fIfirst\fR to \fIlast\fR is selected from the last \fBHISTSIZE\fR commands that were typed at the terminal. The arguments \fIfirst\fR and \fIlast\fR can be specified as a number or as a string. A string is used to locate the most recent command starting with the specified string. A negative number is used as an offset to the current command

number. If the -l option is selected, the commands are listed on standard output. Otherwise, the editor program \fIename\fR is invoked on a file containing these keyboard commands. If \fIename\fR is not supplied, then the value of the variable \fBHISTEDIT\fR is used. If \fBHISTEDIT\fR is not set, then \fBFCEDIT\fR (default \fB/bin/ed\fR) is used as the editor. When editing is complete, the edited command(s) is executed if the changes have been saved. If \fIlast\fR is not specified, then it is set to \fIfirst\fR. If \fIfirst\fR is not specified, the default is the previous command for editing and \fB-16\fR for listing. The option \fB-r\fR reverses the order of the commands and the option \fB-n\fR suppresses command numbers when listing. In the second form, \fIcommand\fR is interpreted as \fIfirst\fR described in this section and defaults to the last command executed. The resulting command is executed after the optional substitution \fIold\fR\fB=\fR\fInew\fR is performed.
.RE

.sp
.ne 2
.na
\fB\fBjobs\fR \fB-lnp\fR \fB[\fR\fIjob ...\fR\fB]\fR\fR
.ad
.sp .6
.RS 4n
Lists information about each specified job, or all active jobs if \fIjob\fR is omitted. The \fB-l\fR option lists process ids in addition to the normal information. The \fB-n\fR option only displays jobs that have stopped or exited since last notified. The \fB-p\fR option causes only the process group to be listed. See \fBJobs\fR for a description of the format of \fIjob\fR.
.RE

.sp
.ne 2
.na
\fB\fBkill\fR \fB[\fR\fB-s\fR \fIsigname\fR\fB]\fR \fIjob ...\fR\fR
.ad
.br
.na
\fB\fBkill\fR \fB[\fR\fB-n\fR \fIsignum\fR\fB]\fR \fIjob ...\fR\fR
.ad
.br
.na
\fB\fBkill\fR \fB-l\fR \fB[\fR\fIsig ...\fR\fB]\fR\fR
.ad
.sp .6
.RS 4n
Sends either the \fBTERM\fR (terminate) signal or the specified signal to the specified jobs or processes. Signals are either specified by number with the \fB-n\fR option or by name with the \fB-s\fR option (as specified in \fB<signal.h\fR>, stripped of the prefix ‘\fBSIG\fR with the exception that \fBSIGCLD\fR is named \fBCHLD\fR). For backward compatibility, the \fBn\fR and \fBs\fR can be omitted and the number or name placed immediately after the \fB-\fR. If the signal being sent is \fBTERM\fR (terminate) or \fBHUP\fR (hang up), then the job or process is sent a \fBCONT\fR (continue) signal if it is stopped. The argument \fIjob\fR can be the process id of a process that is not a member of one of the active jobs. See \fBJobs\fR for a description of the format of \fIjob\fR. In the third form, \fBkill -l\fR, if \fIsig\fR is not specified, the signal names are listed. Otherwise, for each \fIsig\fR that is a name, the corresponding signal number is listed. For each \fIsig\fR that is a number, the signal name corresponding to the least significant 8 bits of \fIsig\fR is listed.
.RE

.sp
.ne 2
.na
\fB\fBlet\fR \fB[\fR\fIarg ...\fR\fB]\fR\fR
.ad

.sp .6
.RS 4n
Each \fIarg\fR is a separate arithmetic expression to be evaluated. See the \fBArithmetic Evaluation\fR section of this manual page for a description of arithmetic expression evaluation. The exit status is \fB0\fR if the value of the last expression is \fBnon-zero\fR, and \fB1\fR otherwise.
.RE

.sp
.ne 2
.na
\fB\fB+newgrp\fR \fB[\fR\fIarg ...\fR\fB]\fR\fR
.ad
.sp .6
.RS 4n
Equivalent to \fBexec\fR \fB/bin/newgrp\fR \fIarg ...\fR
.RE

.sp
.ne 2
.na
\fB\fBprint\fR [\fB-Renprs\fR] \fB[\fR \fB-u\fR \fIunit\fR\fB] [\fR \fB-f\fR \fIformat\fR \fB] [\fR \fIarg ...\fR\fB]\fR\fR
.ad
.sp .6
.RS 4n
With no options or with option \fB-\fR or \fB--\fR, each \fIarg\fR is printed on standard output. The \fB-f\fR option causes the arguments to be printed as described by \fBprintf\fR. In this case, any \fBe\fR, \fBn\fR, \fBr\fR, or \fBR\fR options are ignored. Otherwise, unless the \fB-R\fR or \fB-r,\fR are specified, the following escape conventions are applied:
.sp
.ne 2
.na
\fB\fBea\fR\fR
.ad
.RS 8n
Alert character (\fBASCII\fR 07)
.RE

.sp
.ne 2
.na
\fB\fBeb\fR\fR
.ad
.RS 8n
Backspace character (\fBASCII\fR 010)
.RE

.sp
.ne 2
.na
\fB\fBec\fR\fR
.ad
.RS 8n
Causes print to end without processing more arguments and not adding a NEWLINE
.RE

.sp
.ne 2
.na
\fB\fBef\fR\fR
.ad
.RS 8n
Form-feed character (\fBASCII\fR 014)
.RE

```
.sp
.ne 2
.na
\fB\fB\en\fR\fR
.ad
.RS 8n
NEWLINE character (\fBASCII\fR 012)
.RE

.sp
.ne 2
.na
\fB\fB\er\fR\fR
.ad
.RS 8n
RETURN character (\fBASCII\fR 015)
.RE

.sp
.ne 2
.na
\fB\fB\et\fR\fR
.ad
.RS 8n
TAB character (\fBASCII\fR 011)
.RE

.sp
.ne 2
.na
\fB\fB\ev\fR\fR
.ad
.RS 8n
Vertical TAB character (\fBASCII\fR 013)
.RE

.sp
.ne 2
.na
\fB\fB\eE\fR\fR
.ad
.RS 8n
Escape character (\fBASCII\fR 033)
.RE

.sp
.ne 2
.na
\fB\fB\e\e\fR\fR
.ad
.RS 8n
Backslash character \fB\e\fR
.RE

.sp
.ne 2
.na
\fB\fB\e0\fR\fIx\fR\fR
.ad
.RS 8n
Character defined by the 1, 2, or 3-digit octal string specified by \fIx\fR
.RE

The \fB-R\fR option prints all subsequent arguments and options other than
\fB-n\fR. The \fB-e\fR causes the escape conventions to be applied This is the
```



```
default behavior. It reverses the effect of an earlier \fB-r\fR. The \fB-p\fR
option causes the arguments to be written onto the pipe of the process spawned
with \fB|&\fR instead of standard output. The \fB-s\fR option causes the
arguments to be written onto the history file instead of standard output. The
\fB-u\fR option can be used to specify a one digit file descriptor unit number
\fIunit\fR on which the output is placed. The default is \fB1\fR. If the option
\fB-n\fR is used, no NEWLINE is added to the output.
.RE

.sp
.ne 2
.na
\fB\fBprintf\fR \fIformat\fR\fB[\fR\fIarg ...\fR\fB]\fR\fR
.ad
.sp .6
.RS 4n
The arguments \fIarg\fR are printed on standard output in accordance with the
\fBANSI-C\fR formatting rules associated with the format string \fIformat\fR.
If the number of arguments exceeds the number of format specifications, the
format string is reused to format remaining arguments. The following extensions
can also be used: A \fB%b\fR format can be used instead of \fB%s\fR to cause
escape sequences in the corresponding \fIarg\fR to be expanded as described in
\fBprint\fR. A \fB%B\fR option causes each of the arguments to be treated as
variable names and the binary value of the variables is printed. This is most
useful for variables with an attribute of b. A \fB%H\fR format can be used
instead of \fB%s\fR to cause characters in \fIarg\fR that are special in
\fBHTML\fR and \fBXML\fR to be output as their entity name. A \fB%P\fR format
can be used instead of \fB%s\fR to cause \fIarg\fR to be interpreted as an
extended regular expression and be printed as a shell pattern. A \fB%R\fR
format can be used instead of \fB%s\fR to cause \fIarg\fR to be interpreted as
a shell pattern and to be printed as an extended regular expression. A \fB%q\fR
format can be used instead of \fB%\fRs to cause the resulting string to be
quoted in a manner than can be input again to the shell. A
\fB%(\fR\fIdate-format\fR\fB)T\fR format can be use to treat an argument as a
date/time string and to format the date/time according to the \fIdate-format\fR
as defined for the \fBdate\fR(1) command. A \fB%Z\fR format outputs a byte
whose value is 0. The precision field of the %d format can be followed by a .
and the output base. In this case, the \fB#\fR flag character causes
\fBbase\fR\fI#\fR to be prepended. The \fB#\fR flag when used with the \fBd\fR
specifier without an output base, causes the output to be displayed in
thousands units with one of the suffixes \fBk\fR \fBM\fR \fBG\fR \fBT\fR
\fBP\fR \fBE\fR to indicate the unit. The \fB#\fR flag when used with the i
specifier causes the output to be displayed in \fB1024\fR with one of the
suffixes \fBKi\fR \fBMi\fR \fBGi\fR \fBTi\fR \fBPi\fR \fBEi\fR to indicate the
unit. The \fB=\fR flag has been added to center the output within the specified
field width.
.RE

.sp
.ne 2
.na
\fB\fBpwd\fR [\fB-LP\fR]\fR
.ad
.sp .6
.RS 4n
Outputs the value of the current working directory. The \fB-L\fR option is the
default. It prints the logical name of the current directory. If the \fB-P\fR
option is specified, all symbolic links are resolved from the name. The last
instance of \fB-L\fR or \fB-P\fR on the command line determines which method is
used.
.RE

.sp
.ne 2
.na
\fB\fBread\fR \fB[\fR\fB-Aprs\fR\fB] [\fR\fB-d\fR \fIdelim\fR\fB] [\fR \fB-n\fR
```

```
\fIn\fR\fB] [[\fR \fB-N\fR \fIn\fR\fB] [[\fR\fB-t\fR \fItimeout\fR\fB]
[\fR\fB-u\fR \fIunit\fR\fB] [\fR\fIvname\fR\fB?\fR\fIprompt\fR\fB] [\fR
\fIvname ...\fR \fB]\fR\fR
.ad
.sp .6
.RS 4n
The shell input mechanism. One line is read and is broken up into fields using
the characters in IFS as separators. The escape character, \fB\e\fR, is used to
remove any special meaning for the next character and for line continuation.
The \fB-d\fR option causes the read to continue to the first character of
\fIdelim\fR rather than \fBNEWLINE\fR. The \fB-n\fR option causes at most
\fIn\fR bytes to read rather a full line but returns when reading from a slow
device as soon as any characters have been read. The \fB-N\fR option causes
exactly \fIn\fR to be read unless an end-of-file has been encountered or the
read times out because of the \fB-t\fR option. In raw mode, \fB-r\fR, the
\fB\e\fR character is not treated specially. The first field is assigned to the
first \fIvname\fR, the second field to the second \fIvname\fR, etc., with
leftover fields assigned to the last \fIvname\fR. When \fIvname\fR has the
binary attribute and \fB-n\fR or \fB-N\fR is specified, the bytes that are read
are stored directly into the variable. If the -v is specified, then the value
of the first \fIvname\fR is used as a default value when reading from a
terminal device. The \fB-A\fR option causes the variable \fIvname\fR to be
unset and each field that is read to be stored in successive elements of the
indexed array \fIvname\fR. The \fB-p\fR option causes the input line to be
taken from the input pipe of a process spawned by the shell using \fB|&\fR. If
the \fB-s\fR option is present, the input is saved as a command in the history
file. The option \fB-u\fR can be used to specify a one digit file descriptor
unit \fIunit\fR to read from. The file descriptor can be opened with the
\fBexec\fR special built-in command. The default value of unit \fIn\fR is
\fB0\fR. The option \fB-t\fR is used to specify a time out in seconds when
reading from a terminal or pipe. If \fIvname\fR is omitted, then REPLY is used
as the default \fIvname\fR. An end-of-file with the \fB-p\fR option causes
cleanup for this process so that another can be spawned. If the first argument
contains a \fB?\fR, the remainder of this word is used as a prompt on standard
error when the shell is interactive. The exit status is \fB0\fR unless an
end-of-file is encountered or read has timed out.
.RE

.sp
.ne 2
.na
\fB\fB++readonly\fR \fB[\fR\fB-p\fR\fB] [\fR
\fIvname\fR\fB[=\fR\fIvalue\fR\fB]] ...\fR\fR
.ad
.sp .6
.RS 4n
If \fIvname\fR is not specified, the names and values of each variable with the
read-only attribute is printed with the values quoted in a manner that allows
them to be input again. The \fB-p\fR option causes the word \fBreadonly\fR to
be inserted before each one. Otherwise, the specified \fIvname\fRs are marked
\fBreadonly\fR and these names cannot be changed by subsequent assignment.
.RE

.sp
.ne 2
.na
\fB\fB+return\fR \fB[\fR\fIn\fR\fB]\fR\fR
.ad
.sp .6
.RS 4n
Causes a shell function or script to return to the invoking script with the
exit status specified by \fIn\fR. The value is the least significant 8 bits of
the specified status. If \fBn\fR is omitted, then the return status is that of
the last command executed. If return is invoked while not in a function or a
script, then it behaves the same as exit.
.RE
```



```
.sp
.ne 2
.na
\fB\fB+set [ \(+-BCGabefhkmnoprstuvx] [\(+-o [\fR \fIoption\fR \fB] ] ... [
\(+-A\fR \fIvname\fR\fB]\fR \fB[\fR\fIarg...\fR\fB]\fR\fR
.ad
.sp .6
.RS 4n
The \fBset\fR command supports the following options:
.sp
.ne 2
.na
\fB\fB-a\fR\fR
.ad
.sp .6
.RS 4n
All subsequent variables that are defined are automatically exported.
.RE

.sp
.ne 2
.na
\fB\fB-A\fR\fR
.ad
.sp .6
.RS 4n
Array assignment. Unset the variable \fIvname\fR and assign values sequentially
from the \fIarg\fR list. If \fB+A\fR is used, the variable \fIvname\fR is not
unset first.
.RE

.sp
.ne 2
.na
\fB\fB-b\fR\fR
.ad
.sp .6
.RS 4n
Prints job completion messages as soon as a background job changes state rather
than waiting for the next prompt.
.RE

.sp
.ne 2
.na
\fB\fB-B\fR\fR
.ad
.sp .6
.RS 4n
Enable brace pattern field generation. This is the default behavior.
.RE

.sp
.ne 2
.na
\fB\fB-C\fR\fR
.ad
.sp .6
.RS 4n
Prevents redirection (\fB>\fR) from truncating existing files. Files that are
created are opened with the \fBO_EXCL\fR mode. Requires \fB>|\fR to truncate a
file when turned on.
.RE

.sp
```

```
.ne 2
.na
\fB\fB-e\fR\fR
.ad
.sp .6
.RS 4n
If a command has a \fBnon-zero\fR exit status, execute the \fBERR\fR trap, if
set, and exit. This mode is disabled while reading profiles.
.RE

.sp
.ne 2
.na
\fB\fB-f\fR\fR
.ad
.sp .6
.RS 4n
Disables file name generation.
.RE

.sp
.ne 2
.na
\fB\fB-G\fR\fR
.ad
.sp .6
.RS 4n
Causes the pattern \fB**\fR by itself to match files and zero or more
directories and subdirectories when used for file name generation. If followed
by a \fB/\fR only directories and subdirectories are matched.
.RE

.sp
.ne 2
.na
\fB\fB-h\fR\fR
.ad
.sp .6
.RS 4n
Each command becomes a tracked alias when first encountered.
.RE

.sp
.ne 2
.na
\fB\fB-k\fR\fR
.ad
.sp .6
.RS 4n
Obsolete. All variable assignment arguments are placed in the environment for a
command, not just those that precede the command name.
.RE

.sp
.ne 2
.na
\fB\fB-m\fR\fR
.ad
.sp .6
.RS 4n
Background jobs run in a separate process group and a line prints upon
completion. The exit status of background jobs is reported in a completion
message. On systems with job control, this option is turned on automatically
for interactive shells.
.RE
```



```
.sp
.ne 2
.na
\fB\fB-n\fR\fR
.ad
.sp .6
.RS 4n
Read commands and check them for syntax errors, but do not execute them.
Ignored for interactive shells.
.RE

.sp
.ne 2
.na
\fB\fB-o\fR\fR
.ad
.sp .6
.RS 4n
If no option name is supplied, the list of options and their current settings
are written to standard output. When invoked with a \fB+\fR, the options are
written in a format that can be input again to the shell to restore the
settings. This option can be repeated to enable or disable multiple options.
.sp
The following argument can be one of the following option names:
.sp
.ne 2
.na
\fB\fBallexport\fR\fR
.ad
.sp .6
.RS 4n
Same as \fB-a\fR.
.RE

.sp
.ne 2
.na
\fB\fBbgnice\fR\fR
.ad
.sp .6
.RS 4n
All background jobs are run at a lower priority. This is the default mode.
.RE

.sp
.ne 2
.na
\fB\fBbraceexpand\fR\fR
.ad
.sp .6
.RS 4n
Same as \fB-\fRB.
.RE

.sp
.ne 2
.na
\fB\fBemacs\fR\fR
.ad
.sp .6
.RS 4n
Puts you in an \fBemacs\fR style inline editor for command entry.
.RE

.sp
.ne 2
```

```
.na
\fB\fBerrexit\fR\fR
.ad
.sp .6
.RS 4n
Same as \fB-e\fR.
.RE

.sp
.ne 2
.na
\fB\fBglobstar\fR\fR
.ad
.sp .6
.RS 4n
Same as \fB-G\fR.
.RE

.sp
.ne 2
.na
\fB\fBgmacs\fR\fR
.ad
.sp .6
.RS 4n
Puts you in a \fBgmacs\fR style inline editor for command entry.
.RE

.sp
.ne 2
.na
\fB\fBignoreeof\fR\fR
.ad
.sp .6
.RS 4n
The shell does not exit on end-of-file. The command \fBexit\fR must be used.
.RE

.sp
.ne 2
.na
\fB\fBkeyword\fR\fR
.ad
.sp .6
.RS 4n
Same as \fB-k\fR.
.RE

.sp
.ne 2
.na
\fB\fBmarkdirs\fR\fR
.ad
.sp .6
.RS 4n
All directory names resulting from file name generation have a trailing /
appended.
.RE

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



```
Same as \fB-m\fR.
.RE

.sp
.ne 2
.na
\fB\fBmultiline\fR\fR
.ad
.sp .6
.RS 4n
The built-in editors use multiple lines on the screen for lines that are longer
than the width of the screen. This might not work for all terminals.
.RE

.sp
.ne 2
.na
\fB\fBnoclobber\fR\fR
.ad
.sp .6
.RS 4n
Same as \fB-C\fR.
.RE

.sp
.ne 2
.na
\fB\fBnoexec\fR\fR
.ad
.sp .6
.RS 4n
Same as \fB-n\fR.
.RE

.sp
.ne 2
.na
\fB\fBnoglob\fR\fR
.ad
.sp .6
.RS 4n
Same as \fB-f\fR.
.RE

.sp
.ne 2
.na
\fB\fBnolog\fR\fR
.ad
.sp .6
.RS 4n
Do not save function definitions in the history file.
.RE

.sp
.ne 2
.na
\fB\fBnotify\fR\fR
.ad
.sp .6
.RS 4n
Same as \fB-b\fR.
.RE

.sp
.ne 2
```

```
.na
\fB\fBnounset\fR\fR
.ad
.sp .6
.RS 4n
Same as \fB-u\fR.
.RE

.sp
.ne 2
.na
\fB\fBpipefail\fR\fR
.ad
.sp .6
.RS 4n
A pipeline does not complete until all components of the pipeline have
completed, and the return value is the value of the last \fBnon-zero\fR command
to fail or zero if no command has failed.
.RE

.sp
.ne 2
.na
\fB\fBprivileged\fR\fR
.ad
.sp .6
.RS 4n
Same as \fB-p\fR.
.RE

.sp
.ne 2
.na
\fB\fBshowme\fR\fR
.ad
.sp .6
.RS 4n
When enabled, simple commands or pipelines preceded by a a semicolon (\fB;\fR)
is displayed as if the \fBxtrace\fR option were enabled but is not executed.
Otherwise, the leading \fB;\fR is ignored.
.RE

.sp
.ne 2
.na
\fB\fBtrackall\fR\fR
.ad
.sp .6
.RS 4n
Same as \fB-h\fR.
.RE

.sp
.ne 2
.na
\fB\fBverbose\fR\fR
.ad
.sp .6
.RS 4n
Same as \fB-v\fR.
.RE

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



```
.ad
.sp .6
.RS 4n
Puts you in insert mode of a \fBvi\fR style inline editor until you hit the
escape character 033. This puts you in control mode. A return sends the line.
.RE

.sp
.ne 2
.na
\fB\fBviraw\fR\fR
.ad
.sp .6
.RS 4n
Each character is processed as it is typed in \fBvi\fR mode.
.RE

.sp
.ne 2
.na
\fB\fBxtrace\fR\fR
.ad
.sp .6
.RS 4n
Same as \fB-x\fR.
.sp
If no option name is supplied, the current options settings are printed.
.RE

.RE

.sp
.ne 2
.na
\fB\fB-p\fR\fR
.ad
.sp .6
.RS 4n
Disables processing of the \fB$HOME/.profile\fR file and uses the file
\fB/etc/suid_profile\fR instead of the \fBENV\fR file. This mode is on whenever
the effective \fBuid\fR (\fBgid\fR) is not equal to the real \fBuid\fR
(\fBgid\fR). Turning this off causes the effective \fBuid\fR and \fBgid\fR to
be set to the real \fBuid\fR and \fBgid\fR.
.RE

.sp
.ne 2
.na
\fB\fB-r\fR\fR
.ad
.sp .6
.RS 4n
Enables the restricted shell. This option cannot be unset once set.
.RE

.sp
.ne 2
.na
\fB\fB-s\fR\fR
.ad
.sp .6
.RS 4n
Sort the positional parameters lexicographically.
.RE

.sp
```

```
.ne 2
.na
\fB\fB-t\fR\fR
.ad
.sp .6
.RS 4n
Obsolete. Exit after reading and executing one command.
.RE

.sp
.ne 2
.na
\fB\fB-u\fR\fR
.ad
.sp .6
.RS 4n
Treat \fBunset\fR parameters as an error when substituting.
.RE

.sp
.ne 2
.na
\fB\fB-v\fR\fR
.ad
.sp .6
.RS 4n
Print shell input lines as they are read.
.RE

.sp
.ne 2
.na
\fB\fB-x\fR\fR
.ad
.sp .6
.RS 4n
Print commands and their arguments as they are executed.
.RE

.sp
.ne 2
.na
\fB\fB--\fR\fR
.ad
.sp .6
.RS 4n
Do not change any of the options. This is useful in setting \fB$1\fR to a value
beginning with \fB-\fR. If no arguments follow this option then the positional
parameters are unset.
.RE

As an obsolete feature, if the first \fIarg\fR is - then the \fB-x\fR and
\fB-v\fR options are turned off and the next \fIarg\fR is treated as the first
argument. Using \fB+\fR rather than \fB-\fR causes these options to be turned
off. These options can also be used upon invocation of the shell. The current
set of options can be found in \fB$-\fR. Unless \fB-A\fR is specified, the
remaining arguments are positional parameters and are assigned, in order, to
\fB$1 $2 \&....\fR If no arguments are specified, then the names and values of
all variables are printed on the standard output.
.RE

.sp
.ne 2
.na
\fB\fB+shift\fR \fB[\fR\fIn\fR\fB]\fR\fR
.ad
```



```
.sp .6
.RS 4n
The positional parameters from \fB$\fR\fIn\fR\fB+1 ...\fR are renamed
\fB$1 ...\fR, the default \fIn\fR is \fB1\fR. The parameter \fIn\fR can be any
arithmetic expression that evaluates to a non-negative number less than or
equal to \fB$#\fR.
.RE

.sp
.ne 2
.na
\fB\fBsleep\fR \fIseconds\fR\fR
.ad
.sp .6
.RS 4n
Suspends execution for the number of decimal seconds or fractions of a second
specified by \fIseconds\fR.
.RE

.sp
.ne 2
.na
\fB\fB+trap\fR \fB-p\fR \fB[\fR\fIaction\fR\fB]\fR
\fB[\fR\fIsig\fR\fB] ...\fR\fR
.ad
.sp .6
.RS 4n
The \fB-p\fR option causes the trap action associated with each trap as
specified by the arguments to be printed with appropriate quoting. Otherwise,
\fIaction\fR is processed as if it were an argument to \fBeval\fR when the
shell receives signal(s) \fIsig\fR. Each \fIsig\fR can be specified as a number
or as the name of the signal. Trap commands are executed in order of signal
number. Any attempt to set a trap on a signal that was ignored on entry to the
current shell is ineffective. If \fIaction\fR is omitted and the first
\fIsig\fR is a number, or if \fIaction\fR is \fB-\fR, then the trap(s) for each
\fIsig\fR are reset to their original values. If \fIaction\fR is the null
string then this signal is ignored by the shell and by the commands it invokes.
If \fIsig\fR is \fBERR\fR then \fIaction\fR is executed whenever a command has
a \fBnon-zero\fR exit status. If \fIsig\fR is \fBDEBUG\fR then \fIaction\fR is
executed before each command. The variable \fB\&.sh.command\fR contains the
contents of the current command line when \fIaction\fR is running. If \fIsig\fR
is \fB0\fR or \fBEXIT\fR and the trap statement is executed inside the body of
a function defined with the \fBfunction\fR \fIname\fR syntax, then the command
\fIaction\fR is executed after the function completes. If \fIsig\fR is \fB0\fR
or \fBEXIT\fR for a trap set outside any function then the command \fIaction\fR
is executed on exit from the shell. If \fIsig\fR is \fBKEYBD\fR, then
\fIaction\fR is executed whenever a key is read while in \fBemacs\fR,
\fBgmacs\fR, or \fBvi\fR mode. The \fBtrap\fR command with no arguments prints
a list of commands associated with each signal number.
.RE

.sp
.ne 2
.na
\fB\fBtrue\fR\fR
.ad
.sp .6
.RS 4n
Does nothing, and exits \fB0\fR. Used with while for infinite loops.
.RE

.sp
.ne 2
.na
\fB\fB++typeset [\(+-AHflabnprtux ] [ \(+-EFLRZi[\fR\fIn\fR\fB] ] [\fR
\fIvname\fR\fB[=\fR\fIvalue\fR \fB] ]\fR\fR
```

```
.ad
.sp .6
.RS 4n
Sets attributes and values for shell variables and functions. When invoked
inside a function defined with the \fBfunction\fR \fIname\fR syntax, a new
instance of the variable \fIvname\fR is created, and the variable's value and
type are restored when the function completes.
.sp
Using \fB+\fR rather than \fB-\fR causes these options to be turned off. If no
\fIvname\fR arguments are specified, a list of \fIvname\fRs (and optionally the
\fIvalue\fRs) of the variables is printed. Using \fB+\fR rather than \fB-\fR
keeps the values from being printed.) The \fB-p\fR option causes \fBtypeset\fR
followed by the option letters to be printed before each name rather than the
names of the options. If any option other than \fB-p\fR is specified, only
those variables which have all of the specified options are printed. Otherwise,
the \fIvname\fRs and \fIattributes\fR of all variables that have attributes are
printed.
.sp
The following list of attributes can be specified:
.sp
.ne 2
.na
\fB\fB-a\fR\fR
.ad
.RS 6n
Declares \fIvname\fR to be an indexed array. This is optional unless except for
compound variable assignments.
.RE

.sp
.ne 2
.na
\fB\fB-A\fR\fR
.ad
.RS 6n
Declares \fIvname\fR to be an associative array. Sub-scripts are strings rather
than arithmetic expressions.
.RE

.sp
.ne 2
.na
\fB\fB-b\fR\fR
.ad
.RS 6n
The variable can hold any number of bytes of data. The data can be text or
binary. The value is represented by the \fBbase64\fR encoding of the data. If
\fB-Z\fR is also specified, the size in bytes of the data in the buffer is
determined by the size associated with the \fB-Z\fR. If the \fBbase64\fR string
assigned results in more data, it is truncated. Otherwise, it is filled with
bytes whose value is zero. The \fBprintf\fR format \fB%B\fR can be used to
output the actual data in this buffer instead of the \fBbase64\fR encoding of
the data.
.RE

.sp
.ne 2
.na
\fB\fB-E\fR\fR
.ad
.RS 6n
Declares \fIvname\fR to be a double precision floating point number. If \fIn\fR
is \fBnon-zero\fR, it defines the number of significant figures that are used
when expanding \fIvname\fR. Otherwise, ten significant figures is used.
.RE
```



```
.sp
.ne 2
.na
\fB\fB-f\fR\fR
.ad
.RS 6n
The names refer to function names rather than variable names. No assignments
can be made and the only other valid options are \fB-t\fR, \fB-u\fR, and
\fB-x.\fR The \fB-t\fR option turns on execution tracing for this function. The
\fB-u\fR option causes this function to be marked undefined. The \fBFPATH\fR
variable is searched to find the function definition when the function is
referenced. If no options other than \fB-f\fR is specified, then the function
definition is displayed on standard output. If \fB+f\fR is specified, then a
line containing the function name followed by a shell comment containing the
line number and path name of the file where this function was defined, if any,
is displayed.
.sp
The \fB-i\fR attribute cannot be specified with \fB-f\fR.
.RE

.sp
.ne 2
.na
\fB\fB-F\fR\fR
.ad
.RS 6n
Declares \fIvname\fR to be a double precision floating point number. If \fIn\fR
is \fBnon-zero\fR, it defines the number of places after the decimal point that
are used when expanding \fIvname\fR. Otherwise ten places after the decimal
point is used.
.RE

.sp
.ne 2
.na
\fB\fB-H\fR\fR
.ad
.RS 6n
This option provides UNIX to hostname file mapping on non-UNIX machines.
.RE

.sp
.ne 2
.na
\fB\fB-i\fR\fR
.ad
.RS 6n
Declares \fIvname\fR to be represented internally as integer. The right hand
side of an assignment is evaluated as an arithmetic expression when assigning
to an integer. If \fIn\fR is \fBnon-zero\fR, it defines the output arithmetic
base, otherwise the output base is ten.
.sp
The \fB-i\fR attribute cannot be specified along with \fB-R\fR, \fB-L\fR,
\fB-Z\fR, or \fB-f\fR.
.RE

.sp
.ne 2
.na
\fB\fB-l\fR\fR
.ad
.RS 6n
All uppercase characters are converted to lowercase. The uppercase option,
\fB-u\fR, is turned off.
.RE
```

```
.sp
.ne 2
.na
\fB\fB-L\fR\fR
.ad
.RS 6n
Left justify and remove leading blanks from \fIvalue\fR. If \fIn\fR is
\fBnon-zero\fR, it defines the width of the field, otherwise it is determined
by the width of the value of first assignment. When the variable is assigned
to, it is filled on the right with blanks or truncated, if necessary, to fit
into the field. The \fB-R\fR option is turned off.
.sp
The \fB-i\fR attribute cannot be specified with \fB-L\fR.
.RE

.sp
.ne 2
.na
\fB\fB-n\fR\fR
.ad
.RS 6n
Declares \fIvname\fR to be a reference to the variable whose name is defined by
the value of variable \fIvname\fR. This is usually used to reference a variable
inside a function whose name has been passed as an argument.
.RE

.sp
.ne 2
.na
\fB\fB-R\fR\fR
.ad
.RS 6n
Right justify and fill with leading blanks. If \fIn\fR is \fBnon-zero\fR, it
defines the width of the field, otherwise it is determined by the width of the
value of first assignment. The field is left filled with blanks or truncated
from the end if the variable is reassigned. The \fB-L\fR option is turned off.
.sp
The \fB-i\fR attribute cannot be specified with \fB-R\fR.
.RE

.sp
.ne 2
.na
\fB\fB-r\fR\fR
.ad
.RS 6n
The specified \fIvname\fRs are marked read-only and these names cannot be
changed by subsequent assignment.
.RE

.sp
.ne 2
.na
\fB\fB-t\fR\fR
.ad
.RS 6n
Tags the variables. Tags are user definable and have no special meaning to the
shell.
.RE

.sp
.ne 2
.na
\fB\fB-u\fR\fR
.ad
.RS 6n
```



```
All lowercase characters are converted to uppercase. The lowercase option,
\fB-l\fR, is turned off.
.RE

.sp
.ne 2
.na
\fB\fB-x\fR\fR
.ad
.RS 6n
The specified \fIvname\fRs are marked for automatic export to the environment
of subsequently-executed commands. Variables whose names contain a . cannot be
exported.
.RE

.sp
.ne 2
.na
\fB\fB-Z\fR\fR
.ad
.RS 6n
Right justify and fill with leading zeros if the first non-blank character is a
digit and the \fB-L\fR option has not been set. Remove leading zeros if the
\fB-L\fR option is also set. If \fIn\fR is \fBnon-zero\fR, it defines the width
of the field, otherwise it is determined by the width of the value of first
assignment.
.sp
The \fB-i\fR attribute cannot be specified with \fB-Z\fR.
.RE

.RE

.sp
.ne 2
.na
\fB\fBulimit [\fR\fB-HSacdfmnpstv\fR\fB] [\fR \fIlimit\fR\fB]\fR\fR
.ad
.sp .6
.RS 4n
Set or display a resource limit. Many systems do not support one or more of
these limits. The limit for a specified resource is set when \fIlimit\fR is
specified. The value of \fIlimit\fR can be a number in the unit specified with
each resource, or the value unlimited. When more than one resource is
specified, then the limit name and unit is printed before the value.
.sp
If no option is specified, \fB-f\fR is assumed.
.sp
The following are the available resource limits:
.sp
.ne 2
.na
\fB\fB-a\fR\fR
.ad
.RS 6n
Lists all of the current resource limits.
.RE

.sp
.ne 2
.na
\fB\fB-c\fR\fR
.ad
.RS 6n
The number of 512-byte blocks on the size of core dumps.
.RE
```

```
.sp
.ne 2
.na
\fB\fB-d\fR\fR
.ad
.RS 6n
The number of Kbytes on the size of the data area.
.RE

.sp
.ne 2
.na
\fB\fB-f\fR\fR
.ad
.RS 6n
The number of 512-byte blocks on files that can be written by the current
process or by child processes (files of any size can be read).
.RE

.sp
.ne 2
.na
\fB\fB-H\fR\fR
.ad
.RS 6n
Specifies a hard limit for the specified resource.
.sp
A hard limit cannot be increased once it is set.
.sp
If neither the \fB-H\fR nor \fB-S\fR option is specified, the limit applies to
both. The current resource limit is printed when \fIlimit\fR is omitted. In
this case, the soft limit is printed unless \fB-H\fR is specified.
.RE

.sp
.ne 2
.na
\fB\fB-m\fR\fR
.ad
.RS 6n
The number of Kbytes on the size of physical memory.
.RE

.sp
.ne 2
.na
\fB\fB-n\fR\fR
.ad
.RS 6n
The number of file descriptors plus 1.
.RE

.sp
.ne 2
.na
\fB\fB-p\fR\fR
.ad
.RS 6n
The number of 512-byte blocks for pipe buffering.
.RE

.sp
.ne 2
.na
\fB\fB-s\fR\fR
.ad
```



```
.RS 6n
The number of Kbytes on the size of the stack area.
.RE

.sp
.ne 2
.na
\fB\fB-S\fR\fR
.ad
.RS 6n
Specifies a soft limit for the specified resource.
.sp
A soft limit can be increased up to the value of the hard limit.
.sp
If neither the \fB-H\fR nor \fB-S\fR option is specified, the limit applies to
both. The current resource limit is printed when \fIlimit\fR is omitted. In
this case, the soft limit is printed unless \fB-H\fR is specified.
.RE

.sp
.ne 2
.na
\fB\fB-t\fR\fR
.ad
.RS 6n
The number of CPU seconds to be used by each process.
.RE

.sp
.ne 2
.na
\fB\fB-v\fR\fR
.ad
.RS 6n
The number of Kbytes for virtual memory.
.RE

.RE

.sp
.ne 2
.na
\fB\fBumask\fR \fB[\fR\fB-S\fR\fB]\fR\fB[\fR\fImask\fR\fB]\fR\fR
.ad
.sp .6
.RS 4n
The user file-creation mask is set to \fImask\fR. \fImask\fR can either be an
octal number or a symbolic value as described in \fBchmod\fR(1).
.sp
If a symbolic value is specified, the new \fBumask\fR value is the complement
of the result of applying \fImask\fR to the complement of the previous
\fBumask\fR value. If \fImask\fR is omitted, the current value of the mask is
printed. The \fB-S\fR option causes the mode to be printed as a symbolic value.
Otherwise, the mask is printed in octal.
.sp
See \fBumask\fR(2)
.RE

.sp
.ne 2
.na
\fB\fB+unalias\fR \fB[\fR\fB-a\fR\fB]\fR \fIname\fR\fR
.ad
.sp .6
.RS 4n
The aliases specified by the list of \fIname\fRs are removed from the alias
```

list. The \fB-a\fR option causes all the aliases to be unset.
.RE

.sp
.ne 2
.na
\fB\fB+unset\fR \fB[\fR\fB-fnv\fR\fB]\fR \fIvname\fR\fR
.ad
.sp .6
.RS 4n
The variables specified by the list of \fIvname\fRs are unassigned, i.e., their values and attributes are erased. Read-only variables cannot be unset. If the \fB-f\fR option is set, then the names refer to function names. If the \fB-v\fR option is set, then the names refer to variable names. The \fB-f\fR option overrides \fB-v\fR. If \fB-n\fR is set and \fIname\fR is a name reference, then \fIname\fR is unset rather than the variable that it references. The default is equivalent to \fB-v\fR. Unsetting \fBLINENO\fR, \fBMAILCHECK\fR, \fBOPTARG\fR, \fBOPTIND\fR, \fBRANDOM\fR, \fBSECONDS\fR, \fBTMOUT\fR, and \fB_\fR removes their special meaning even if they are subsequently assigned to.
.RE

.sp
.ne 2
.na
\fB\fBwait\fR \fB[\fR\fIjob\fR\fB]\fR\fR
.ad
.sp .6
.RS 4n
Wait for the specified job and report its termination status. If \fIjob\fR is not specified, then all currently active child processes are waited for. The exit status from this command is that of the last process waited for if \fIjob\fR is specified; otherwise it is zero. See \fBJobs\fR for a description of the format of \fIjob\fR.
.RE

.sp
.ne 2
.na
\fB\fBwhence\fR \fB[\fR\fB-afpv\fR\fB]\fR \fIname ...\fR\fR
.ad
.sp .6
.RS 4n
For each \fIname\fR, indicate how it would be interpreted if used as a command name. The \fB-v\fR option produces a more verbose report. The \fB-f\fR option skips the search for functions. The \fB-p\fR option does a path search for \fIname\fR even if name is an alias, a function, or a reserved word. The \fB-a\fR option is similar to the -v option but causes all interpretations of the specified name to be reported.
.RE

.SS "Invocation"
.sp
.LP
If the shell is invoked by \fBexec\fR(2), and the first character of argument zero (\fB$0\fR) is \fB-\fR, then the shell is assumed to be a login shell and commands are read from \fB/etc/profile\fR and then from either .\fBprofile\fR in the current directory or \fB$HOME/.profile\fR, if either file exists. Next, for interactive shells, commands are read first from \fB/etc/ksh.kshrc\fR, and then from the file named by performing parameter expansion, command substitution, and arithmetic substitution on the value of the environment variable \fBENV\fR if the file exists. If the \fB-s\fR option is not present and \fIarg\fR and a file by the name of \fIarg\fR exists, then it reads and executes this script. Otherwise, if the first \fIarg\fR does not contain a \fB/\fR, a path search is performed on the first \fIarg\fR to determine the name of the script to execute. The script \fIarg\fR must have execute permission and any \fBsetuid\fR and \fBsetgid\fR settings are ignored. If the

script is not found on the path, \fIarg\fR is processed as if it named a built-in command or function.
.sp
.LP
Commands are then read as described, and the following options are interpreted by the shell when it is invoked:
.sp
.ne 2
.na
\fB\fB-c\fR\fR
.ad
.RS 15n
If the \fB-c\fR option is present, then commands are read from the first \fIarg\fR. Any remaining arguments become positional parameters starting at \fB0\fR.
.RE

.sp
.ne 2
.na
\fB\fB-D\fR\fR
.ad
.RS 15n
A list of all double quoted strings that are preceded by a \fB$\fR is printed on standard output and the shell exits. This set of strings is subject to language translation when the locale is not C or POSIX. No commands are executed.
.RE

.sp
.ne 2
.na
\fB\fB-i\fR\fR
.ad
.RS 15n
If the \fB-i\fR option is present or if the shell input and output are attached to a terminal (as told by \fBtcgetattr\fR(3C), this shell is interactive. In this case \fBTERM\fR is ignored (so that \fBkill 0\fR does not kill an interactive shell) and \fBINTR\fR is caught and ignored (so that wait is interruptible). In all cases, \fBQUIT\fR is ignored by the shell.
.RE

.sp
.ne 2
.na
\fB\fB-R\fR \fIfilename\fR\fR
.ad
.RS 15n
The \fB-R\fR \fIfilename\fR option is used to generate a cross reference database that can be used by a separate utility to find definitions and references for variables and commands.
.RE

.sp
.ne 2
.na
\fB\fB-r\fR\fR
.ad
.RS 15n
If the \fB-r\fR option is present, the shell is a restricted shell.
.RE

.sp
.ne 2
.na
\fB\fB-s\fR\fR

```
.ad
.RS 15n
If the \fB-s\fR option is present or if no arguments remain, then commands are
read from the standard input. Shell output, except for the output of the
\fBSpecial Commands\fR listed, is written to file descriptor 2.
.RE

.sp
.LP
The remaining options and arguments are described under the \fBset\fR command.
An optional \fB-\fR as the first argument is ignored.
.SS "\fBrksh93\fR Only"
.sp
.LP
\fBrksh93\fR is used to set up login names and execution environments whose
capabilities are more controlled than those of the standard shell.
.sp
.LP
The actions of \fBrksh93\fR are identical to those of \fBksh93\fR, except that
the following are disallowed:
.RS +4
.TP
.ie t \(bu
.el o
Unsetting the restricted option
.RE
.RS +4
.TP
.ie t \(bu
.el o
Changing directory. See \fBcd\fR(1).
.RE
.RS +4
.TP
.ie t \(bu
.el o
Setting or unsetting the value or attributes of \fBSHELL\fR, \fBENV\fR,
\fBFPATH\fR, or \fBPATH\fR
.RE
.RS +4
.TP
.ie t \(bu
.el o
Specifying path or command names containing \fB/\fR,
.RE
.RS +4
.TP
.ie t \(bu
.el o
Redirecting output (\fB>\fR, \fB>\fR\fB|\fR, \fB<>\fR, and \fB>>\fR).
.RE
.RS +4
.TP
.ie t \(bu
.el o
Adding or deleting built-in commands.
.RE
.RS +4
.TP
.ie t \(bu
.el o
Using \fBcommand\fR \fB-p\fR to invoke a command.
.RE
.sp
.LP
These restrictions are enforced after .\fBprofile\fR and the \fBENV\fR files
```



```
are interpreted.
.sp
.LP
When a command to be executed is found to be a shell procedure, \fBrksh93\fR
invokes \fBksh93\fR to execute it. Thus, it is possible to provide to the
end-user shell procedures that have access to the full power of the standard
shell, while imposing a limited menu of commands. This scheme assumes that the
end-user does not have write and execute permissions in the same directory. The
net effect of these rules is that the writer of the .\fBprofile\fR has complete
control over user actions, by performing guaranteed setup actions and leaving
the user in an appropriate directory (probably not the login directory). The
system administrator often sets up a directory of commands, for example,
\fB/usr/rbin\fR, that can be safely invoked by \fBrksh\fR.
.SH USAGE
.sp
.LP
See \fBlargefile\fR(5) for the description of the behavior of \fBksh93\fR and
\fBrksh93\fR when encountering files greater than or equal to 2 Gbyte ( 2^31
bytes).
.SH EXIT STATUS
.sp
.LP
The following exit values are returned:
.sp
.ne 2
.na
\fB\fBnon-zero\fR\fR
.ad
.sp .6
.RS 4n
Returns \fBnon-zero\fR when errors, such as syntax errors, are detected by the
shell.
.sp
If the shell is being used non-interactively, then execution of the shell file
is abandoned unless the error occurs inside a sub-shell in which case the
sub-shell is abandoned.
.RE

.sp
.ne 2
.na
\fB\fIexit status of last command executed\fR\fR
.ad
.sp .6
.RS 4n
Returns the exit status of the last command executed.
.sp
Run time errors detected by the shell are reported by printing the command or
function name and the error condition. If the line number that the error
occurred on is greater than one, then the line number is also printed in square
brackets (\fB[]\fR) after the command or function name.
.sp
See the \fBksh93 exit\fR command for additional details.
.RE

.SH FILES
.sp
.ne 2
.na
\fB\fB/etc/profile\fR\fR
.ad
.sp .6
.RS 4n
The system initialization file, executed for login shells.
.RE
```

```
6858 .sp
6859 .ne 2
6860 .na
6861 \fB\fB/etc/ksh.kshrc\fR\fR
6862 .ad
6863 .sp .6
6864 .RS 4n
6865 The system wide startup file, executed for interactive shells.
6866 .RE

6868 .sp
6869 .ne 2
6870 .na
6871 \fB\fB$HOME/.profile\fR\fR
6872 .ad
6873 .sp .6
6874 .RS 4n
6875 The personal initialization file, executed for login shells after
6876 \fB/etc/profile\fR.
6877 .RE

6879 .sp
6880 .ne 2
6881 .na
6882 \fB\fB$HOME/.kshrc\fR\fR
6883 .ad
6884 .sp .6
6885 .RS 4n
6886 Default personal initialization file, executed after \fB/etc/ksh.kshrc\fR, for
6887 interactive shells when \fBENV\fR is not set.
6888 .RE

6890 .sp
6891 .ne 2
6892 .na
6893 \fB\fB/etc/suid-profile\fR\fR
6894 .ad
6895 .sp .6
6896 .RS 4n
6897 Alternative initialization file, executed instead of the personal
6898 initialization file when the real and effective user or group id do not match.
6899 .RE

6901 .sp
6902 .ne 2
6903 .na
6904 \fB\fB/dev/null\fR\fR
6905 .ad
6906 .sp .6
6907 .RS 4n
6908 NULL device.
6909 .RE

6911 .SH AUTHORS
6912 .sp
6913 .LP
6914 David Korn, \fBdgk@research.att.com\fR
6915 .SH ATTRIBUTES
6916 .sp
6917 .LP
6918 See \fBattributes\fR(5) for descriptions of the following attributes:
6919 .sp

6921 .sp
6922 .TS
6923 box;
```



```
6924 c | c
6925 l | l .
6926 ATTRIBUTE TYPE  ATTRIBUTE VALUE
6927 _
6928 Interface Stability     See below.
6929 .TE

6931 .sp
6932 .LP
6933 The scripting interface is Uncommitted. The environment variables,
6934 \fB\&.paths\fR feature, and editing modes are Volatile.
6935 .SH SEE ALSO
6936 .sp
6937 .LP
6938 \fBcat\fR(1), \fBcd\fR(1), \fBchmod\fR(1), \fBcut\fR(1), \fBdate\fR(1),
6939 \fBegrep\fR(1), \fBecho\fR(1), \fBegrep\fR(1), \fBenv\fR(1), \fBfgrep\fR(1),
6940 \fBgrep\fR(1), \fBlogin\fR(1), \fBnewgrp\fR(1), \fBpaste\fR(1),
6941 \fBprintf\fR(1), \fBstty\fR(1), \fBtest\fR(1), \fBumask\fR(1), \fBvi\fR(1),
6942 \fBdup\fR(2), \fBexec\fR(2), \fBfork\fR(2), \fBioctl\fR(2), \fBlseek\fR(2),
6943 \fBpathconf\fR(2), \fBpipe\fR(2), \fBsysconf\fR(3C), \fBulimit\fR(2),
6944 \fBumask\fR(2), \fBrand\fR(3C)\fBtcgetattr\fR(3C), \fBwait\fR(3C),
6945 \fBa.out\fR(4), \fBprofile\fR(4), \fBattributes\fR(5), \fBenviron\fR(5),
6946 \fBlargefile\fR(5), \fBstandards\fR(5)
6947 .sp
6948 .LP
6949 Bolsky, Morris I. and Korn, David G., \fIThe New KornShell Command and
6950 Programming Language\fR, Prentice Hall, 1995.
6951 .sp
6952 .LP
6953 \fIPOSIX-Part 2: Shell and Utilities, IEEE Std 1003.2-1992, ISO/IEC 9945-2\fR,
6954 IEEE, 1993.
6955 .SH NOTES
6956 .sp
6957 .LP
6958 \fBksh93\fR scripts should choose shell function names outside the namespace
6959 used by reserved keywords of the ISO C99, C++ and JAVA languages to avoid
6960 collisions with future enhancements to \fBksh93\fR.
6961 .sp
6962 .LP
6963 If a command is executed, and then a command with the same name is installed in
6964 a directory in the search path before the directory where the original command
6965 was found, the shell continues to \fBexec\fR the original command. Use the
6966 \fB-t\fR option of the alias command to correct this situation.
6967 .sp
6968 .LP
6969 Some very old shell scripts contain a caret (\fB^\fR) as a synonym for the pipe
6970 character (\fB|\fR).
6971 .sp
6972 .LP
6973 Using the \fBhist\fR built-in command within a compound command causes the
6974 whole command to disappear from the history file.
6975 .sp
6976 .LP
6977 The built-in command \fB\&.\fR \fIfile\fR reads the whole file before any
6978 commands are executed. \fBalias\fR and \fBunalias\fR commands in the file do
6979 not apply to any commands defined in the file.
6980 .sp
6981 .LP
6982 Traps are not processed while a job is waiting for a foreground process. Thus,
6983 a trap on \fBCHLD\fR is not executed until the foreground job terminates.
6984 .sp
6985 .LP
6986 It is a good idea to leave a space after the comma operator in arithmetic
6987 expressions to prevent the comma from being interpreted as the decimal point
6988 character in certain locales.
6989 .sp
```

```
6990 .LP
6991 There might be some restrictions on creating a \fB\&.paths\fR file which is
6992 portable across other operating systems.
6993 .sp
6994 .LP
6995 If the system supports the 64-bit instruction set, \fB/bin/ksh93\fR executes
6996 the 64-bit version of \fBksh93\fR.
```

```
**********************************************************
   59046 Tue Sep 10 18:35:18 2013
new/usr/src/man/man1/ld.1
4023 - Typo in file(1) manpage and various others
**********************************************************
```

```
   1 '\" te
   2 .\" Copyright 1989 AT&T
   3 .\" Copyright (c) 2009, Sun Microsystems, Inc. All Rights Reserved
   4 .\" Copyright (c) 2012, Joyent, Inc. All Rights Reserved
   5 .\" The contents of this file are subject to the terms of the Common Development
   6 .\"  See the License for the specific language governing permissions and limitat
   7 .\" the fields enclosed by brackets "[]" replaced with your own identifying info
   8 .TH LD 1 "Sep 10, 2013"
   8 .TH LD 1 "Oct 24, 2012"
   9 .SH NAME
  10 ld \- link-editor for object files
  11 .SH SYNOPSIS
  12 .LP
  13 .nf
  14 \fBld\fR [\fB-32\fR | \fB-64\fR] [\fB-a\fR | \fB-r\fR] [\fB-b\fR] [\fB-B\fRdirec
  15 [\fB-B\fR dynamic | static] [\fB-B\fR eliminate] [\fB-B\fR group] [\fB-B\fR loca
  16 [\fB-B\fR reduce] [\fB-B\fR symbolic] [\fB-c\fR \fIname\fR] [\fB-C\fR] [\fB-d\fR
  17 [\fB-D\fR \fItoken\fR,...] [\fB-e\fR \fIepsym\fR] [\fB-f\fR \fIname\fR | \fB-F\f
  18 [\fB-i\fR] [\fB-I\fR \fIname\fR] [\fB-l\fR \fIx\fR] [\fB-L\fR \fIpath\fR] [\fB-m
  19 [\fB-N\fR \fIstring\fR] [\fB-o\fR \fIoutfile\fR] [\fB-p\fR \fIauditlib\fR] [\fB-
  20 [\fB-Q\fR y | n] [\fB-R\fR \fIpath\fR] [\fB-s\fR] [\fB-S\fR \fIsupportlib\fR] [\
  21 [\fB-u\fR \fIsymname\fR] [\fB-V\fR] [\fB-Y P\fR\fI,dirlist\fR] [\fB-z\fR absexec
  22 [\fB-z\fR allextract | defaultextract | weakextract ] [\fB-z\fR altexec64]
  23 [\fB-z\fR assert-deflib ] [ \fB-z\fR assert-deflib=\fIlibname\fR ]
  24 [\fB-z\fR combreloc | nocombreloc ] [\fB-z\fR defs | nodefs]
  25 [\fB-z\fR direct | nodirect] [\fB-z\fR endfiltee]
  26 [\fB-z\fR fatal-warnings | nofatal-warnings ] [\fB-z\fR finiarray=\fIfunction\fR
  27 [\fB-z\fR globalaudit] [\fB-z\fR groupperm | nogroupperm]
  28 [\fB-z\fR guidance[=\fIid1\fR,\fIid2\fR...] [\fB-z\fR help ]
  29 [\fB-z\fR ignore | record] [\fB-z\fR initarray=\fIfunction\fR] [\fB-z\fR initfir
  30 [\fB-z\fR interpose] [\fB-z\fR lazyload | nolazyload]
  31 [\fB-z\fR ld32=\fIarg1\fR,\fIarg2\fR,...] [\fB-z\fR ld64=\fIarg1\fR,\fIarg2\fR,.
  32 [\fB-z\fR loadfltr] [\fB-z\fR muldefs] [\fB-z\fR nocompstrtab] [\fB-z\fR nodefau
  33 [\fB-z\fR nodelete] [\fB-z\fR nodlopen] [\fB-z\fR nodump] [\fB-z\fR noldynsym]
  34 [\fB-z\fR nopartial] [\fB-z\fR noversion] [\fB-z\fR now] [\fB-z\fR origin]
  35 [\fB-z\fR preinitarray=\fIfunction\fR] [\fB-z\fR redlocsym] [\fB-z\fR relaxreloc
  36 [\fB-z\fR rescan-now] [\fB-z\fR recan] [\fB-z\fR rescan-start \fI\&...\fR \fB-z\
  37 [\fB-z\fR target=sparc|x86] [\fB-z\fR text | textwarn | textoff]
  38 [\fB-z\fR verbose] [\fB-z\fR wrap=\fIsymbol\fR] \fIfilename\fR...
  39 .fi

  41 .SH DESCRIPTION
  42 .sp
  43 .LP
  44 The link-editor, \fBld\fR, combines relocatable object files by resolving
  45 symbol references to symbol definitions, together with performing relocations.
  46 \fBld\fR operates in two modes, static or dynamic, as governed by the \fB-d\fR
  47 option. In all cases, the output of \fBld\fR is left in the file \fBa.out\fR by
  48 default. See NOTES.
  49 .sp
  50 .LP
  51 In dynamic mode, \fB-dy\fR, the default, relocatable object files that are
  52 provided as arguments are combined to produce an executable object file. This
  53 file is linked at execution with any shared object files that are provided as
  54 arguments. If the \fB-G\fR option is specified, relocatable object files are
  55 combined to produce a shared object. Without the \fB-G\fR option, a dynamic
  56 executable is created.
  57 .sp
  58 .LP
  59 In static mode, \fB-dn\fR, relocatable object files that are provided as
  60 arguments are combined to produce a static executable file. If the \fB-r\fR
```



```
  61 option is specified, relocatable object files are combined to produce one
  62 relocatable object file. See \fBStatic Executables\fR.
  63 .sp
  64 .LP
  65 Dynamic linking is the most common model for combining relocatable objects, and
  66 the eventual creation of processes within Solaris. This environment tightly
  67 couples the work of the link-editor and the runtime linker, \fBld.so.1\fR(1).
  68 Both of these utilities, together with their related technologies and
  69 utilities, are extensively documented in the \fILinker and Libraries Guide\fR.
  70 .sp
  71 .LP
  72 If any argument is a library, \fBld\fR by default searches the library exactly
  73 once at the point the library is encountered on the argument list. The library
  74 can be either a shared object or relocatable archive. See \fBar.h\fR(3HEAD)).
  75 .sp
  76 .LP
  77 A shared object consists of an indivisible, whole unit that has been generated
  78 by a previous link-edit of one or more input files. When the link-editor
  79 processes a shared object, the entire contents of the shared object become a
  80 logical part of the resulting output file image. The shared object is not
  81 physically copied during the link-edit as its actual inclusion is deferred
  82 until process execution. This logical inclusion means that all symbol entries
  83 defined in the shared object are made available to the link-editing process.
  84 See Chapter 4, \fIShared Objects,\fR in \fILinker and Libraries Guide\fR
  85 .sp
  86 .LP
  87 For an archive library, \fBld\fR loads only those routines that define an
  88 unresolved external reference. \fBld\fR searches the symbol table of the
  89 archive library sequentially to resolve external references that can be
  90 satisfied by library members. This search is repeated until no external
  91 references can be resolved by the archive. Thus, the order of members in the
  92 library is functionally unimportant, unless multiple library members exist that
  93 define the same external symbol. Archive libraries that have interdependencies
  94 can require multiple command line definitions, or the use of one of the
  95 \fB-z\fR \fBrescan\fR options. See \fIArchive Processing\fR in \fILinker and
  96 Libraries Guide\fR.
  97 .sp
  98 .LP
  99 \fBld\fR is a cross link-editor, able to link 32-bit objects or 64-bit objects,
 100 for Sparc or x86 targets. \fBld\fR uses the \fBELF\fR class and machine type of
 101 the first relocatable object on the command line to govern the mode in which to
 102 operate. The mixing of 32-bit objects and 64-bit objects is not permitted.
 103 Similarly, only objects of a single machine type are allowed. See the
 104 \fB-32\fR, \fB-64\fR and \fB-z target\fR options, and the \fBLD_NOEXEC_64\fR
 105 environment variable.
 106 .SS "Static Executables"
 107 .sp
 108 .LP
 109 The creation of static executables has been discouraged for many releases. In
 110 fact, 64-bit system archive libraries have never been provided. Because a
 111 static executable is built against system archive libraries, the executable
 112 contains system implementation details. This self-containment has a number of
 113 drawbacks.
 114 .RS +4
 115 .TP
 116 .ie t \(bu
 117 .el o
 118 The executable is immune to the benefits of system patches delivered as shared
 119 objects. The executable therefore, must be rebuilt to take advantage of many
 120 system improvements.
 121 .RE
 122 .RS +4
 123 .TP
 124 .ie t \(bu
 125 .el o
 126 The ability of the executable to run on future releases can be compromised.
```

```
.RE
.RS +4
.TP
.ie t \(bu
.el o
The duplication of system implementation details negatively affects system
performance.
.RE
.sp
.LP
With Solaris 10, 32-bit system archive libraries are no longer provided.
Without these libraries, specifically \fBlibc.a\fR, the creation of static
executables is no longer achievable without specialized system knowledge.
However, the capability of \fBld\fR to process static linking options, and the
processing of archive libraries, remains unchanged.
.SH OPTIONS
.sp
.LP
The following options are supported.
.sp
.ne 2
.na
\fB\fB-32\fR | \fB-64\fR\fR
.ad
.sp .6
.RS 4n
Creates a 32-bit, or 64-bit object.
.sp
By default, the class of the object being generated is determined from the
first \fBELF\fR object processed from the command line. If no objects are
specified, the class is determined by the first object encountered within the
first archive processed from the command line. If there are no objects or
archives, the link-editor creates a 32-bit object.
.sp
The \fB-64\fR option is required to create a 64-bit object solely from a
mapfile.
.sp
This \fB-32\fR or \fB-64\fR options can also be used in the rare case of
linking entirely from an archive that contains a mixture of 32 and 64-bit
objects. If the first object in the archive is not the class of the object that
is required to be created, then the \fB-32\fR or \fB-64\fR option can be used
to direct the link-editor. See \fIThe 32-bit link-editor and 64-bit
link-editor\fR in \fILinker and Libraries Guide\fR.
.RE

.sp
.ne 2
.na
\fB\fB-a\fR\fR
.ad
.sp .6
.RS 4n
In static mode only, produces an executable object file. Undefined references
are not permitted. This option is the default behavior for static mode. The
\fB-a\fR option can not be used with the \fB-r\fR option. See \fBStatic
Executables\fR under DESCRIPTION.
.RE

.sp
.ne 2
.na
\fB\fB-b\fR\fR
.ad
.sp .6
.RS 4n
In dynamic mode only, provides no special processing for dynamic executable
```



```
relocations that reference symbols in shared objects. Without the \fB-b\fR
option, the link-editor applies techniques within a dynamic executable so that
the text segment can remain read-only. One technique is the creation of special
position-independent relocations for references to functions that are defined
in shared objects. Another technique arranges for data objects that are defined
in shared objects to be copied into the memory image of an executable at
runtime.
.sp
The \fB-b\fR option is intended for specialized dynamic objects and is not
recommended for general use. Its use suppresses all specialized processing
required to ensure an object's shareability, and can even prevent the
relocation of 64-bit executables.
.RE

.sp
.ne 2
.na
\fB\fB-B\fR \fBdirect\fR | \fBnodirect\fR\fR
.ad
.sp .6
.RS 4n
These options govern direct binding. \fB-B\fR \fBdirect\fR establishes direct
binding information by recording the relationship between each symbol reference
together with the dependency that provides the definition. In addition, direct
binding information is established between each symbol reference and an
associated definition within the object being created. The runtime linker uses
this information to search directly for a symbol in the associated object
rather than to carry out a default symbol search.
.sp
Direct binding information can only be established to dependencies specified
with the link-edit. Thus, you should use the \fB-z\fR \fBdefs\fR option.
Objects that wish to interpose on symbols in a direct binding environment
should identify themselves as interposers with the \fB-z\fR \fBinterpose\fR
option. The use of \fB-B\fR \fBdirect\fR enables \fB-z\fR \fBlazyload\fR for
all dependencies.
.sp
The \fB-B\fR \fBnodirect\fR option prevents any direct binding to the
interfaces offered by the object being created. The object being created can
continue to directly bind to external interfaces by specifying the \fB-z\fR
\fBdirect\fR option. See Appendix D, \fIDirect Bindings,\fR in \fILinker and
Libraries Guide\fR.
.RE

.sp
.ne 2
.na
\fB\fB-B\fR \fBdynamic\fR | \fBstatic\fR\fR
.ad
.sp .6
.RS 4n
Options governing library inclusion. \fB-B\fR \fBdynamic\fR is valid in dynamic
mode only. These options can be specified any number of times on the command
line as toggles: if the \fB-B\fR \fBstatic\fR option is given, no shared
objects are accepted until \fB-B\fR \fBdynamic\fR is seen. See the \fB-l\fR
option.
.RE

.sp
.ne 2
.na
\fB\fB-B\fR \fBeliminate\fR\fR
.ad
.sp .6
.RS 4n
Causes any global symbols, not assigned to a version definition, to be
eliminated from the symbol table. Version definitions can be supplied by means
```

of a \fBmapfile\fR to indicate the global symbols that should remain visible in the generated object. This option achieves the same symbol elimination as the \fIauto-elimination\fR directive that is available as part of a \fBmapfile\fR version definition. This option can be useful when combining versioned and non-versioned relocatable objects. See also the \fB-B\fR \fBlocal\fR option and the \fB-B\fR \fBreduce\fR option. See \fIDefining Additional Symbols with a mapfile\fR in \fILinker and Libraries Guide\fR.
.RE

.sp
.ne 2
.na
\fB\fB-B\fR \fBgroup\fR\fR
.ad
.sp .6
.RS 4n
Establishes a shared object and its dependencies as a group. Objects within the group are bound to other members of the group at runtime. This mode is similar to adding the object to the process by using \fBdlopen\fR(3C) with the \fBRTLD_GROUP\fR mode. An object that has an explicit dependency on a object identified as a group, becomes a member of the group.
.sp
As the group must be self contained, use of the \fB-B\fR \fBgroup\fR option also asserts the \fB-z\fR \fBdefs\fR option.
.RE

.sp
.ne 2
.na
\fB\fB-B\fR \fBlocal\fR\fR
.ad
.sp .6
.RS 4n
Causes any global symbols, not assigned to a version definition, to be reduced to local. Version definitions can be supplied by means of a \fBmapfile\fR to indicate the global symbols that should remain visible in the generated object. This option achieves the same symbol reduction as the \fIauto-reduction\fR directive that is available as part of a \fBmapfile\fR version definition. This option can be useful when combining versioned and non-versioned relocatable objects. See also the \fB-B\fR \fBeliminate\fR option and the \fB-B\fR \fBreduce\fR option. See \fIDefining Additional Symbols with a mapfile\fR in \fILinker and Libraries Guide\fR.
.RE

.sp
.ne 2
.na
\fB\fB-B\fR \fBreduce\fR\fR
.ad
.sp .6
.RS 4n
When generating a relocatable object, causes the reduction of symbolic information defined by any version definitions. Version definitions can be supplied by means of a \fBmapfile\fR to indicate the global symbols that should remain visible in the generated object. By default, when a relocatable object is generated, version definitions are only recorded in the output image. The actual reduction of symbolic information is carried out when the object is used in the construction of a dynamic executable or shared object. The \fB-B\fR \fBreduce\fR option is applied automatically when a dynamic executable or shared object is created.
.RE

.sp
.ne 2
.na
\fB\fB-B\fR \fBsymbolic\fR\fR

.ad
.sp .6
.RS 4n
In dynamic mode only. When building a shared object, binds references to global symbols to their definitions, if available, within the object. Normally, references to global symbols within shared objects are not bound until runtime, even if definitions are available. This model allows definitions of the same symbol in an executable or other shared object to override the object’s own definition. \fBld\fR issues warnings for undefined symbols unless \fB-z\fR \fBdefs\fR overrides.
.sp
The \fB-B\fR \fBsymbolic\fR option is intended for specialized dynamic objects and is not recommended for general use. To reduce the runtime relocation processing that is required an object, the creation of a version definition is recommended.
.RE

.sp
.ne 2
.na
\fB\fB-c\fR \fIname\fR\fR
.ad
.sp .6
.RS 4n
Records the configuration file \fIname\fR for use at runtime. Configuration files can be employed to alter default search paths, provide a directory cache, together with providing alternative object dependencies. See \fBcrle\fR(1).
.RE

.sp
.ne 2
.na
\fB\fB-C\fR\fR
.ad
.sp .6
.RS 4n
Demangles C++ symbol names displayed in diagnostic messages.
.RE

.sp
.ne 2
.na
\fB\fB-d\fR \fBy\fR | \fBn\fR\fR
.ad
.sp .6
.RS 4n
When \fB-d\fR \fBy\fR, the default, is specified, \fBld\fR uses dynamic linking. When \fB-d\fR \fBn\fR is specified, \fBld\fR uses static linking. See \fBStatic Executables\fR under DESCRIPTION, and \fB-B\fR \fBdynamic\fR|\fBstatic\fR.
.RE

.sp
.ne 2
.na
\fB\fB-D\fR \fItoken\fR,...\fR
.ad
.sp .6
.RS 4n
Prints debugging information as specified by each \fItoken\fR, to the standard error. The special token \fBhelp\fR indicates the full list of tokens available. See \fIDebugging Aids\fR in \fILinker and Libraries Guide\fR.
.RE

.sp
.ne 2

```
391 .na
392 \fB\fB-e\fR \fIepsym\fR\fR
393 .ad
394 .br
395 .na
396 \fB\fB--entry\fR \fIepsym\fR\fR
397 .ad
398 .sp .6
399 .RS 4n
400 Sets the entry point address for the output file to be the symbol \fIepsym\fR.
401 .RE

403 .sp
404 .ne 2
405 .na
406 \fB\fB-f\fR \fIname\fR\fR
407 .ad
408 .br
409 .na
410 \fB\fB--auxiliary\fR \fIname\fR\fR
411 .ad
412 .sp .6
413 .RS 4n
414 Useful only when building a shared object. Specifies that the symbol table of
415 the shared object is used as an auxiliary filter on the symbol table of the
416 shared object specified by \fIname\fR. Multiple instances of this option are
417 allowed. This option can not be combined with the \fB-F\fR option. See
418 \fIGenerating Auxiliary Filters\fR in \fILinker and Libraries Guide\fR.
419 .RE

421 .sp
422 .ne 2
423 .na
424 \fB\fB-F\fR \fIname\fR\fR
425 .ad
426 .br
427 .na
428 \fB\fB--filter\fR \fIname\fR\fR
429 .ad
430 .sp .6
431 .RS 4n
432 Useful only when building a shared object. Specifies that the symbol table of
433 the shared object is used as a filter on the symbol table of the shared object
434 specified by \fIname\fR. Multiple instances of this option are allowed. This
435 option can not be combined with the \fB-f\fR option. See \fIGenerating Standard
436 Filters\fR in \fILinker and Libraries Guide\fR.
437 .RE

439 .sp
440 .ne 2
441 .na
442 \fB\fB-G\fR\fR
443 .ad
444 .br
445 .na
446 \fB\fB-shared\fR\fR
447 .ad
448 .sp .6
449 .RS 4n
450 In dynamic mode only, produces a shared object. Undefined symbols are allowed.
451 See Chapter 4, \fIShared Objects,\fR in \fILinker and Libraries Guide\fR.
452 .RE

454 .sp
455 .ne 2
456 .na
```



```
457 \fB\fB-h\fR \fIname\fR\fR
458 .ad
459 .br
460 .na
461 \fB\fB--soname\fR \fIname\fR\fR
462 .ad
463 .sp .6
464 .RS 4n
465 In dynamic mode only, when building a shared object, records \fIname\fR in the
466 object's dynamic section. \fIname\fR is recorded in any dynamic objects that
467 are linked with this object rather than the object's file system name.
468 Accordingly, \fIname\fR is used by the runtime linker as the name of the shared
469 object to search for at runtime. See \fIRecording a Shared Object Name\fR in
470 \fILinker and Libraries Guide\fR.
471 .RE

473 .sp
474 .ne 2
475 .na
476 \fB\fB-i\fR\fR
477 .ad
478 .sp .6
479 .RS 4n
480 Ignores \fBLD_LIBRARY_PATH\fR. This option is useful when an
481 \fBLD_LIBRARY_PATH\fR setting is in effect to influence the runtime library
482 search, which would interfere with the link-editing being performed.
483 .RE

485 .sp
486 .ne 2
487 .na
488 \fB\fB-I\fR \fIname\fR\fR
489 .ad
490 .br
491 .na
492 \fB\fB--dynamic-linker\fR \fIname\fR\fR
493 .ad
494 .sp .6
495 .RS 4n
496 When building an executable, uses \fIname\fR as the path name of the
497 interpreter to be written into the program header. The default in static mode
498 is no interpreter. In dynamic mode, the default is the name of the runtime
499 linker, \fBld.so.1\fR(1). Either case can be overridden by \fB-I\fR \fIname\fR.
500 \fBexec\fR(2) loads this interpreter when the \fBa.out\fR is loaded, and passes
501 control to the interpreter rather than to the \fBa.out\fR directly.
502 .RE

504 .sp
505 .ne 2
506 .na
507 \fB\fB-l\fR \fIx\fR\fR
508 .ad
509 .br
510 .na
511 \fB\fB--library\fR \fIx\fR\fR
512 .ad
513 .sp .6
514 .RS 4n
515 Searches a library \fBlib\fR\fIx\fR\fB\&.so\fR or \fBlib\fR\fIx\fR\fB\&.a\fR,
516 the conventional names for shared object and archive libraries, respectively.
517 In dynamic mode, unless the \fB-B\fR \fBstatic\fR option is in effect, \fBld\fR
518 searches each directory specified in the library search path for a
519 \fBlib\fR\fIx\fR\fB\&.so\fR or \fBlib\fR\fIx\fR\fB\&.a\fR file. The directory
520 search stops at the first directory containing either. \fBld\fR chooses the
521 file ending in \fB\&.so\fR if \fB-l\fR\fIx\fR expands to two files with names
522 of the form \fBlib\fR\fIx\fR\fB\&.so\fR and \fBlib\fR\fIx\fR\fB\&.a\fR. If no
```

\fBlib\fR\fIx\fR\fB\&.so\fR is found, then \fBld\fR accepts
\fBlib\fR\fIx\fR\fB\&.a\fR. In static mode, or when the \fB-B\fR \fBstatic\fR
option is in effect, \fBld\fR selects only the file ending in \fB\&.a\fR.
\fBld\fR searches a library when the library is encountered, so the placement
of \fB-l\fR is significant. See \fILinking With Additional Libraries\fR in
\fILinker and Libraries Guide\fR.
.RE

.sp
.ne 2
.na
\fB\fB-L\fR \fIpath\fR\fR
.ad
.br
.na
\fB\fB--library-path\fR \fIpath\fR\fR
.ad
.sp .6
.RS 4n
Adds \fIpath\fR to the library search directories. \fBld\fR searches for
libraries first in any directories specified by the \fB-L\fR options and then
in the standard directories. This option is useful only if the option precedes
the \fB-l\fR options to which the \fB-L\fR option applies. See \fIDirectories
Searched by the Link-Editor\fR in \fILinker and Libraries Guide\fR.
.sp
The environment variable \fBLD_LIBRARY_PATH\fR can be used to supplement the
library search path, however the \fB-L\fR option is recommended, as the
environment variable is also interpreted by the runtime environment. See
\fBLD_LIBRARY_PATH\fR under ENVIRONMENT VARIABLES.
.RE

.sp
.ne 2
.na
\fB\fB-m\fR\fR
.ad
.sp .6
.RS 4n
Produces a memory map or listing of the input/output sections, together with
any non-fatal multiply-defined symbols, on the standard output.
.RE

.sp
.ne 2
.na
\fB\fB-M\fR \fImapfile\fR\fR
.ad
.sp .6
.RS 4n
Reads \fImapfile\fR as a text file of directives to \fBld\fR. This option can
be specified multiple times. If \fImapfile\fR is a directory, then all regular
files, as defined by \fBstat\fR(2), within the directory are processed. See
Chapter 9, \fIMapfile Option,\fR in \fILinker and Libraries Guide\fR. Example
mapfiles are provided in \fB/usr/lib/ld\fR. See FILES.
.RE

.sp
.ne 2
.na
\fB\fB-N\fR \fIstring\fR\fR
.ad
.sp .6
.RS 4n
This option causes a \fBDT_NEEDED\fR entry to be added to the \fB\&.dynamic\fR
section of the object being built. The value of the \fBDT_NEEDED\fR string is
the \fIstring\fR that is specified on the command line. This option is position

dependent, and the \fBDT_NEEDED\fR \fB\&.dynamic\fR entry is relative to the
other dynamic dependencies discovered on the link-edit line. This option is
useful for specifying dependencies within device driver relocatable objects
when combined with the \fB-dy\fR and \fB-r\fR options.
.RE

.sp
.ne 2
.na
\fB\fB-o\fR \fIoutfile\fR\fR
.ad
.br
.na
\fB\fB--output\fR \fIoutfile\fR\fR
.ad
.sp .6
.RS 4n
Produces an output object file that is named \fIoutfile\fR. The name of the
default object file is \fBa.out\fR.
.RE

.sp
.ne 2
.na
\fB\fB-p\fR \fIauditlib\fR\fR
.ad
.sp .6
.RS 4n
Identifies an audit library, \fIauditlib\fR. This audit library is used to
audit the object being created at runtime. A shared object identified as
requiring auditing with the \fB-p\fR option, has this requirement inherited by
any object that specifies the shared object as a dependency. See the \fB-P\fR
option. See \fIRuntime Linker Auditing Interface\fR in \fILinker and Libraries
Guide\fR.
.RE

.sp
.ne 2
.na
\fB\fB-P\fR \fIauditlib\fR\fR
.ad
.sp .6
.RS 4n
Identifies an audit library, \fIauditlib\fR. This audit library is used to
audit the dependencies of the object being created at runtime. Dependency
auditing can also be inherited from dependencies that are identified as
requiring auditing. See the \fB-p\fR option, and the \fB-z\fR \fBglobalaudit\fR
option. See \fIRuntime Linker Auditing Interface\fR in \fILinker and Libraries
Guide\fR.
.RE

.sp
.ne 2
.na
\fB\fB-Q\fR \fBy\fR | \fBn\fR\fR
.ad
.sp .6
.RS 4n
Under \fB-Q\fR \fBy\fR, an \fBident\fR string is added to the \fB\&.comment\fR
section of the output file. This string identifies the version of the \fBld\fR
used to create the file. This results in multiple \fBld\fR \fBidents\fR when
there have been multiple linking steps, such as when using \fBld\fR \fB-r\fR.
This identification is identical with the default action of the \fBcc\fR
command. \fB-Q\fR \fBn\fR suppresses version identification. \fB\&.comment\fR
sections can be manipulated by the \fBmcs\fR(1) utility.
.RE

```
.sp
.ne 2
.na
\fB\fB-r\fR\fR
.ad
.br
.na
\fB\fB--relocatable\fR\fR
.ad
.sp .6
.RS 4n
Combines relocatable object files to produce one relocatable object file.
\fBld\fR does not complain about unresolved references. This option cannot be
used with the \fB-a\fR option.
.RE

.sp
.ne 2
.na
\fB\fB-R\fR \fIpath\fR\fR
.ad
.br
.na
\fB\fB-rpath\fR \fIpath\fR\fR
.ad
.sp .6
.RS 4n
A colon-separated list of directories used to specify library search
directories to the runtime linker. If present and not NULL, the path is
recorded in the output object file and passed to the runtime linker. Multiple
instances of this option are concatenated together with each \fIpath\fR
separated by a colon. See \fIDirectories Searched by the Runtime Linker\fR in
\fILinker and Libraries Guide\fR.
.sp
The use of a runpath within an associated object is preferable to setting
global search paths such as through the \fBLD_LIBRARY_PATH\fR environment
variable. Only the runpaths that are necessary to find the objects dependencies
should be recorded. \fBldd\fR(1) can also be used to discover unused runpaths
in dynamic objects, when used with the \fB-U\fR option.
.sp
Various tokens can also be supplied with a runpath that provide a flexible
means of identifying system capabilities or an objects location. See Appendix
C, \fIEstablishing Dependencies with Dynamic String Tokens,\fR in \fILinker and
Libraries Guide\fR. The \fB$ORIGIN\fR token is especially useful in allowing
dynamic objects to be relocated to different locations in the file system.
.RE

.sp
.ne 2
.na
\fB\fB-s\fR\fR
.ad
.br
.na
\fB\fB--strip-all\fR\fR
.ad
.sp .6
.RS 4n
Strips symbolic information from the output file. Any debugging information,
that is, \fB\&.line\fR, \fB\&.debug*\fR, and \fB\&.stab*\fR sections, and their
associated relocation entries are removed. Except for relocatable files, a
symbol table \fBSHT_SYMTAB\fR and its associated string table section are not
created in the output object file. The elimination of a \fBSHT_SYMTAB\fR symbol
table can reduce the \fB\&.stab*\fR debugging information that is generated
using the compiler drivers \fB-g\fR option. See the \fB-z\fR \fBredlocsym\fR
```



```
and \fB-z\fR \fBnoldynsym\fR options.
.RE

.sp
.ne 2
.na
\fB\fB-S\fR \fIsupportlib\fR\fR
.ad
.sp .6
.RS 4n
The shared object \fIsupportlib\fR is loaded with \fBld\fR and given
information regarding the linking process. Shared objects that are defined by
using the \fB-S\fR option can also be supplied using the \fBSGS_SUPPORT\fR
environment variable. See \fILink-Editor Support Interface\fR in \fILinker and
Libraries Guide\fR.
.RE

.sp
.ne 2
.na
\fB\fB-t\fR\fR
.ad
.sp .6
.RS 4n
Turns off the warning for multiply-defined symbols that have different sizes or
different alignments.
.RE

.sp
.ne 2
.na
\fB\fB-u\fR \fIsymname\fR\fR
.ad
.br
.na
\fB\fB--undefined\fR \fIsymname\fR\fR
.ad
.sp .6
.RS 4n
Enters \fIsymname\fR as an undefined symbol in the symbol table. This option is
useful for loading entirely from an archive library. In this instance, an
unresolved reference is needed to force the loading of the first routine. The
placement of this option on the command line is significant. This option must
be placed before the library that defines the symbol. See \fIDefining
Additional Symbols with the u option\fR in \fILinker and Libraries Guide\fR.
.RE

.sp
.ne 2
.na
\fB\fB-V\fR\fR
.ad
.br
.na
\fB\fB--version\fR\fR
.ad
.sp .6
.RS 4n
Outputs a message giving information about the version of \fBld\fR being used.
.RE

.sp
.ne 2
.na
\fB\fB-Y\fR \fBP,\fR\fIdirlist\fR\fR
.ad
```

```
.sp .6
.RS 4n
Changes the default directories used for finding libraries. \fIdirlist\fR is a
colon-separated path list.
.RE

.sp
.ne 2
.na
\fB\fB-z\fR \fBabsexec\fR\fR
.ad
.sp .6
.RS 4n
Useful only when building a dynamic executable. Specifies that references to
external absolute symbols should be resolved immediately instead of being left
for resolution at runtime. In very specialized circumstances, this option
removes text relocations that can result in excessive swap space demands by an
executable.
.RE

.sp
.ne 2
.na
\fB\fB-z\fR \fBallextract\fR | \fBdefaultextract\fR | \fBweakextract\fR\fR
.ad
.br
.na
\fB\fB--whole-archive\fR | \fB--no-whole-archive\fR\fR
.ad
.sp .6
.RS 4n
Alters the extraction criteria of objects from any archives that follow. By
default, archive members are extracted to satisfy undefined references and to
promote tentative definitions with data definitions. Weak symbol references do
not trigger extraction. Under the \fB-z\fR \fBallextract\fR or
\fB--whole-archive\fR options, all archive members are extracted from the
archive. Under \fB-z\fR \fBweakextract\fR, weak references trigger archive
extraction. The \fB-z\fR \fBdefaultextract\fR or \fB--no-whole-archive\fR
options provide a means of returning to the default following use of the former
extract options. See \fIArchive Processing\fR in \fILinker and Libraries
Guide\fR.
.RE

.sp
.ne 2
.na
\fB\fB-z\fR \fBaltexec64\fR\fR
.ad
.sp .6
.RS 4n
Execute the 64-bit \fBld\fR. The creation of very large 32-bit objects can
exhaust the virtual memory that is available to the 32-bit \fBld\fR. The
\fB-z\fR \fBaltexec64\fR option can be used to force the use of the associated
64-bit \fBld\fR. The 64-bit \fBld\fR provides a larger virtual address space
for building 32-bit objects. See \fIThe 32-bit link-editor and 64-bit
link-editor\fR in \fILinker and Libraries Guide\fR.
.RE

.sp
.ne 2
.na
\fB\fB-z\fR \fBcombreloc\fR | \fBnocombreloc\fR\fR
.ad
.sp .6
.RS 4n
By default, \fBld\fR combines multiple relocation sections when building
```



```
executables or shared objects. This section combination differs from
relocatable objects, in which relocation sections are maintained in a
one-to-one relationship with the sections to which the relocations must be
applied. The \fB-z\fR \fBnocombreloc\fR option disables this merging of
relocation sections, and preserves the one-to-one relationship found in the
original relocatable objects.
.sp
\fBld\fR sorts the entries of data relocation sections by their symbol
reference. This sorting reduces runtime symbol lookup. When multiple relocation
sections are combined, this sorting produces the least possible relocation
overhead when objects are loaded into memory, and speeds the runtime loading of
dynamic objects.
.sp
Historically, the individual relocation sections were carried over to any
executable or shared object, and the \fB-z\fR \fBcombreloc\fR option was
required to enable the relocation section merging previously described.
Relocation section merging is now the default. The \fB-z\fR \fBcombreloc\fR
option is still accepted for the benefit of old build environments, but the
option is unnecessary, and has no effect.
.RE

.sp
.ne 2
.na
\fB\fB-z\fR \fBassert-deflib\fR\fR
.ad
.br
.na
\fB\fB-z\fR \fBassert-deflib=\fR\fIlibname\fR\fR
.ad
.sp .6
.RS 4n
Enables warnings that check the location of where libraries passed in with
\fB-l\fR are found. If the link-editor finds a library on its default search
path it will emit a warning. This warning can be made fatal in conjunction with
the option \fB-z fatal-warnings\fR. Passing \fIlibname\fR white lists a library
from this check. The library must be the full name of the library, e.g.
\fIlibc.so\fR. To white list multiple libraries, the \fB-z
assert-deflib=\fR\fIlibname\fR option can be repeated multiple times. This
option is useful when trying to build self-contained objects where a referenced
library might exist in the default system library path and in alternate paths
specified by \fB-L\fR, but you only want the alternate paths to be used.
.RE

.sp
.ne 2
.na
\fB\fB-z\fR \fBdefs\fR | \fBnodefs\fR\fR
.ad
.br
.na
\fB\fB--no-undefined\fR\fR
.ad
.sp .6
.RS 4n
The \fB-z\fR \fBdefs\fR option and the \fB--no-undefined\fR option force a
fatal error if any undefined symbols remain at the end of the link. This mode
is the default when an executable is built. For historic reasons, this mode is
\fBnot\fR the default when building a shared object. Use of the \fB-z\fR
\fBdefs\fR option is recommended, as this mode assures the object being built
is self-contained. A self-contained object has all symbolic references resolved
internally, or to the object’s immediate dependencies.
.sp
The \fB-z\fR \fBnodefs\fR option allows undefined symbols. For historic
reasons, this mode is the default when a shared object is built. When used with
executables, the behavior of references to such undefined symbols is
```

```
unspecified. Use of the \fB-z\fR \fBnodefs\fR option is not recommended.
.RE

.sp
.ne 2
.na
\fB\fB-z\fR \fBdirect\fR | \fBnodirect\fR\fR
.ad
.sp .6
.RS 4n
Enables or disables direct binding to any dependencies that follow on the
command line. These options allow finer control over direct binding than the
global counterpart \fB-B\fR \fBdirect\fR. The \fB-z\fR \fBdirect\fR option also
differs from the \fB-B\fR \fBdirect\fR option in the following areas. Direct
binding information is not established between a symbol reference and an
associated definition within the object being created. Lazy loading is not
enabled.
.RE

.sp
.ne 2
.na
\fB\fB-z\fR \fBendfiltee\fR\fR
.ad
.sp .6
.RS 4n
Marks a filtee so that when processed by a filter, the filtee terminates any
further filtee searches by the filter. See \fIReducing Filtee Searches\fR in
\fILinker and Libraries Guide\fR.
.RE

.sp
.ne 2
.na
\fB\fB-z\fR \fBfatal-warnings\fR | \fBnofatal-warnings\fR\fR
.ad
.br
.na
\fB\fB--fatal-warnings\fR | \fB--no-fatal-warnings\fR
.ad
.sp .6
.RS 4n
Controls the behavior of warnings emitted from the link-editor. Setting \fB-z
fatal-warnings\fR promotes warnings emitted by the link-editor to fatal errors
that will cause the link-editor to fail before linking. \fB-z
nofatal-warnings\fR instead demotes these warnings such that they will not cause
the link-editor to exit prematurely.
.RE


.sp
.ne 2
.na
\fB\fB-z\fR \fBfiniarray=\fR\fIfunction\fR\fR
.ad
.sp .6
.RS 4n
Appends an entry to the \fB\&.finiarray\fR section of the object being built.
If no \fB\&.finiarray\fR section is present, a section is created. The new
entry is initialized to point to \fIfunction\fR. See \fIInitialization and
Termination Sections\fR in \fILinker and Libraries Guide\fR.
.RE

.sp
.ne 2
.na
```



```
\fB\fB-z\fR \fBglobalaudit\fR\fR
.ad
.sp .6
.RS 4n
This option supplements an audit library definition that has been recorded with
the \fB-P\fR option. This option is only meaningful when building a dynamic
executable. Audit libraries that are defined within an object with the \fB-P\fR
option typically allow for the auditing of the immediate dependencies of the
object. The \fB-z\fR \fBglobalaudit\fR promotes the auditor to a global
auditor, thus allowing the auditing of all dependencies. See \fIInvoking the
Auditing Interface\fR in \fILinker and Libraries Guide\fR.
.sp
An auditor established with the \fB-P\fR option and the \fB-z\fR
\fBglobalaudit\fR option, is equivalent to the auditor being established with
the \fBLD_AUDIT\fR environment variable. See \fBld.so.1\fR(1).
.RE

.sp
.ne 2
.na
\fB\fB-z\fR \fBgroupperm\fR | \fBnogroupperm\fR\fR
.ad
.sp .6
.RS 4n
Assigns, or deassigns each dependency that follows to a unique group. The
assignment of a dependency to a group has the same effect as if the dependency
had been built using the \fB-B\fR \fBgroup\fR option.
.RE

.sp
.ne 2
.na
\fB-z\fR \fBguidance\fR[=\fIid1\fR,\fIid2\fR...]
.ad
.sp .6
.RS 4n
Give messages suggesting link-editor features that could improve the resulting
dynamic object.
.LP
Specific classes of suggestion can be silenced by specifying an optional comma s
list of guidance identifiers.
.LP
The current classes of suggestion provided are:

.sp
.ne 2
.na
Enable use of direct binding
.ad
.sp .6
.RS 4n
Suggests that \fB-z direct\fR or \fB-B direct\fR be present prior to any
specified dependency.  This allows predictable symbol binding at runtime.

Can be disabled with \fB-z guidance=nodirect\fR
.RE

.sp
.ne 2
.na
Enable lazy dependency loading
.ad
.sp .6
.RS 4n
Suggests that \fB-z lazyload\fR be present prior to any specified dependency.
This allows the dynamic object to be loaded more quickly.
```

```
1052 Can be disabled with \fB-z guidance=nolazyload\fR.
1053 .RE

1055 .sp
1056 .ne 2
1057 .na
1058 Shared objects should define all their dependencies.
1059 .ad
1060 .sp .6
1061 .RS 4n
1062 Suggests that \fB-z defs\fR be specified on the link-editor command line.
1063 Shared objects that explicitly state all their dependencies behave more
1064 predictably when used.

1066 Can be be disabled with \fB-z guidance=nodefs\fR
1067 .RE

1069 .sp
1070 .ne 2
1071 .na
1072 Version 2 mapfile syntax
1073 .ad
1074 .sp .6
1075 .RS 4n
1076 Suggests that any specified mapfiles use the more readable version 2 syntax.

1078 Can be disabled with \fB-z guidance=nomapfile\fR.
1079 .RE

1081 .sp
1082 .ne 2
1083 .na
1084 Read-only text segment
1085 .ad
1086 .sp .6
1087 .RS 4n
1088 Should any runtime relocations within the text segment exist, suggests that
1089 the object be compiled with position independent code (PIC).  Keeping large
1090 allocatable sections read-only allows them to be shared between processes
1091 using a given shared object.

1093 Can be disabled with \fB-z guidance=notext\fR
1094 .RE

1096 .sp
1097 .ne 2
1098 .na
1099 No unused dependencies
1100 .ad
1101 .sp .6
1102 .RS 4n
1103 Suggests that any dependency not referenced by the resulting dynamic object be
1104 removed from the link-editor command line.

1106 Can be disabled with \fB-z guidance=nounused\fR.
1107 .RE
1108 .RE

1110 .sp
1111 .ne 2
1112 .na
1113 \fB\fB-z\fR \fBhelp\fR\fR
1114 .ad
1115 .br
1116 .na
```



```
1117 \fB\fB--help\fR\fR
1118 .ad
1119 .sp .6
1120 .RS 4n
1121 Print a summary of the command line options on the standard output and exit.
1122 .RE

1124 .sp
1125 .ne 2
1126 .na
1127 \fB\fB-z\fR \fBignore\fR | \fBrecord\fR\fR
1128 .ad
1129 .sp .6
1130 .RS 4n
1131 Ignores, or records, dynamic dependencies that are not referenced as part of
1132 the link-edit. Ignores, or records, unreferenced \fBELF\fR sections from the
1133 relocatable objects that are read as part of the link-edit. By default,
1134 \fB-z\fR \fBrecord\fR is in effect.
1135 .sp
1136 If an \fBELF\fR section is ignored, the section is eliminated from the output
1137 file being generated. A section is ignored when three conditions are true. The
1138 eliminated section must contribute to an allocatable segment. The eliminated
1139 section must provide no global symbols. No other section from any object that
1140 contributes to the link-edit, must reference an eliminated section.
1141 .RE

1143 .sp
1144 .ne 2
1145 .na
1146 \fB\fB-z\fR \fBinitarray=\fR\fIfunction\fR\fR
1147 .ad
1148 .sp .6
1149 .RS 4n
1150 Appends an entry to the \fB\&.initarray\fR section of the object being built.
1151 If no \fB\&.initarray\fR section is present, a section is created. The new
1152 entry is initialized to point to \fIfunction\fR. See \fIInitialization and
1153 Termination Sections\fR in \fILinker and Libraries Guide\fR.
1154 .RE

1156 .sp
1157 .ne 2
1158 .na
1159 \fB\fB-z\fR \fBinitfirst\fR\fR
1160 .ad
1161 .sp .6
1162 .RS 4n
1163 Marks the object so that its runtime initialization occurs before the runtime
1164 initialization of any other objects brought into the process at the same time.
1165 In addition, the object runtime finalization occurs after the runtime
1166 finalization of any other objects removed from the process at the same time.
1167 This option is only meaningful when building a shared object.
1168 .RE

1170 .sp
1171 .ne 2
1172 .na
1173 \fB\fB-z\fR \fBinterpose\fR\fR
1174 .ad
1175 .sp .6
1176 .RS 4n
1177 Marks the object as an interposer. At runtime, an object is identified as an
1178 explicit interposer if the object has been tagged using the \fB-z interpose\fR
1179 option. An explicit interposer is also established when an object is loaded
1180 using the \fBLD_PRELOAD\fR environment variable. Implicit interposition can
1181 occur because of the load order of objects, however, this implicit
1182 interposition is unknown to the runtime linker. Explicit interposition can
```

ensure that interposition takes place regardless of the order in which objects are loaded. Explicit interposition also ensures that the runtime linker searches for symbols in any explicit interposers when direct bindings are in effect.
.RE

.sp
.ne 2
.na
\fB\fB-z\fR \fBlazyload\fR | \fBnolazyload\fR\fR
.ad
.sp .6
.RS 4n
Enables or disables the marking of dynamic dependencies to be lazily loaded. Dynamic dependencies which are marked \fBlazyload\fR are not loaded at initial process start-up. These dependencies are delayed until the first binding to the object is made. \fBNote:\fR Lazy loading requires the correct declaration of dependencies, together with associated runpaths for each dynamic object used within a process. See \fILazy Loading of Dynamic Dependencies\fR in \fILinker and Libraries Guide\fR.
.RE

.sp
.ne 2
.na
\fB\fB-z\fR \fBld32\fR=\fIarg1\fR,\fIarg2\fR,...\fR
.ad
.br
.na
\fB\fB-z\fR \fBld64\fR=\fIarg1\fR,\fIarg2\fR,...\fR
.ad
.sp .6
.RS 4n
The class of the link-editor is affected by the class of the output file being created and by the capabilities of the underlying operating system. The \fB-z\fR \fBld\fR[\fB32\fR|\fB64\fR] options provide a means of defining any link-editor argument. The defined argument is only interpreted, respectively, by the 32-bit class or 64-bit class of the link-editor.
.sp
For example, support libraries are class specific, so the correct class of support library can be ensured using:
.sp
.in +2
.nf
\fBld ... -z ld32=-Saudit32.so.1 -z ld64=-Saudit64.so.1 ...\fR
.fi
.in -2
.sp

The class of link-editor that is invoked is determined from the \fBELF\fR class of the first relocatable file that is seen on the command line. This determination is carried out \fBprior\fR to any \fB-z\fR \fBld\fR[\fB32\fR|\fB64\fR] processing.
.RE

.sp
.ne 2
.na
\fB\fB-z\fR \fBloadfltr\fR\fR
.ad
.sp .6
.RS 4n
Marks a filter to indicate that filtees must be processed immediately at runtime. Normally, filter processing is delayed until a symbol reference is bound to the filter. The runtime processing of an object that contains this flag mimics that which occurs if the \fBLD_LOADFLTR\fR environment variable is

in effect. See the \fBld.so.1\fR(1).
.RE

.sp
.ne 2
.na
\fB\fB-z\fR \fBmuldefs\fR\fR
.ad
.br
.na
\fB\fB--allow-multiple-definition\fR\fR
.ad
.sp .6
.RS 4n
Allows multiple symbol definitions. By default, multiple symbol definitions that occur between relocatable objects result in a fatal error condition. This option, suppresses the error condition, allowing the first symbol definition to be taken.
.RE

.sp
.ne 2
.na
\fB\fB-z\fR \fBnocompstrtab\fR\fR
.ad
.sp .6
.RS 4n
Disables the compression of \fBELF\fR string tables. By default, string compression is applied to \fBSHT_STRTAB\fR sections, and to \fBSHT_PROGBITS\fR sections that have their \fBSHF_MERGE\fR and \fBSHF_STRINGS\fR section flags set.
.RE

.sp
.ne 2
.na
\fB\fB-z\fR \fBnodefaultlib\fR\fR
.ad
.sp .6
.RS 4n
Marks the object so that the runtime default library search path, used after any \fBLD_LIBRARY_PATH\fR or runpaths, is ignored. This option implies that all dependencies of the object can be satisfied from its runpath.
.RE

.sp
.ne 2
.na
\fB\fB-z\fR \fBnodelete\fR\fR
.ad
.sp .6
.RS 4n
Marks the object as non-deletable at runtime. This mode is similar to adding the object to the process by using \fBdlopen\fR(3C) with the \fBRTLD_NODELETE\fR mode.
.RE

.sp
.ne 2
.na
\fB\fB-z\fR \fBnodlopen\fR\fR
.ad
.sp .6
.RS 4n
Marks the object as not available to \fBdlopen\fR(3C), either as the object specified by the \fBdlopen()\fR, or as any form of dependency required by the

object specified by the \fBdlopen()\fR. This option is only meaningful when building a shared object.
.RE

.sp
.ne 2
.na
\fB\fB-z\fR \fBnodump\fR\fR
.ad
.sp .6
.RS 4n
Marks the object as not available to \fBdldump\fR(3C).
.RE

.sp
.ne 2
.na
\fB\fB-z\fR \fBnoldynsym\fR\fR
.ad
.sp .6
.RS 4n
Prevents the inclusion of a \fB\&.SUNW_ldynsym\fR section in dynamic executables or sharable libraries. The \fB\&.SUNW_ldynsym\fR section augments the \fB\&.dynsym\fR section by providing symbols for local functions. Local function symbols allow debuggers to display local function names in stack traces from stripped programs. Similarly, \fBdladdr\fR(3C) is able to supply more accurate results.
.sp
The \fB-z\fR \fBnoldynsym\fR option also prevents the inclusion of the two symbol sort sections that are related to the \fB\&.SUNW_ldynsym\fR section. The \fB\&.SUNW_dynsymsort\fR section provides sorted access to regular function and variable symbols. The \fB\&.SUNW_dyntlssort\fR section provides sorted access to thread local storage (\fBTLS\fR) variable symbols.
.sp
The \fB\&.SUNW_ldynsym\fR, \fB\&.SUNW_dynsymsort\fR, and \fB\&.SUNW_dyntlssort\fR sections, which becomes part of the allocable text segment of the resulting file, cannot be removed by \fBstrip\fR(1). Therefore, the \fB-z\fR \fBnoldynsym\fR option is the only way to prevent their inclusion. See the \fB-s\fR and \fB-z\fR \fBredlocsym\fR options.
.RE

.sp
.ne 2
.na
\fB\fB-z\fR \fBnopartial\fR\fR
.ad
.sp .6
.RS 4n
Partially initialized symbols, that are defined within relocatable object files, are expanded in the output file being generated.
.RE

.sp
.ne 2
.na
\fB\fB-z\fR \fBnoversion\fR\fR
.ad
.sp .6
.RS 4n
Does not record any versioning sections. Any version sections or associated \fB\&.dynamic\fR section entries are not generated in the output image.
.RE

.sp
.ne 2
.na

\fB\fB-z\fR \fBnow\fR\fR
.ad
.sp .6
.RS 4n
Marks the object as requiring non-lazy runtime binding. This mode is similar to adding the object to the process by using \fBdlopen\fR(3C) with the \fBRTLD_NOW\fR mode. This mode is also similar to having the \fBLD_BIND_NOW\fR environment variable in effect. See \fBld.so.1\fR(1).
.RE

.sp
.ne 2
.na
\fB\fB-z\fR \fBorigin\fR\fR
.ad
.sp .6
.RS 4n
Marks the object as requiring immediate \fB$ORIGIN\fR processing at runtime. This option is only maintained for historic compatibility, as the runtime analysis of objects to provide for \fB$ORIGIN\fR processing is now default.
.RE

.sp
.ne 2
.na
\fB\fB-z\fR \fBpreinitarray=\fR\fIfunction\fR\fR
.ad
.sp .6
.RS 4n
Appends an entry to the \fB\&.preinitarray\fR section of the object being built. If no \fB\&.preinitarray\fR section is present, a section is created. The new entry is initialized to point to \fIfunction\fR. See \fIInitialization and Termination Sections\fR in \fILinker and Libraries Guide\fR.
.RE

.sp
.ne 2
.na
\fB\fB-z\fR \fBredlocsym\fR\fR
.ad
.sp .6
.RS 4n
Eliminates all local symbols except for the \fISECT\fR symbols from the symbol table \fBSHT_SYMTAB\fR. All relocations that refer to local symbols are updated to refer to the corresponding \fISECT\fR symbol. This option allows specialized objects to greatly reduce their symbol table sizes. Eliminated local symbols can reduce the \fB\&.stab*\fR debugging information that is generated using the compiler drivers \fB-g\fR option. See the \fB-s\fR and \fB-z\fR \fBnoldynsym\fR options.
.RE

.sp
.ne 2
.na
\fB\fB-z\fR \fBrelaxreloc\fR\fR
.ad
.sp .6
.RS 4n
\fBld\fR normally issues a fatal error upon encountering a relocation using a symbol that references an eliminated COMDAT section. If \fB-z\fR \fBrelaxreloc\fR is enabled, \fBld\fR instead redirects such relocations to the equivalent symbol in the COMDAT section that was kept. \fB-z\fR \fBrelaxreloc\fR is a specialized option, mainly of interest to compiler authors, and is not intended for general use.
.RE

.sp
.ne 2
.na
\fB\fB-z\fR \fBrescan-now\fR\fR
.ad
.br
.na
\fB\fB-z\fR \fBrescan\fR\fR
.ad
.sp .6
.RS 4n
These options rescan the archive files that are provided to the link-edit. By default, archives are processed once as the archives appear on the command line. Archives are traditionally specified at the end of the command line so that their symbol definitions resolve any preceding references. However, specifying archives multiple times to satisfy their own interdependencies can be necessary.
.sp
\fB-z\fR \fBrescan-now\fR is a positional option, and is processed by the link-editor immediately when encountered on the command line. All archives seen on the command line up to that point are immediately reprocessed in an attempt to locate additional archive members that resolve symbol references. This archive rescanning is repeated until a pass over the archives occurs in which no new members are extracted.
.sp
\fB-z\fR \fBrescan\fR is a position independent option. The link-editor defers the rescan operation until after it has processed the entire command line, and then initiates a final rescan operation over all archives seen on the command line. The \fB-z\fR \fBrescan\fR operation can interact incorrectly with objects that contain initialization (.init) or finalization (.fini) sections, preventing the code in those sections from running. For this reason, \fB-z\fR \fBrescan\fR is deprecated, and use of \fB-z\fR \fBrescan-now\fR is advised.
.RE

.sp
.ne 2
.na
\fB\fB-z\fR \fBrescan-start\fR ... \fB-z\fR \fBrescan-end\fR\fR
.ad
.br
.na
\fB\fB--start-group\fR ... \fB--end-group\fR\fR
.ad
.br
.na
\fB\fB-(\fR ... \fB-)\fR\fR
.ad
.sp .6
.RS 4n
Defines an archive rescan group. This is a positional construct, and is processed by the link-editor immediately upon encountering the closing delimiter option. Archives found within the group delimiter options are reprocessed as a group in an attempt to locate additional archive members that resolve symbol references. This archive rescanning is repeated until a pass over the archives On the occurs in which no new members are extracted. Archive rescan groups cannot be nested.
.RE

.sp
.ne 2
.na
\fB\fB-z\fR \fBtarget=sparc|x86\fR \fI\fR\fR
.ad
.sp .6
.RS 4n

Specifies the machine type for the output object. Supported targets are Sparc and x86. The 32-bit machine type for the specified target is used unless the \fB-64\fR option is also present, in which case the corresponding 64-bit machine type is used. By default, the machine type of the object being generated is determined from the first \fBELF\fR object processed from the command line. If no objects are specified, the machine type is determined by the first object encountered within the first archive processed from the command line. If there are no objects or archives, the link-editor assumes the native machine. This option is useful when creating an object directly with \fBld\fR whose input is solely from a \fBmapfile\fR. See the \fB-M\fR option. It can also be useful in the rare case of linking entirely from an archive that contains objects of different machine types for which the first object is not of the desired machine type. See \fIThe 32-bit link-editor and 64-bit link-editor\fR in \fILinker and Libraries Guide\fR.
.RE

.sp
.ne 2
.na
\fB\fB-z\fR \fBtext\fR\fR
.ad
.sp .6
.RS 4n
In dynamic mode only, forces a fatal error if any relocations against non-writable, allocatable sections remain. For historic reasons, this mode is not the default when building an executable or shared object. However, its use is recommended to ensure that the text segment of the dynamic object being built is shareable between multiple running processes. A shared text segment incurs the least relocation overhead when loaded into memory. See \fIPosition-Independent Code\fR in \fILinker and Libraries Guide\fR.
.RE

.sp
.ne 2
.na
\fB\fB-z\fR \fBtextoff\fR\fR
.ad
.sp .6
.RS 4n
In dynamic mode only, allows relocations against all allocatable sections, including non-writable ones. This mode is the default when building a shared object.
.RE

.sp
.ne 2
.na
\fB\fB-z\fR \fBtextwarn\fR\fR
.ad
.sp .6
.RS 4n
In dynamic mode only, lists a warning if any relocations against non-writable, allocatable sections remain. This mode is the default when building an executable.
.RE

.sp
.ne 2
.na
\fB\fB-z\fR \fBverbose\fR\fR
.ad
.sp .6
.RS 4n
This option provides additional warning diagnostics during a link-edit. Presently, this option conveys suspicious use of displacement relocations. This option also conveys the restricted use of static \fBTLS\fR relocations when

```
building shared objects. In future, this option might be enhanced to provide
additional diagnostics that are deemed too noisy to be generated by default.
.RE

.sp
.ne 2
.na
\fB\fB-z\fR\fBwrap=\fR\fIsymbol\fR\fR
.ad
.br
.na
\fB\fB-wrap=\fR \fIsymbol\fR\fR
.ad
.br
.na
\fB\fB--wrap=\fR \fIsymbol\fR\fR
.ad
.sp .6
.RS 4n
Rename undefined references to \fIsymbol\fR in order to allow wrapper code to
be linked into the output object without having to modify source code. When
\fB-z wrap\fR is specified, all undefined references to \fIsymbol\fR are
modified to reference \fB__wrap_\fR\fIsymbol\fR, and all references to
\fB__real_\fR\fIsymbol\fR are modified to reference \fIsymbol\fR. The user is
expected to provide an object containing the \fB__wrap_\fR\fIsymbol\fR
function. This wrapper function can call \fB__real_\fR\fIsymbol\fR in order to
reference the actual function being wrapped.
.sp
The following is an example of a wrapper for the \fBmalloc\fR(3C) function:
.sp
.in +2
.nf
void *
__wrap_malloc(size_t c)
{
        (void) printf("malloc called with %zu\en", c);
        (void) printf("malloc called with %zu\n", c);
        return (__real_malloc(c));
}
______unchanged_portion_omitted_
```

```
**********************************************************
   14878 Tue Sep 10 18:35:19 2013
new/usr/src/man/man1/nm.1
4023 - Typo in file(1) manpage and various others
**********************************************************
   1 '\" te
   2 .\" Copyright 1989 AT&T
   3 .\" Copyright (c) 2007, Sun Microsystems, Inc.  All Rights Reserved
   4 .\" Portions Copyright (c) 1992, X/Open Company Limited  All Rights Reserved
   5 .\" Sun Microsystems, Inc. gratefully acknowledges The Open Group for permission
   6 .\" http://www.opengroup.org/bookstore/.
   7 .\" The Institute of Electrical and Electronics Engineers and The Open Group, ha
   8 .\"  This notice shall appear on any product containing this material.
   9 .\" The contents of this file are subject to the terms of the Common Development
  10 .\" You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE or http:
  11 .\" When distributing Covered Code, include this CDDL HEADER in each file and in
  12 .TH NM 1 "Sep 10, 2013"
  12 .TH NM 1 "Oct 10, 2007"
  13 .SH NAME
  14 nm \- print name list of an object file
  15 .SH SYNOPSIS
  16 .LP
  17 .nf
  18 \fB/usr/bin/nm\fR [\fB-ACDhlnPprRsTuVv\fR] [\fB-efox\fR] [\fB-g\fR | \fB-u\fR]
  19      [\fB-t\fR \fIformat\fR] \fIfile\fR...
  20 .fi

  22 .LP
  23 .nf
  24 \fB/usr/xpg4/bin/nm\fR [\fB-ACDhlnPprRsTuVv\fR] [\fB-efox\fR] [\fB-g\fR | \fB-u\
  25      [\fB-t\fR \fIformat\fR] \fIfile\fR...
  26 .fi

  28 .SH DESCRIPTION
  29 .sp
  30 .LP
  31 The \fBnm\fR utility displays the symbol table of each \fBELF\fR object file
  32 that is specified by \fIfile\fR.
  33 .sp
  34 .LP
  35 If no symbolic information is available for a valid input file, the \fBnm\fR
  36 utility reports that fact, but not consider it an error condition.
  37 .SH OPTIONS
  38 .sp
  39 .LP
  40 The output of \fBnm\fR can be controlled using the following options:
  41 .sp
  42 .ne 2
  43 .na
  44 \fB\fB-A\fR\fR
  45 .ad
  46 .RS 13n
  47 Writes the full path name or library name of an object on each line.
  48 .RE

  50 .sp
  51 .ne 2
  52 .na
  53 \fB\fB-C\fR\fR
  54 .ad
  55 .RS 13n
  56 Demangles C++ symbol names before printing them out.
  57 .RE

  59 .sp
  60 .ne 2
```



```
  61 .na
  62 \fB\fB-D\fR\fR
  63 .ad
  64 .RS 13n
  65 Displays the \fBSHT_DYNSYM\fR symbol information. This is the symbol table used
  66 by \fBld.so.1\fR and is present even in stripped dynamic executables. If
  67 \fB-D\fR is not specified, the default behavior is to display the
  68 \fBSHT_SYMTAB\fR symbol information.
  69 .RE

  71 .sp
  72 .ne 2
  73 .na
  74 \fB\fB-e\fR\fR
  75 .ad
  76 .RS 13n
  77 See NOTES below.
  78 .RE

  80 .sp
  81 .ne 2
  82 .na
  83 \fB\fB-f\fR\fR
  84 .ad
  85 .RS 13n
  86 See NOTES below.
  87 .RE

  89 .sp
  90 .ne 2
  91 .na
  92 \fB\fB-g\fR\fR
  93 .ad
  94 .RS 13n
  95 Writes only external (global) symbol information.
  96 .RE

  98 .sp
  99 .ne 2
 100 .na
 101 \fB\fB-h\fR\fR
 102 .ad
 103 .RS 13n
 104 Does not display the output heading data.
 105 .RE

 107 .sp
 108 .ne 2
 109 .na
 110 \fB\fB-L\fR\fR
 111 .ad
 112 .RS 13n
 113 Displays the \fBSHT_SUNW_LDYNSYM\fR symbol information. This symbol table
 114 contains local function symbols. \fBSHT_SUNW_LDYNSM\fR symbol tables are
 115 present even in stripped dynamic executables. These symbols augment the global
 116 symbols that are found in \fBSHT_DYNSYM\fR symbol table. If \fB-L\fR is not
 117 specified, the default behavior is to display the \fBSHT_SYMTAB\fR symbol
 118 information.
 119 .RE

 121 .sp
 122 .ne 2
 123 .na
 124 \fB\fB-l\fR\fR
 125 .ad
 126 .RS 13n
```

```
127 Distinguishes between \fBWEAK\fR and \fBGLOBAL\fR symbols by appending a * to
128 the key letter for \fBWEAK\fR symbols.
129 .RE

131 .sp
132 .ne 2
133 .na
134 \fB\fB-n\fR\fR
135 .ad
136 .RS 13n
137 Sorts external symbols by name before they are printed.
138 .RE

140 .sp
141 .ne 2
142 .na
143 \fB\fB-o\fR\fR
144 .ad
145 .RS 13n
146 Prints the value and size of a symbol in octal instead of decimal (equivalent
147 to \fB-t\fR \fBo\fR).
148 .RE

150 .sp
151 .ne 2
152 .na
153 \fB\fB-p\fR\fR
154 .ad
155 .RS 13n
156 Produces easy to parse, terse output. Each symbol name is preceded by its value
157 (blanks if undefined) and one of the letters:
158 .sp
159 .ne 2
160 .na
161 \fB\fBA\fR\fR
162 .ad
163 .RS 5n
164 Absolute symbol.
165 .RE

167 .sp
168 .ne 2
169 .na
170 \fB\fBB\fR\fR
171 .ad
172 .RS 5n
173 bss (uninitialized data space) symbol.
174 .RE

176 .sp
177 .ne 2
178 .na
179 \fB\fBC\fR\fR
180 .ad
181 .RS 5n
182 COMMON symbol.
183 .RE

185 .sp
186 .ne 2
187 .na
188 \fB\fBD\fR\fR
189 .ad
190 .RS 5n
191 Data object symbol.
192 .RE
```



```
194 .sp
195 .ne 2
196 .na
197 \fB\fBF\fR\fR
198 .ad
199 .RS 5n
200 File symbol.
201 .RE

203 .sp
204 .ne 2
205 .na
206 \fB\fBN\fR\fR
207 .ad
208 .RS 5n
209 Symbol has no type.
210 .RE

212 .sp
213 .ne 2
214 .na
215 \fB\fBL\fR\fR
216 .ad
217 .RS 5n
218 Thread-Local storage symbol.
219 .RE

221 .sp
222 .ne 2
223 .na
224 \fB\fBS\fR\fR
225 .ad
226 .RS 5n
227 Section symbol.
228 .RE

230 .sp
231 .ne 2
232 .na
233 \fB\fBT\fR\fR
234 .ad
235 .RS 5n
236 Text symbol.
237 .RE

239 .sp
240 .ne 2
241 .na
242 \fB\fBU\fR\fR
243 .ad
244 .RS 5n
245 Undefined.
246 .RE

248 If the symbol’s binding attribute is:
249 .sp
250 .ne 2
251 .na
252 \fB\fBLOCAL\fR\fR
253 .ad
254 .RS 10n
255 The key letter is lower case.
256 .RE

258 .sp
```

```
.ne 2
.na
\fB\fBWEAK\fR\fR
.ad
.RS 10n
The key letter is upper case. If the \fB-l\fR modifier is specified, the upper
case key letter is followed by a \fB*\fR
.RE

.sp
.ne 2
.na
\fB\fBGLOBAL\fR\fR
.ad
.RS 10n
The key letter is upper case.
.RE

.RE

.sp
.ne 2
.na
\fB\fB-P\fR\fR
.ad
.RS 13n
Writes information in a portable output format, as specified in \fBStandard
Output\fR.
.RE

.sp
.ne 2
.na
\fB\fB-r\fR\fR
.ad
.RS 13n
Prepends the name of the object file or archive to each output line.
.RE

.sp
.ne 2
.na
\fB\fB-R\fR\fR
.ad
.RS 13n
Prints the archive name (if present), followed by the object file and symbol
name. If the \fB-r\fR option is also specified, this option is ignored.
.RE

.sp
.ne 2
.na
\fB\fB-s\fR\fR
.ad
.RS 13n
Prints section name instead of section index.
.RE

.sp
.ne 2
.na
\fB\fB-t\fR \fIformat\fR\fR
.ad
.RS 13n
Writes each numeric value in the specified format. The format is dependent on
the single character used as the \fIformat\fR option-argument:
```



```
.sp
.ne 2
.na
\fB\fBd\fR\fR
.ad
.RS 5n
The offset is written in decimal (default).
.RE

.sp
.ne 2
.na
\fB\fBo\fR\fR
.ad
.RS 5n
The offset is written in octal.
.RE

.sp
.ne 2
.na
\fB\fBx\fR\fR
.ad
.RS 5n
The offset is written in hexadecimal.
.RE

.RE

.sp
.ne 2
.na
\fB\fB-T\fR\fR
.ad
.RS 13n
See \fBNOTES\fR.
.RE

.SS "/usr/bin/nm"
.sp
.ne 2
.na
\fB\fB-u\fR\fR
.ad
.RS 6n
Prints undefined symbols only.
.RE

.SS "/usr/xpg4/bin/nm"
.sp
.ne 2
.na
\fB\fB-u\fR\fR
.ad
.RS 6n
Prints long listing for each undefined symbol. See \fBOUTPUT\fR below.
.RE

.sp
.ne 2
.na
\fB\fB-v\fR\fR
.ad
.RS 6n
Sorts external symbols by value before they are printed.
.RE
```

```
392 .sp
393 .ne 2
394 .na
395 \fB\fB-V\fR\fR
396 .ad
397 .RS 6n
398 Prints the version of the \fBnm\fR command executing on the standard error
399 output.
400 .RE

402 .sp
403 .ne 2
404 .na
405 \fB\fB-x\fR\fR
406 .ad
407 .RS 6n
408 Prints the value and size of a symbol in hexadecimal instead of decimal
409 (equivalent to \fB-t\fR \fBx\fR).
410 .RE

412 .sp
413 .LP
414 Options can be used in any order, either singly or in combination, and can
415 appear anywhere in the command line. When conflicting options are specified
416 (such as \fB-v\fR and \fB-n\fR, or \fB-o\fR and \fB-x\fR) the first is taken
417 and the second ignored with a warning message to the user. (See \fB-R\fR for
418 exception.)
419 .SH OPERANDS
420 .sp
421 .LP
422 The following operand is supported:
423 .sp
424 .ne 2
425 .na
426 \fB\fIfile\fR\fR
427 .ad
428 .RS 8n
429 A path name of an object file, executable file or object-file library.
430 .RE

432 .SH OUTPUT
433 .sp
434 .LP
435 This section describes the \fBnm\fR utility’s output options.
436 .SS "Standard Output"
437 .sp
438 .LP
439 For each symbol, the following information is printed:
440 .sp
441 .ne 2
442 .na
443 \fB\fBIndex\fR\fR
444 .ad
445 .RS 15n
446 The index of the symbol. (The index appears in brackets.)
447 .RE

449 .sp
450 .ne 2
451 .na
452 \fB\fBValue\fR\fR
453 .ad
454 .RS 15n
455 The value of the symbol is one of the following:
456 .RS +4
```



```
457 .TP
458 .ie t \(bu
459 .el o
460 A section offset for defined symbols in a relocatable file.
461 .RE
462 .RS +4
463 .TP
464 .ie t \(bu
465 .el o
466 Alignment constraints for symbols whose section index is \fBSHN_COMMON\fR.
467 .RE
468 .RS +4
469 .TP
470 .ie t \(bu
471 .el o
472 A virtual address in executable and dynamic library files.
473 .RE
474 .RE

476 .sp
477 .ne 2
478 .na
479 \fB\fBSize\fR\fR
480 .ad
481 .RS 15n
482 The size in bytes of the associated object.
483 .RE

485 .sp
486 .ne 2
487 .na
488 \fB\fBType\fR\fR
489 .ad
490 .RS 15n
491 A symbol is of one of the following types:
492 .sp
493 .ne 2
494 .na
495 \fB\fBNOTYPE\fR\fR
496 .ad
497 .RS 11n
498 No type was specified.
499 .RE

501 .sp
502 .ne 2
503 .na
504 \fB\fBOBJECT\fR\fR
505 .ad
506 .RS 11n
507 A data object such as an array or variable.
508 .RE

510 .sp
511 .ne 2
512 .na
513 \fB\fBFUNC\fR\fR
514 .ad
515 .RS 11n
516 A function or other executable code.
517 .RE

519 .sp
520 .ne 2
521 .na
522 \fB\fBREGI\fR\fR
```

```
523 .ad
524 .RS 11n
525 A register symbol (\fBSPARC\fR only).
526 .RE

528 .sp
529 .ne 2
530 .na
531 \fB\fBSECTION\fR\fR
532 .ad
533 .RS 11n
534 A section symbol.
535 .RE

537 .sp
538 .ne 2
539 .na
540 \fB\fBFILE\fR\fR
541 .ad
542 .RS 11n
543 Name of the source file.
544 .RE

546 .sp
547 .ne 2
548 .na
549 \fB\fBCOMMON\fR\fR
550 .ad
551 .RS 11n
552 An uninitialized common block.
553 .RE

555 .sp
556 .ne 2
557 .na
558 \fB\fBTLS\fR\fR
559 .ad
560 .RS 11n
561 A variable associated with Thread-Local storage.
562 .RE

564 .RE

566 .sp
567 .ne 2
568 .na
569 \fB\fBBind\fR\fR
570 .ad
571 .RS 15n
572 The symbol's binding attributes.
573 .sp
574 .ne 2
575 .na
576 \fB\fBLOCAL symbols\fR\fR
577 .ad
578 .RS 18n
579 Have a scope limited to the object file containing their definition.
580 .RE

582 .sp
583 .ne 2
584 .na
585 \fB\fBGLOBAL symbols\fR\fR
586 .ad
587 .RS 18n
588 Are visible to all object files being combined.
```



```
589 .RE

591 .sp
592 .ne 2
593 .na
594 \fB\fBWEAK symbols\fR\fR
595 .ad
596 .RS 18n
597 Are essentially global symbols with a lower precedence than \fBGLOBAL\fR.
598 .RE

600 .RE

602 .sp
603 .ne 2
604 .na
605 \fB\fBOther\fR\fR
606 .ad
607 .RS 15n
608 A symbol's visibility.
609 .sp
610 The lower bits of the \fBst_other\fR member of the \fBElf32_Sym\fR structure,
611 and the \fBElf64_Sym\fR structure, defined in \fB<sys/elf.h>\fR, are currently
612 used and can be one of:
613 .sp
614 .in +2
615 .nf
616 #define STV_DEFAULT     0
617 #define STV_INTERNAL    1
618 #define STV_HIDDEN      2
619 #define STV_PROTECTED   3
620 #define STV_EXPORTED    4
621 #define STV_SINGLETON   5
622 #define STV_ELIMINATE   6
623 .fi
624 .in -2
625 .sp

627 .RE

629 .sp
630 .ne 2
631 .na
632 \fB\fBShndx\fR\fR
633 .ad
634 .RS 15n
635 Except for three special values, this is the section header table index in
636 relation to which the symbol is defined. The following special values exist:
637 .sp
638 .ne 2
639 .na
640 \fB\fBABS\fR\fR
641 .ad
642 .RS 10n
643 Indicates the symbol's value does not change through relocation.
644 .RE

646 .sp
647 .ne 2
648 .na
649 \fB\fBCOMMON\fR\fR
650 .ad
651 .RS 10n
652 Indicates an unallocated block and the value provides alignment constraints.
653 .RE
```

```
655 .sp
656 .ne 2
657 .na
658 \fB\fBUNDEF\fR\fR
659 .ad
660 .RS 10n
661 Indicates an undefined symbol.
662 .RE

664 .RE

666 .sp
667 .ne 2
668 .na
669 \fB\fBName\fR\fR
670 .ad
671 .RS 15n
672 The name of the symbol.
673 .RE

675 .sp
676 .ne 2
677 .na
678 \fB\fBObject Name\fR\fR
679 .ad
680 .RS 15n
681 The name of the object or library if \fB-A\fR is specified.
682 .RE

684 .sp
685 .LP
686 If the \fB-P\fR option is specified, the previous information is displayed
687 using the following portable format. The three versions differ depending on
688 whether \fB-t\fR \fBd\fR, \fB-t\fR \fBo\fR, or \fB-t\fR \fBx\fR was specified,
689 respectively:
690 .sp
691 .in +2
692 .nf
693 \fB"%s%s %s %d %d\n",\fR \fIlibrary/object name\fR, \fIname\fR\fB, type,\fR \fIv
694      \fBsize "%s%s %s %o %o\en",\fR \fIlibrary/object name\fR, \fIname\fR, \e
695      \fBtype,\fR \fIvalue\fR \fB, size "%s%s %s %x %x\en",\fR \fIlibrary/object
694      \fBsize "%s%s %s %o %o\n",\fR \fIlibrary/object name\fR, \fIname\fR, \e
695      \fBtype,\fR \fIvalue\fR \fB, size "%s%s %s %x %x\n",\fR \fIlibrary/object n
696      \fBtype,\fR \fIvalue\fR\fB, size\fR
697 .fi
698 .in -2
699 .sp

701 .sp
702 .LP
703 where \fIlibrary/object name\fR is formatted as follows:
704 .RS +4
705 .TP
706 .ie t \(bu
707 .el o
708 If \fB-A\fR is not specified, \fIlibrary/object name\fR is an empty string.
709 .RE
710 .RS +4
711 .TP
712 .ie t \(bu
713 .el o
714 If \fB-A\fR is specified and the corresponding \fIfile\fR operand does not name
715 a library:
716 .sp
717 .in +2
718 .nf
```



```
719 \fB"%s: ", \fIfile\fR\fR
720 .fi
721 .in -2
722 .sp

724 .RE
725 .RS +4
726 .TP
727 .ie t \(bu
728 .el o
729 If \fB-A\fR is specified and the corresponding \fIfile\fR operand names a
730 library. In this case, \fIobject file\fR names the object file in the library
731 containing the symbol being described:
732 .sp
733 .in +2
734 .nf
735 \fB"%s[%s]: ", \fIfile\fR, \fIobject file\fR\fR
736 .fi
737 .in -2
738 .sp

740 .RE
741 .sp
742 .LP
743 If \fB-A\fR is not specified, then if more than one \fIfile\fR operand is
744 specified or if only one \fIfile\fR operand is specified and it names a
745 library, \fBnm\fR writes a line identifying the object containing the following
746 symbols before the lines containing those symbols, in the form:
747 .RS +4
748 .TP
749 .ie t \(bu
750 .el o
751 If the corresponding \fIfile\fR operand does not name a library:
752 .sp
753 .in +2
754 .nf
755 \fB"%s:\en", \fIfile\fR\fR
755 \fB"%s:\n", \fIfile\fR\fR
756 .fi
757 .in -2
758 .sp

760 .RE
761 .RS +4
762 .TP
763 .ie t \(bu
764 .el o
765 If the corresponding \fIfile\fR operand names a library; in this case,
766 \fIobject file\fR is the name of the file in the library containing the
767 following symbols:
768 .sp
769 .in +2
770 .nf
771 \fB"%s[%s]:\en", \fIfile\fR, \fIobject file\fR\fR
771 \fB"%s[%s]:\n", \fIfile\fR, \fIobject file\fR\fR
772 .fi
773 .in -2
774 .sp

776 .RE
777 .sp
778 .LP
779 If \fB-P\fR is specified, but \fB-t\fR is not, the format is as if \fB-t\fR
780 \fBx\fR had been specified.
781 .SH ENVIRONMENT VARIABLES
782 .sp
```

```
783 .LP
784 See \fBenviron\fR(5) for descriptions of the following environment variables
785 that affect the execution of \fBnm\fR: \fBLANG\fR, \fBLC_ALL\fR,
786 \fBLC_COLLATE\fR, \fBLC_CTYPE\fR, \fBLC_MESSAGES\fR, and \fBNLSPATH\fR.
787 .SH EXIT STATUS
788 .sp
789 .LP
790 The following exit values are returned:
791 .sp
792 .ne 2
793 .na
794 \fB\fB0\fR\fR
795 .ad
796 .RS 6n
797 Successful completion.
798 .RE

800 .sp
801 .ne 2
802 .na
803 \fB\fB>0\fR\fR
804 .ad
805 .RS 6n
806 An error occurred.
807 .RE

809 .SH ATTRIBUTES
810 .sp
811 .LP
812 See \fBattributes\fR(5) for descriptions of the following attributes:
813 .SH /USR/XPG4/BIN/NM
814 .sp

816 .sp
817 .TS
818 box;
819 c | c
820 l | l .
821 ATTRIBUTE TYPE  ATTRIBUTE VALUE
822 _
823 Interface Stability     Committed
824 .TE

826 .SH SEE ALSO
827 .sp
828 .LP
829 \fBar\fR(1), \fBas\fR(1), \fBdump\fR(1), \fBld\fR(1), \fBld.so.1\fR(1),
830 \fBar.h\fR(3HEAD), \fBa.out\fR(4), \fBattributes\fR(5), \fBenviron\fR(5),
831 \fBstandards\fR(5)
832 .SH NOTES
833 .sp
834 .LP
835 The following options are obsolete because of changes to the object file format
836 and might be deleted in a future release.
837 .sp
838 .ne 2
839 .na
840 \fB\fB-e\fR\fR
841 .ad
842 .RS 6n
843 Prints only external and static symbols. The symbol table now contains only
844 static and external symbols. Automatic symbols no longer appear in the symbol
845 table. They do appear in the debugging information produced by \fBcc\fR
846 \fB-g\fR, which can be examined using \fBdump\fR(1).
847 .RE
```



```
849 .sp
850 .ne 2
851 .na
852 \fB\fB-f\fR\fR
853 .ad
854 .RS 6n
855 Produces full output. Redundant symbols (such as \fB\&.text\fR, \fB\&.data\fR,
856 and so forth), which existed previously, do not exist and producing full output
857 is identical to the default output.
858 .RE

860 .sp
861 .ne 2
862 .na
863 \fB\fB-T\fR\fR
864 .ad
865 .RS 6n
866 By default, \fBnm\fR prints the entire name of the symbols listed. Since symbol
867 names have been moved to the last column, the problem of overflow is removed
868 and it is no longer necessary to truncate the symbol name.
869 .RE
```

```
**********************************************************
   21233 Tue Sep 10 18:35:19 2013
new/usr/src/man/man1/printf.1
4023 - Typo in file(1) manpage and various others
**********************************************************
'\" te
.\" Copyright (c) 2009, Sun Microsystems, Inc. All Rights Reserved
.\" Copyright 1992, X/Open Company Limited All Rights Reserved
.\" Portions Copyright (c) 1982-2007 AT&T Knowledge Ventures
.\" Sun Microsystems, Inc. gratefully acknowledges The Open Group for permission
.\" The Institute of Electrical and Electronics Engineers and The Open Group, ha
.\" are reprinted and reproduced in electronic form in the Sun OS Reference Manu
.\" and Electronics Engineers, Inc and The Open Group. In the event of any discr
.\"  This notice shall appear on any product containing this material.
.\" 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 PRINTF 1 "Aug 11, 2009"
.SH NAME
printf \- write formatted output
.SH SYNOPSIS
.SS "/usr/bin/printf"
.LP
.nf
\fBprintf\fR \fIformat\fR [\fIargument\fR]...
.fi

.SS "ksh93"
.LP
.nf
\fBprintf\fR \fIformat\fR [\fIstring\fR...]
.fi

.SH DESCRIPTION
.SS "/usr/bin/printf"
.sp
.LP
The \fBprintf\fR utility writes each string operand to standard output using
\fIformat\fR to control the output format.
.SH OPERANDS
.SS "/usr/bin/printf"
.sp
.LP
The following operands are supported by \fB/usr/bin/printf\fR:
.sp
.ne 2
.na
\fB\fIformat\fR\fR
.ad
.RS 12n
A string describing the format to use to write the remaining operands. The
\fIformat\fR operand is used as the \fIformat\fR string described on the
\fBformats\fR(5) manual page, with the following exceptions:
.RS +4
.TP
.ie t \(bu
.el o
A \fBSPACE\fR character in the format string, in any context other than a flag
of a conversion specification, is treated as an ordinary character that is
copied to the output.
.RE
.RS +4
.TP
.ie t \(bu
.el o
A character in the format string is treated as a character, not as a
```



```
\fBSPACE\fR character.
.RE
.RS +4
.TP
.ie t \(bu
.el o
In addition to the escape sequences described on the \fBformats\fR(5) manual
page (\fB\e\e\fR, \fB\ea\fR, \fB\eb\fR, \fB\ef\fR, \fB\en\fR, \fB\er\fR,
\fB\et\fR, \fB\ev\fR), \fB\e\fR\fIddd\fR, where \fIddd\fR is a one-, two- or
three-digit octal number, is written as a byte with the numeric value specified
by the octal number.
.RE
.RS +4
.TP
.ie t \(bu
.el o
The program does not precede or follow output from the \fBd\fR or \fBu\fR
conversion specifications with blank characters not specified by the
\fIformat\fR operand.
.RE
.RS +4
.TP
.ie t \(bu
.el o
The program does not precede output from the \fBo\fR conversion specification
with zeros not specified by the \fIformat\fR operand.
.RE
.RS +4
.TP
.ie t \(bu
.el o
An additional conversion character, \fBb\fR, is supported as follows. The
argument is taken to be a string that can contain backslash-escape sequences.
The following backslash-escape sequences are supported:
.RS +4
.TP
.ie t \(bu
.el o
the escape sequences listed on the \fBformats\fR(5) manual page (\fB\e\e\fR,
\fB\ea\fR, \fB\eb\fR, \fB\ef\fR, \fB\en\fR, \fB\er\fR, \fB\et\fR, \fB\ev\fR),
which are converted to the characters they represent
.RE
.RS +4
.TP
.ie t \(bu
.el o
\fB\e0\fR\fIddd\fR, where \fIddd\fR is a zero-, one-, two- or three-digit octal
number that is converted to a byte with the numeric value specified by the
octal number
.RE
.RS +4
.TP
.ie t \(bu
.el o
\fB\ec\fR, which is written and causes \fBprintf\fR to ignore any remaining
characters in the string operand containing it, any remaining string operands
and any additional characters in the \fIformat\fR operand.
.RE
.RE
The interpretation of a backslash followed by any other sequence of characters
is unspecified.
.sp
Bytes from the converted string are written until the end of the string or the
number of bytes indicated by the precision specification is reached. If the
precision is omitted, it is taken to be infinite, so all bytes up to the end of
the converted string are written. For each specification that consumes an
```

```
128 argument, the next argument operand is evaluated and converted to the
129 appropriate type for the conversion as specified below. The \fIformat\fR
130 operand is reused as often as necessary to satisfy the argument operands. Any
131 extra \fBc\fR or \fBs\fR conversion specifications are evaluated as if a null
132 string argument were supplied; other extra conversion specifications are
133 evaluated as if a zero argument were supplied. If the \fIformat\fR operand
134 contains no conversion specifications and \fIargument\fR operands are present,
135 the results are unspecified. If a character sequence in the \fIformat\fR
136 operand begins with a \fB%\fR character, but does not form a valid conversion
137 specification, the behavior is unspecified.
138 .RE

140 .sp
141 .ne 2
142 .na
143 \fB\fIargument\fR\fR
144 .ad
145 .RS 12n
146 The strings to be written to standard output, under the control of
147 \fBformat\fR. The \fIargument\fR operands are treated as strings if the
148 corresponding conversion character is \fBb\fR, \fBc\fR or \fBs\fR. Otherwise,
149 it is evaluated as a C constant, as described by the ISO C standard, with the
150 following extensions:
151 .RS +4
152 .TP
153 .ie t \(bu
154 .el o
155 A leading plus or minus sign is allowed.
156 .RE
157 .RS +4
158 .TP
159 .ie t \(bu
160 .el o
161 If the leading character is a single- or double-quote, the value is the numeric
162 value in the underlying codeset of the character following the single- or
163 double-quote.
164 .RE
165 If an argument operand cannot be completely converted into an internal value
166 appropriate to the corresponding conversion specification, a diagnostic message
167 is written to standard error and the utility does not exit with a zero exit
168 status, but continues processing any remaining operands and writes the value
169 accumulated at the time the error was detected to standard output.
170 .RE

172 .SS "ksh93"
173 .sp
174 .LP
175 The \fIformat\fR operands support the full range of ANSI C/C99/XPG6 formatting
176 specifiers as well as additional specifiers:
177 .sp
178 .ne 2
179 .na
180 \fB\fB%b\fR\fR
181 .ad
182 .RS 6n
183 Each character in the string operand is processed specially, as follows:
184 .sp
185 .ne 2
186 .na
187 \fB\fB\ea\fR\fR
188 .ad
189 .RS 8n
190 Alert character.
191 .RE

193 .sp
```



```
194 .ne 2
195 .na
196 \fB\fB\eb\fR\fR
197 .ad
198 .RS 8n
199 Backspace character.
200 .RE

202 .sp
203 .ne 2
204 .na
205 \fB\fB\ec\fR\fR
206 .ad
207 .RS 8n
208 Terminate output without appending NEWLINE. The remaining string operands are
209 ignored.
210 .RE

212 .sp
213 .ne 2
214 .na
215 \fB\fB\eE\fR\fR
216 .ad
217 .RS 8n
218 Escape character (\fBASCII\fR octal \fB033\fR).
219 .RE

221 .sp
222 .ne 2
223 .na
224 \fB\fB\ef\fR\fR
225 .ad
226 .RS 8n
227 FORM FEED character.
228 .RE

230 .sp
231 .ne 2
232 .na
233 \fB\fB\en\fR\fR
234 .ad
235 .RS 8n
236 NEWLINE character.
237 .RE

239 .sp
240 .ne 2
241 .na
242 \fB\fB\et\fR\fR
243 .ad
244 .RS 8n
245 TAB character.
246 .RE

248 .sp
249 .ne 2
250 .na
251 \fB\fB\ev\fR\fR
252 .ad
253 .RS 8n
254 Vertical tab character.
255 .RE

257 .sp
258 .ne 2
259 .na
```

```
260 \fB\fB\e\e\fR\fR
261 .ad
262 .RS 8n
263 Backslash character.
264 .RE

266 .sp
267 .ne 2
268 .na
269 \fB\fB\e0\fR\fIx\fR\fR
270 .ad
271 .RS 8n
272 The 8-bit character whose \fBASCII\fR code is the \fB1\fR-, \fB2\fR-, or
273 \fB3\fR-digit octal number \fIx\fR.
274 .RE

276 .RE

278 .sp
279 .ne 2
280 .na
281 \fB\fB%B\fR\fR
282 .ad
283 .RS 6n
284 Treat the argument as a variable name and output the value without converting
285 it to a string. This is most useful for variables of type \fB-b\fR.
286 .RE

288 .sp
289 .ne 2
290 .na
291 \fB\fB%H\fR\fR
292 .ad
293 .RS 6n
294 Output string with characters \fB<\fR, \fB&\fR, \fB>\fR, \fB"\fR, and
295 non-printable characters, properly escaped for use in HTML and XML documents.
296 .RE

298 .sp
299 .ne 2
300 .na
301 \fB\fB%P\fR\fR
302 .ad
303 .RS 6n
304 Treat \fIstring\fR as an extended regular expression and convert it to a shell
305 pattern.
306 .RE

308 .sp
309 .ne 2
310 .na
311 \fB\fB%q\fR\fR
312 .ad
313 .RS 6n
314 Output \fIstring\fR quoted in a manner that it can be read in by the shell to
315 get back the same string. However, empty strings resulting from missing string
316 operands are not quoted.
317 .RE

319 .sp
320 .ne 2
321 .na
322 \fB\fB%R\fR\fR
323 .ad
324 .RS 6n
325 Treat \fIstring\fR as an shell pattern expression and convert it to an extended
```



```
326 regular expression.
327 .RE

329 .sp
330 .ne 2
331 .na
332 \fB\fB%T\fR\fR
333 .ad
334 .RS 6n
335 Treat \fIstring\fR as a date/time string and format it. The \fBT\fR can be
336 preceded by (\fIdformat\fR), where \fIdformat\fR is a date format as defined by
337 the \fBdate\fR(1) command.
338 .RE

340 .sp
341 .ne 2
342 .na
343 \fB\fB%Z\fR\fR
344 .ad
345 .RS 6n
346 Output a byte whose value is \fB0\fR.
347 .RE

349 .sp
350 .LP
351 When performing conversions of \fIstring\fR to satisfy a numeric format
352 specifier, if the first character of \fIstring\fR is \fB"or'\fR, the value is
353 the numeric value in the underlying code set of the character following the
354 \fB"or'\fR. Otherwise, \fIstring\fR is treated like a shell arithmetic
355 expression and evaluated.
356 .sp
357 .LP
358 If a \fIstring\fR operand cannot be completely converted into a value
359 appropriate for that format specifier, an error occurs, but remaining
360 \fIstring\fR operands continue to be processed.
361 .sp
362 .LP
363 In addition to the format specifier extensions, the following extensions of
364 ANSI C/C99/XPG6 are permitted in format specifiers:
365 .RS +4
366 .TP
367 .ie t \(bu
368 .el o
369 The escape sequences \fB\eE\fR and \fB\ee\fR expand to the escape character
370 which is octal 033 in ASCII.
371 .RE
372 .RS +4
373 .TP
374 .ie t \(bu
375 .el o
376 The escape sequence \fB\ecx\fR expands to CTRL-x.
377 .RE
378 .RS +4
379 .TP
380 .ie t \(bu
381 .el o
382 The escape sequence \fB\eC[.\fR\fIname\fR\fB\&.]\fR expands to the collating
383 element \fIname\fR.
384 .RE
385 .RS +4
386 .TP
387 .ie t \(bu
388 .el o
389 The escape sequence \fB\ex{hex}\fRexpands to the character corresponding to the
390 hexadecimal value \fBhex\fR.
391 .RE
```

```
.RS +4
.TP
.ie t \(bu
.el o
The format modifier flag = can be used to center a field to a specified width.
When the output is a terminal, the character width is used rather than the
number of bytes.
.RE
.RS +4
.TP
.ie t \(bu
.el o
Each of the integral format specifiers can have a third modifier after width
and precision that specifies the base of the conversion from 2 to 64. In this
case, the \fB#\fR modifier causes \fIbase\fR\fB#\fR to be prepended to the
value.
.RE
.RS +4
.TP
.ie t \(bu
.el o
The \fB#\fR modifier can be used with the \fBd\fR specifier when no base is
specified to cause the output to be written in units of 1000 with a suffix of
one of \fBk M G T P E\fR.
.RE
.RS +4
.TP
.ie t \(bu
.el o
The \fB#\fR modifier can be used with the \fBi\fR specifier to cause the output
to be written in units of \fB1024\fR with a suffix of one of \fBKi Mi Gi Ti Pi
Ei\fR.
.RE
.sp
.LP
If there are more \fIstring\fR operands than format specifiers, the format
string is reprocessed from the beginning. If there are fewer \fIstring\fR
operands than format specifiers, then \fIstring\fR specifiers are treated as if
empty strings were supplied, numeric conversions are treated as if \fB0\fR was
supplied, and time conversions are treated as if \fBnow\fR was supplied.
.sp
.LP
\fB/usr/bin/printf\fR is equivalent to \fBksh93\fR's \fBprintf\fR built-in and
\fBprint -f\fR, which allows additional options to be specified.
.SH USAGE
.SS "/usr/bin/printf"
.sp
.LP
The \fBprintf\fR utility, like the \fBprintf\fR(3C) function on which it is
based, makes no special provision for dealing with multi-byte characters when
using the \fB%c\fR conversion specification. Applications should be extremely
cautious using either of these features when there are multi-byte characters in
the character set.
.sp
.LP
Field widths and precisions cannot be specified as \fB*\fR.
.sp
.LP
The \fB%b\fR conversion specification is not part of the ISO C standard; it has
been added here as a portable way to process backslash escapes expanded in
string operands as provided by the \fBecho\fR utility. See also the USAGE
section of the \fBecho\fR(1) manual page for ways to use \fBprintf\fR as a
replacement for all of the traditional versions of the \fBecho\fR utility.
.sp
.LP
If an argument cannot be parsed correctly for the corresponding conversion
```



```
specification, the \fBprintf\fR utility reports an error. Thus, overflow and
extraneous characters at the end of an argument being used for a numeric
conversion are to be reported as errors.
.sp
.LP
It is not considered an error if an argument operand is not completely used for
a \fBc\fR or \fBs\fR conversion or if a string operand's first or second
character is used to get the numeric value of a character.
.SH EXAMPLES
.SS "/usr/bin/printf"
.LP
\fBExample 1 \fRPrinting a Series of Prompts
.sp
.LP
The following example alerts the user, then prints and reads a series of
prompts:

.sp
.in +2
.nf
example% \fBprintf "\eaPlease fill in the following: \enName: "
read name
printf "Phone number: "
read phone\fR
.fi
.in -2
.sp

.LP
\fBExample 2 \fRPrinting a Table of Calculations
.sp
.LP
The following example prints a table of calculations. It reads out a list of
right and wrong answers from a file, calculates the percentage correctly, and
prints them out. The numbers are right-justified and separated by a single tab
character. The percentage is written to one decimal place of accuracy:

.sp
.in +2
.nf
example% \fBwhile read right wrong ; do
   percent=$(echo "scale=1;($right*100)/($right+$wrong)" | bc)
   printf "%2d right\et%2d wrong\et(%s%%)\en" \e
           $right $wrong $percent
done < database_file\fR
.fi
.in -2
.sp

.LP
\fBExample 3 \fRPrinting number strings
.sp
.LP
The command:

.sp
.in +2
.nf
example% \fBprintf "%5d%4d\en" 1 21 321 4321 54321\fR
.fi
.in -2
.sp

.sp
.LP
produces:
```

```
.sp
.in +2
.nf
    1  21
  3214321
54321   0
.fi
.in -2
.sp

.sp
.LP
The \fIformat\fR operand is used three times to print all of the given strings
and that a \fB0\fR was supplied by \fBprintf\fR to satisfy the last \fB%4d\fR
conversion specification.

.LP
\fBExample 4 \fRTabulating Conversion Errors
.sp
.LP
The following example tabulates conversion errors.

.sp
.LP
The \fBprintf\fR utility tells the user when conversion errors are detected
while producing numeric output. These results would be expected on an
implementation with 32-bit twos-complement integers when \fB%d\fR is specified
as the \fIformat\fR operand:

.sp

.sp
.TS
box;
c c c
l l l .
Arguments       Standard        Diagnostic
5a      5       printf: 5a not completely converted
9999999999      2147483647      printf: 9999999999: Results too large
-9999999999     -2147483648     printf: -9999999999: Results too large
ABC     0       printf: ABC expected numeric value
.TE

.sp
.LP
The value shown on standard output is what would be expected as the return
value from the function \fBstrtol\fR(3C). A similar correspondence exists
between \fB%u\fR and \fBstrtoul\fR(3C), and \fB%e\fR, \fB%f\fR and \fB%g\fR and
\fBstrtod\fR(3C).

.LP
\fBExample 5 \fRPrinting Output for a Specific Locale
.sp
.LP
The following example prints output for a specific locale. In a locale using
the ISO/IEC 646:1991 standard as the underlying codeset, the command:

.sp
.in +2
.nf
example% \fBprintf "%d\en" 3 +3 -3 \e'3 \e"+3 "'-3"\fR
.fi
.in -2
.sp
```



```
.sp
.LP
produces:

.sp

.sp
.TS
box;
l l
l l .
\fB3\fR Numeric value of constant 3
\fB3\fR Numeric value of constant 3
\fB\(mi3\fR     Numeric value of constant \(mi3
\fB51\fR        T{
Numeric value of the character '3' in the ISO/IEC 646:1991 standard codeset
T}
\fB43\fR        T{
Numeric value of the character '+' in the ISO/IEC 646:1991 standard codeset
T}
\fB45\fR        T{
Numeric value of the character '\(mi' in the SO/IEC 646:1991 standard codeset
T}
.TE

.sp
.LP
In a locale with multi-byte characters, the value of a character is intended to
be the value of the equivalent of the \fBwchar_t\fR representation of the
character.

.sp
.LP
If an argument operand cannot be completely converted into an internal value
appropriate to the corresponding conversion specification, a diagnostic message
is written to standard error and the utility does exit with a zero exit status,
but continues processing any remaining operands and writes the value
accumulated at the time the error was detected to standard output.

.LP
\fBExample 6 \fRAlternative floating point representation 1
.sp
.LP
The \fBprintf\fR utility supports an alternative floating point representation
(see \fBprintf\fR(3C) entry for the "\fB%a\fR"/"\fB%A\fR"), which allows the
output of floating-point values in a format that avoids the usual base16 to
base10 rounding errors.

.sp
.in +2
.nf
example% printf "%a\en" 2 3.1 NaN
example% printf "%a\n" 2 3.1 NaN
.fi
.in -2
.sp

.sp
.LP
produces:

.sp
.in +2
.nf
0x1.0000000000000000000000000000p+01
0x1.8ccccccccccccccccccccccccccdp+01
```

```
655 nan
656 .fi
657 .in -2
658 .sp

660 .LP
661 \fBExample 7 \fRAlternative floating point representation 2
662 .sp
663 .LP
664 The following example shows two different representations of the same
665 floating-point value.

667 .sp
668 .in +2
669 .nf
670 example% x=2 ; printf "%f == %a\en" x x
670 example% x=2 ; printf "%f == %a\n" x x
671 .fi
672 .in -2
673 .sp

675 .sp
676 .LP
677 produces:

679 .sp
680 .in +2
681 .nf
682 2.000000 == 0x1.0000000000000000000000000000p+01
683 .fi
684 .in -2
685 .sp

687 .LP
688 \fBExample 8 \fROutput of unicode values
689 .sp
690 .LP
691 The following command will print the EURO unicode symbol (code-point 0x20ac).

693 .sp
694 .in +2
695 .nf
696 example% LC_ALL=en_US.UTF-8 printf "\u[20ac]\en"
696 example% LC_ALL=en_US.UTF-8 printf "\u[20ac]\n"
697 .fi
698 .in -2
699 .sp

701 .sp
702 .LP
703 produces:

705 .sp
706 .in +2
707 .nf
708 <euro>
709 .fi
710 .in -2
711 .sp

713 .sp
714 .LP
715 where "<euro>" represents the EURO currency symbol character.

717 .LP
718 \fBExample 9 \fRConvert unicode character to unicode code-point value
```



```
719 .sp
720 .LP
721 The following command will print the hexadecimal value of a given character.

723 .sp
724 .in +2
725 .nf
726 example% export LC_ALL=en_US.UTF-8
727 example% printf "%x\en" "'<euro>"
727 example% printf "%x\n" "'<euro>"
728 .fi
729 .in -2
730 .sp

732 .sp
733 .LP
734 where "<euro>" represents the EURO currency symbol character (code-point
735 0x20ac).

737 .sp
738 .LP
739 produces:

741 .sp
742 .in +2
743 .nf
744 20ac
745 .fi
746 .in -2
747 .sp

749 .LP
750 \fBExample 10 \fRPrint the numeric value of an ASCII character
751 .sp
752 .in +2
753 .nf
754 example% printf "%d\en" "'A"
754 example% printf "%d\n" "'A"
755 .fi
756 .in -2
757 .sp

759 .sp
760 .LP
761 produces:

763 .sp
764 .in +2
765 .nf
766 65
767 .fi
768 .in -2
769 .sp

771 .LP
772 \fBExample 11 \fRPrint the language-independent date and time format
773 .sp
774 .LP
775 To print the language-independent date and time format, the following statement
776 could be used:

778 .sp
779 .in +2
780 .nf
781 example% printf "format" weekday month day hour min
782 .fi
```

```
783 .in -2
784 .sp

786 .sp
787 .LP
788 For example,

790 .sp
791 .in +2
792 .nf
793 $ printf format "Sunday" "July" 3 10 2
794 .fi
795 .in -2
796 .sp

798 .sp
799 .LP
800 For American usage, format could be the string:

802 .sp
803 .in +2
804 .nf
805 "%s, %s %d, %d:%.2d\en"
805 "%s, %s %d, %d:%.2d\n"
806 .fi
807 .in -2
808 .sp

810 .sp
811 .LP
812 producing the message:

814 .sp
815 .in +2
816 .nf
817 Sunday, July 3, 10:02
818 .fi
819 .in -2
820 .sp

822 .sp
823 .LP
824 Whereas for EU usage, format could be the string:

826 .sp
827 .in +2
828 .nf
829 "%1$s, %3$d. %2$s, %4$d:%5$.2d\en"
829 "%1$s, %3$d. %2$s, %4$d:%5$.2d\n"
830 .fi
831 .in -2
832 .sp

834 .sp
835 .LP
836 Note that the '$' characters must be properly escaped, such as

838 .sp
839 .in +2
840 .nf
841 "%1\$s, %3\$d. %2\$s, %4\$d:%5\$.2d\en" in this case
841 "%1\$s, %3\$d. %2\$s, %4\$d:%5\$.2d\n" in this case
842 .fi
843 .in -2
844 .sp
```



```
846 .sp
847 .LP
848 producing the message:

850 .sp
851 .in +2
852 .nf
853 Sunday, 3. July, 10:02
854 .fi
855 .in -2
856 .sp

858 .SH ENVIRONMENT VARIABLES
859 .sp
860 .LP
861 See \fBenviron\fR(5) for descriptions of the following environment variables
862 that affect the execution of \fBprintf\fR: \fBLANG\fR, \fBLC_ALL\fR,
863 \fBLC_CTYPE\fR, \fBLC_MESSAGES\fR, \fBLC_NUMERIC\fR, and \fBNLSPATH\fR.
864 .SH EXIT STATUS
865 .sp
866 .LP
867 The following exit values are returned:
868 .sp
869 .ne 2
870 .na
871 \fB\fB0\fR\fR
872 .ad
873 .RS 6n
874 Successful completion.
875 .RE

877 .sp
878 .ne 2
879 .na
880 \fB\fB>0\fR\fR
881 .ad
882 .RS 6n
883 An error occurred.
884 .RE

886 .SH ATTRIBUTES
887 .sp
888 .LP
889 See \fBattributes\fR(5) for descriptions of the following attributes:
890 .SS "/usr/bin/printf"
891 .sp

893 .sp
894 .TS
895 box;
896 c | c
897 l | l .
898 ATTRIBUTE TYPE  ATTRIBUTE VALUE
899 _
900 CSI     Enabled
901 _
902 Interface Stability     Committed
903 _
904 Standard        See \fBstandards\fR(5).
905 .TE

907 .SS "ksh93"
908 .sp

910 .sp
911 .TS
```

```
box;
c | c
l | l .
ATTRIBUTE TYPE  ATTRIBUTE VALUE
_
Interface Stability     Uncommitted
.TE

.SH SEE ALSO
.sp
.LP
\fBawk\fR(1), \fBbc\fR(1), \fBdate\fR(1), \fBecho\fR(1), \fBksh93\fR(1),
\fBprintf\fR(3C), \fBstrtod\fR(3C), \fBstrtol\fR(3C), \fBstrtoul\fR(3C),
\fBattributes\fR(5), \fBenviron\fR(5), \fBformats\fR(5), \fBstandards\fR(5)
.SH NOTES
.sp
.LP
Using format specifiers (characters following '%') which are not listed in the
\fBprintf\fR(3C) or this manual page will result in undefined behavior.
.sp
.LP
Using escape sequences (the character following a backslash ('\e')) which are
Using escape sequences (the character following a backslash ('\')) which are
not listed in the \fBprintf\fR(3C) or this manual page will result in undefined
behavior.
.sp
.LP
Floating-point values follow C99, XPG6 and IEEE 754 standard behavior and can
handle values the same way as the platform's |\fBlong double\fR| datatype.
.sp
.LP
Floating-point values handle the sign separately which allows signs for values
like NaN (for example, -nan), Infinite (for example, -inf) and zero (for
example, -0.0).
```

```
**********************************************************
    5851 Tue Sep 10 18:35:19 2013
new/usr/src/man/man1/sleep.1
4023 - Typo in file(1) manpage and various others
**********************************************************
'\" te
.\" Copyright (c) 2007, Sun Microsystems, Inc.  All Rights Reserved  Portions
.\" Copyright 1989 AT&T
.\" Portions Copyright (c) 1982-2007 AT&T Knowledge Ventures
.\" Portions Copyright (c) 1992, X/Open Company Limited  All Rights Reserved
.\" Sun Microsystems, Inc. gratefully acknowledges The Open Group for permission
.\" The Institute of Electrical and Electronics Engineers and The Open Group, ha
.\" are reprinted and reproduced in electronic form in the Sun OS Reference Manu
.\" and Electronics Engineers, Inc and The Open Group. In the event of any discr
.\"  This notice shall appear on any product containing this material.
.\" 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 SLEEP 1 "Sep 10, 2013"
.TH SLEEP 1 "Nov 20, 2007"
.SH NAME
sleep \- suspend execution for an interval
.SH SYNOPSIS
.SS "/usr/bin/sleep"
.LP
.nf
\fB/usr/bin/sleep\fR \fIseconds\fR
.fi

.SS "ksh93"
.LP
.nf
\fBsleep\fR \fIseconds\fR
.fi

.SH DESCRIPTION
.sp
.LP
\fBsleep\fR suspends execution for at least the time in seconds specified by
\fIseconds\fR or until a \fBSIGALRM\fR signal is received. The \fIseconds\fR
operand can be specified as a floating point number but the actual granularity
normally depends on the underlying system.
.SH OPERANDS
.SS "/usr/bin/sleep"
.sp
.LP
The following operands are supported for \fB/usr/bin/sleep\fR and \fBksh93\fR's
\fBsleep\fR built-in command:
.sp
.ne 2
.na
\fB\fIseconds\fR\fR
.ad
.RS 11n
A non-negative floating-point number specifying the number of seconds for which
to suspend execution. The floating-point number may be specified in all formats
required by C99/XPG6, including constants such as "Inf" or "infinite".
.RE

.SS "ksh93"
.sp
.LP
The following operands are supported:
.sp
.ne 2
.na
```



```
\fB\fItime\fR\fR
.ad
.RS 8n
Specify \fItime\fR in seconds as a floating point number. The actual
granularity depends on the underlying system, normally around 1 millisecond.
.RE

.SH EXAMPLES
.LP
\fBExample 1 \fRSuspending Command Execution
.sp
.LP
The following example executes a command after a certain amount of time:

.sp
.in +2
.nf
example% \fB(sleep 105; \fIcommand\fR)&\fR
.fi
.in -2
.sp

.LP
\fBExample 2 \fRExecuting a Command Every So Often
.sp
.LP
The following example executes a command every so often:

.sp
.in +2
.nf
example% \fBwhile true
do
        \fIcommand\fR
        sleep 37
done\fR
.fi
.in -2
.sp

.LP
\fBExample 3 \fRSuspend command execution forever (or until a \fBSIGALRM\fR
signal is received)
.sp
.in +2
.nf
example% sleep Inf
.fi
.in -2
.sp

.LP
\fBExample 4 \fRSuspending command execution for 0.5 seconds
.sp
.LP
Suspending command execution for 0.5 seconds using an alternative
floating-point representation for the value "0.5"

.sp
.in +2
.nf
example% printf "%a\en" 0.5
example% printf "%a\n" 0.5
0x1.00000000000000000000000000000p-01
.fi
.in -2
```

```
.sp

.sp
.in +2
.nf
example% sleep 0x1.0000000000000000000000000000p-01
.fi
.in -2
.sp

.SH ENVIRONMENT VARIABLES
.sp
.LP
See \fBenviron\fR(5) for descriptions of the following environment variables
that affect the execution of \fBsleep\fR: \fBLANG\fR, \fBLC_ALL\fR,
\fBLC_CTYPE\fR, \fBLC_MESSAGES\fR, and \fBNLSPATH\fR.
.SH EXIT STATUS
.sp
.LP
The following exit values are returned:
.sp
.ne 2
.na
\fB\fB0\fR\fR
.ad
.RS 6n
The execution was successfully suspended for at least \fItime\fR seconds, or a
\fBSIGALRM\fR signal was received (see NOTES).
.RE

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

.SH ATTRIBUTES
.sp
.LP
See \fBattributes\fR(5) for descriptions of the following attributes:
.SS "/usr/bin/sleep"
.sp

.sp
.TS
box;
c | c
l | l .
ATTRIBUTE TYPE  ATTRIBUTE VALUE
_
Interface Stability     Committed
_
Standard        See \fBstandards\fR(5).
.TE

.SS "ksh93"
.sp

.sp
.TS
box;
c | c
l | l .
```



```
ATTRIBUTE TYPE  ATTRIBUTE VALUE
_
Interface Stability     Uncommitted
.TE

.SH SEE ALSO
.sp
.LP
\fBksh93\fR(1), \fBwait\fR(1), \fBalarm\fR(2), \fBsleep\fR(3C),
\fBwait\fR(3UCB), \fBattributes\fR(5), \fBenviron\fR(5), \fBstandards\fR(5)
.SH NOTES
.sp
.LP
If the \fBsleep\fR utility receives a \fBSIGALRM\fR signal, one of the
following actions is taken:
.RS +4
.TP
.ie t \(bu
.el o
Terminate normally with a zero exit status.
.RE
.RS +4
.TP
.ie t \(bu
.el o
Effectively ignore the signal.
.RE
.sp
.LP
The \fBsleep\fR utility takes the standard action for all other signals.
.sp
.LP
The behavior for input values such as "NaN" (not-a-number) or negative values
is undefined.
```

```
**********************************************************
   14058 Tue Sep 10 18:35:19 2013
new/usr/src/man/man1/sum.1
4023 - Typo in file(1) manpage and various others
**********************************************************
'\" te
.\" Copyright (c) 1992, X/Open Company Limited  All Rights Reserved
.\" Copyright 1989 AT&T
.\" Portions Copyright (c) 2009, Sun Microsystems, Inc. All Rights Reserved.
.\" Sun Microsystems, Inc. gratefully acknowledges The Open Group for permission
.\" The Institute of Electrical and Electronics Engineers and The Open Group, ha
.\" are reprinted and reproduced in electronic form in the Sun OS Reference Manu
.\" and Electronics Engineers, Inc and The Open Group. In the event of any discr
.\"  This notice shall appear on any product containing this material.
.\" 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 SUM 1 "Aug 11, 2009"
.SH NAME
sum \- print checksum and block count for a file
.SH SYNOPSIS
.LP
.nf
\fB/usr/bin/sum\fR [-abBchHlLpPrRstTw] [-x method] [\fIfile\fR...]
.fi

.SS "ksh93"
.LP
.nf
sum [-abBchHlLpPrRstTw] [-x method] [\fIfile\fR...]
.fi

.SH DESCRIPTION
.sp
.LP
The \fBsum\fR utility and ksh93 built-in command list the checksum, and for
most methods the block count, for each file argument. The standard input is
read if there are no file arguments.
.sp
.LP
The \fBgetconf\fR(1) \fBUNIVERSE\fR determines the default sum method: att for
the att universe, bsd otherwise. The default for the other commands is the
command name itself. The att method is a true sum, all others are order
dependent.
.sp
.LP
Method names consist of a leading identifier and 0 or more options separated by
-.
.sp
.LP
\fBgetconf\fR \fBPATH_RESOLVE\fR determines how symbolic links are handled.
This can be explicitly overridden by the \fB--logical\fR, \fB--metaphysical\fR,
and \fB--physical\fR options below. \fBPATH_RESOLVE\fR can be one of:
.sp
.ne 2
.na
\fB\fB--logical\fR\fR
.ad
.RS 18n
Follow all symbolic links.
.RE

.sp
.ne 2
.na
\fB\fB--metaphysical\fR\fR
```



```
.ad
.RS 18n
Follow command argument symbolic links, otherwise do not follow.
.RE

.sp
.ne 2
.na
\fB\fB--physical\fR\fR
.ad
.RS 18n
Do not follow symbolic links.
.RE

.SH OPTIONS
.sp
.LP
The following options are supported for \fB/usr/bin/sum\fR:
.sp
.ne 2
.na
\fB\fB-a\fR\fR
.ad
.br
.na
\fB\fB--all\fR\fR
.ad
.sp .6
.RS 4n
List the checksum for all files. Use with \fB--total\fR to list both individual
and total checksums and block counts.
.RE

.sp
.ne 2
.na
\fB\fB-b\fR\fR
.ad
.br
.na
\fB\fB--binary\fR\fR
.ad
.sp .6
.RS 4n
Read files in binary mode. This is the default.
.RE

.sp
.ne 2
.na
\fB\fB-B\fR\fR
.ad
.br
.na
\fB\fB--scale=scale\fR\fR
.ad
.sp .6
.RS 4n
Block count scale (bytes per block) override for methods that include size in
the output. The default is method-specific.
.RE

.sp
.ne 2
.na
\fB\fB-c\fR\fR
```

```
.ad
.br
.na
\fB\fB--check\fR\fR
.ad
.sp .6
.RS 4n
Each file is interpreted as the output from a previous sum. If \fB--header\fR
or \fB--permissions\fR was specified in the previous sum then the checksum
method is automatically determined, otherwise \fB--method\fR must be specified.
The listed checksum is compared with the current value and a warning is issued
for each file that does not match. If file was generated by
\fB--permissions\fR, then the file mode, user and group are also checked. Empty
lines, lines starting with #<space>, or the line # are ignored. Lines
containing no blanks are interpreted as [no]name[=\fIvalue\fR] options:
.sp
.ne 2
.na
\fB\fBmethod=name\fR\fR
.ad
.sp .6
.RS 4n
Checksum method to apply to subsequent lines.
.RE

.sp
.ne 2
.na
\fB\fBpermissions\fR\fR
.ad
.sp .6
.RS 4n
Subsequent lines were generated with \fB--permissions\fR.
.RE

.RE

.sp
.ne 2
.na
\fB\fB-h\fR\fR
.ad
.br
.na
\fB\fB--header\fR\fR
.ad
.sp .6
.RS 4n
Print the checksum method as the first output line. Used with \fB--check\fR and
\fB--permissions\fR.
.RE

.sp
.ne 2
.na
\fB\fB-l\fR\fR
.ad
.br
.na
\fB\fB--list\fR\fR
.ad
.sp .6
.RS 4n
Each file is interpreted as a list of files, one per line, that is checksummed.
.RE
```



```
.sp
.ne 2
.na
\fB\fB-p\fR\fR
.ad
.br
.na
\fB\fB--permissions\fR\fR
.ad
.sp .6
.RS 4n
If \fB--check\fR is not specified then list the file mode, user and group
between the checksum and path. User and group matching the caller are output as
-. If \fB--check\fR is specified then the mode, user and group for each path in
file are updated if necessary to match those in file. A warning is printed on
the standard error for each changed file.
.RE

.sp
.ne 2
.na
\fB\fB-R\fR\fR
.ad
.br
.na
\fB\fB--recursive\fR\fR
.ad
.sp .6
.RS 4n
Recursively checksum the contents of directories.
.RE

.sp
.ne 2
.na
\fB\fB-t\fR\fR
.ad
.br
.na
\fB\fB--total\fR\fR
.ad
.sp .6
.RS 4n
List only the total checksum and block count of all files. \fB--all\fR
\fB--total\fR lists each checksum and the total. The total checksum and block
count may be different from the checksum and block count of the catenation of
all files due to partial blocks that may occur when the files are treated
separately.
.RE

.sp
.ne 2
.na
\fB\fB-T\fR\fR
.ad
.br
.na
\fB\fB--text\fR\fR
.ad
.sp .6
.RS 4n
Read files in text mode (for example, treat \er\en as \en).
Read files in text mode (for example, treat \r\n as \n).
.RE

.sp
```

```
.ne 2
.na
\fB\fB-w\fR\fR
.ad
.br
.na
\fB\fB--warn\fR\fR
.ad
.sp .6
.RS 4n
Warn about invalid \fB--check\fR lines. On by default; \fB-w\fR means
\fB--nowarn\fR.
.RE

.sp
.ne 2
.na
\fB\fB-x\fR\fR
.ad
.br
.na
\fB\fB--method|algorithm=method\fR\fR
.ad
.sp .6
.RS 4n
Specifies the checksum method to apply. Parenthesized method options are
readonly implementation details.
.sp
.ne 2
.na
\fB\fBatt\fR|\fBsys5\fR|\fBs5\fR|\fBdefault\fR\fR
.ad
.sp .6
.RS 4n
The system 5 release 4 checksum. This is the default for sum when \fBgetconf\fR
\fBUNIVERSE\fR is \fBatt\fR. This is the only true sum; all of the other
methods are order dependent.
.RE

.sp
.ne 2
.na
\fB\fBast4\fR|\fB32x4\fR|\fBtw\fR\fR
.ad
.sp .6
.RS 4n
The \fBast\fR 128 bit \fBPRNG\fR hash generated by catenating 4 separate 32 bit
\fBPNRG\fR hashes. The block count is not printed.
.RE

.sp
.ne 2
.na
\fB\fBbsd\fR|\fBucb\fR\fR
.ad
.sp .6
.RS 4n
The BSD checksum.
.RE

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



```
.RS 4n
32 bit CRC (cyclic redundancy check).
.sp
.ne 2
.na
\fB\fBpolynomial\fR=\fImask\fR\fR
.ad
.sp .6
.RS 4n
The 32 bit \fBcrc\fR polynomial bitmask with implicit bit 32. The default value
is 0xedb88320.
.RE

.sp
.ne 2
.na
\fB\fBdone\fR[=\fInumber\fR]\fR
.ad
.sp .6
.RS 4n
XOR the final \fBcrc\fR value with number. 0xffffffff is used if number is
omitted. The option value may be omitted. The default value is 0.
.RE

.sp
.ne 2
.na
\fB\fBinit\fR[=\fInumber\fR]\fR
.ad
.sp .6
.RS 4n
The initial \fBcrc\fR value. 0xffffffff is used if number is omitted. The
option value may be omitted. The default value is 0.
.RE

.sp
.ne 2
.na
\fB\fBrotate\fR\fR
.ad
.sp .6
.RS 4n
XOR each input character with the high order \fBcrc\fR byte (instead of the low
order).
.RE

.sp
.ne 2
.na
\fB\fBsize\fR[=\fInumber\fR]\fR
.ad
.sp .6
.RS 4n
Include the total number of bytes in the crc. number, if specified, is first
XOR'd into the size. The option value may be omitted. The default value is 0.
.RE

.RE

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

```
391 32 bit \fBPRNG\fR (pseudo random number generator) hash.
392 .sp
393 .ne 2
394 .na
395 \fB\fBmpy\fR=\fInumber\fR\fR
396 .ad
397 .RS 17n
398 The 32 bit \fBPRNG\fR multiplier. The default value is 0x01000193.
399 .RE

401 .sp
402 .ne 2
403 .na
404 \fB\fBadd\fR=\fInumber\fR\fR
405 .ad
406 .RS 17n
407 The 32 bit \fBPRNG\fR addend. The default value is 0.
408 .RE

410 .sp
411 .ne 2
412 .na
413 \fB\fBinit\fR[=\fInumber\fR]\fR
414 .ad
415 .RS 17n
416 The \fBPRNG\fR initial value. 0xffffffff is used if number is omitted. The
417 option value may be omitted. The default value is 0x811c9dc5.
418 .RE

420 .RE

422 .sp
423 .ne 2
424 .na
425 \fB\fBmd4\fR|\fBMD4\fR\fR
426 .ad
427 .sp .6
428 .RS 4n
429 \fBRFC1320\fR \fBMD4\fR message digest. Cryptographically weak. The block count
430 is not printed.  (version) \fBmd4\fR (\fBsolaris\fR \fB-lmd\fR) 2005-07-26
431 .RE

433 .sp
434 .ne 2
435 .na
436 \fB\fBmd5\fR|\fBMD5\fR\fR
437 .ad
438 .sp .6
439 .RS 4n
440 \fBRFC1321\fR \fBMD5\fR message digest. Cryptographically weak. The block count
441 is not printed. (version) \fBmd5\fR (\fBsolaris\fR \fB-lmd\fR) 2005-07-26
442 .RE

444 .sp
445 .ne 2
446 .na
447 \fB\fBsha1\fR|\fBSHA1\fR|\fBsha-1\fR|\fBSHA-1\fR\fR
448 .ad
449 .sp .6
450 .RS 4n
451 \fBRFC3174\fR / \fBFIPS 180-1\fR \fBSHA-1\fR secure hash algorithm 1.
452 Cryptographically weak. The block count is not printed. (version) \fBsha1\fR
453 (\fBsolaris\fR \fB-lmd\fR) 2005-07-26
454 .RE

456 .sp
```



```
457 .ne 2
458 .na
459 \fB\fBsha256\fR|\fBsha-256\fR|\fBSHA256\fR|\fBSHA-256\fR\fR
460 .ad
461 .sp .6
462 .RS 4n
463 \fBFIPS 180-2\fR \fBSHA256\fR secure hash algorithm. The block count is not
464 printed. (version) \fBsha256\fR (\fBsolaris\fR \fB-lmd\fR) 2005-07-26
465 .RE

467 .sp
468 .ne 2
469 .na
470 \fB\fBsha384\fR|\fBsha-384\fR|\fBSHA384\fR|\fBSHA-384\fR\fR
471 .ad
472 .sp .6
473 .RS 4n
474 \fBFIPS 180-2\fR \fBSHA384\fR secure hash algorithm. The block count is not
475 printed. (version) \fBsha384\fR (\fBsolaris\fR \fB-lmd\fR) 2005-07-26
476 .RE

478 .sp
479 .ne 2
480 .na
481 \fB\fBsha512\fR|\fBsha-512\fR|\fBSHA512\fR|\fBSHA-512\fR\fR
482 .ad
483 .sp .6
484 .RS 4n
485 \fBFIPS 180-2\fR \fBSHA512\fR secure hash algorithm. The block count is not
486 printed. (version) \fBsha512\fR (\fBsolaris\fR \fB-lmd\fR) 2005-07-26
487 .RE

489 .sp
490 .ne 2
491 .na
492 \fB\fBposix\fR|\fBcksum\fR|\fBstd\fR|\fBstandard\fR\fR
493 .ad
494 .sp .6
495 .RS 4n
496 The \fBposix 1003.2-1992\fR 32 bit \fBcrc\fR checksum. This is the default
497 \fBcksum\fR(1) method. Shorthand for \fBcrc-0x04c11db7-rotate-done-size\fR.
498 .RE

500 .sp
501 .ne 2
502 .na
503 \fB\fBzip\fR\fR
504 .ad
505 .sp .6
506 .RS 4n
507 The \fBzip\fR(1) \fBcrc\fR. Shorthand for \fBcrc-0xedb88320-init-done\fR.
508 .RE

510 .sp
511 .ne 2
512 .na
513 \fB\fBfddi\fR\fR
514 .ad
515 .sp .6
516 .RS 4n
517 The \fBFDDI\fR \fBcrc\fR. Shorthand for
518 \fBcrc-0xedb88320-size\fR=\fB0xcc55cc55\fR.
519 .RE

521 .sp
522 .ne 2
```

```
.na
\fB\fBfnv\fR|\fBfnv1\fR\fR
.ad
.sp .6
.RS 4n
The \fBFowler-Noll-Vo\fR 32 bit \fBPRNG\fR hash with non-zero initializer
(\fBFNV-1\fR). Shorthand for \fBprng-0x01000193-init\fR=\fB0x811c9dc5\fR.
.RE

.sp
.ne 2
.na
\fB\fBast\fR|\fBstrsum\fR\fR
.ad
.sp .6
.RS 4n
The \fBast\fR \fBstrsum\fR \fBPRNG\fR hash. Shorthand for
\fBprng-0x63c63cd9-add\fR=\fB0x9c39c33d\fR.
.RE

.RE

.sp
.ne 2
.na
\fB\fB-L\fR\fR
.ad
.br
.na
\fB\fB--logical\fR|\fBfollow\fR\fR
.ad
.sp .6
.RS 4n
Follow symbolic links when traversing directories. The default is determined by
\fBgetconf\fR \fBPATH_RESOLVE\fR.
.RE

.sp
.ne 2
.na
\fB\fB-H\fR\fR
.ad
.br
.na
\fB\fB--metaphysical\fR\fR
.ad
.sp .6
.RS 4n
Follow command argument symbolic links, otherwise do not follow symbolic links
when traversing directories. The default is determined by \fBgetconf\fR
\fBPATH_RESOLVE\fR.
.RE

.sp
.ne 2
.na
\fB\fB-P\fR\fR
.ad
.br
.na
\fB\fB--physical\fR\fR
.ad
.sp .6
.RS 4n
Do not follow symbolic links when traversing directories. The default is
determined by \fBgetconf\fR \fBPATH_RESOLVE\fR.
```



```
.RE

.sp
.ne 2
.na
\fB\fB-r\fR\fR
.ad
.br
.na
\fB\fB--bsd\fR\fR
.ad
.sp .6
.RS 4n
Equivalent to \fB--method=bsd\fR \fB--scale=512\fR for compatibility with other
sum implementations.
.RE

.sp
.ne 2
.na
\fB\fB-s\fR\fR
.ad
.br
.na
\fB\fB--sysv\fR\fR
.ad
.sp .6
.RS 4n
Equivalent to \fB--method=sys5\fR for compatibility with other sum
implementations.
.RE

.sp
.ne 2
.na
\fB\fB-S\fR\fR
.ad
.br
.na
\fB\fB--silent\fR|\fBstatus\fR\fR
.ad
.sp .6
.RS 4n
No output for \fB--check\fR;  0 exit status means all sums matched, non-0 means
at least one sum failed to match. Ignored for \fB--permissions\fR.
.RE

.SH OPERANDS
.sp
.LP
The following operands are supported:
.sp
.ne 2
.na
\fB\fIfile\fR\fR
.ad
.RS 8n
A path name of a file.  If no files are named, the standard input is used.
.RE

.SH USAGE
.sp
.LP
See \fBlargefile\fR(5) for the description of the behavior of \fBsum\fR when
encountering files greater than or equal to 2 Gbyte ( 2^31 bytes).
.SH ENVIRONMENT VARIABLES
```

```
655 .sp
656 .LP
657 See \fBenviron\fR(5) for descriptions of the following environment variables
658 that affect the execution of  \fBsum\fR: \fBLC_CTYPE\fR, \fBLC_MESSAGES\fR, and
659 \fBNLSPATH\fR.
660 .SH EXIT STATUS
661 .sp
662 .LP
663 The following exit values are returned.
664 .sp
665 .ne 2
666 .na
667 \fB\fB0\fR\fR
668 .ad
669 .RS 6n
670 Successful completion.
671 .RE

673 .sp
674 .ne 2
675 .na
676 \fB\fB>0\fR\fR
677 .ad
678 .RS 6n
679 An error occurred.
680 .RE

682 .SH ATTRIBUTES
683 .sp
684 .LP
685 See \fBattributes\fR(5) for descriptions of the following attributes:
686 .sp

688 .sp
689 .TS
690 box;
691 c | c
692 l | l .
693 ATTRIBUTE TYPE  ATTRIBUTE VALUE
694 _
695 CSI     Enabled
696 .TE

698 .SH SEE ALSO
699 .sp
700 .LP
701 \fBcksum\fR(1), \fBgetconf\fR(1), \fBksh93\fR(1), \fBsum\fR(1B), \fBwc\fR(1),
702 \fBzip\fR(1) , \fBlibmd\fR(3LIB), \fBattributes\fR(5), \fBenviron\fR(5),
703 \fBlargefile\fR(5)
704 .SH DIAGNOSTICS
705 .sp
706 .LP
707 \fBRead error\fR is indistinguishable from end of file on most devices. Check
708 the block count.
709 .SH NOTES
710 .sp
711 .LP
712 Portable applications should use \fBcksum\fR(1). The default algorithm for this
713 command is defined in the POSIX standard and is identical across platforms.
714 .sp
715 .LP
716 \fBsum\fR and \fBusr/ucb/sum\fR (see \fBsum\fR(1B)) return different checksums.
```

```
**********************************************************
   11721 Tue Sep 10 18:35:19 2013
new/usr/src/man/man1c/uuencode.1c
4023 - Typo in file(1) manpage and various others
**********************************************************
   1 '\" te
   2 .\"  Copyright 1989 AT&T  Copyright (c) 1980 Regents of the University of Califo
   3 .\" Sun Microsystems, Inc. gratefully acknowledges The Open Group for permission
   4 .\" http://www.opengroup.org/bookstore/.
   5 .\" The Institute of Electrical and Electronics Engineers and The Open Group, ha
   6 .\"  This notice shall appear on any product containing this material.
   7 .TH UUENCODE 1C "Sep 10, 2013"
   7 .TH UUENCODE 1C "Aug 6, 2003"
   8 .SH NAME
   9 uuencode, uudecode \- encode a binary file, or decode its encoded
  10 representation
  11 .SH SYNOPSIS
  12 .LP
  13 .nf
  14 \fBuuencode\fR [\fIsource-file\fR] \fIdecode_pathname\fR
  15 .fi

  17 .LP
  18 .nf
  19 \fBuuencode\fR [\fB-m\fR] [\fIsource-file\fR] \fIdecode_pathname\fR
  20 .fi

  22 .LP
  23 .nf
  24 \fBuudecode\fR [\fB-p\fR] [\fIencoded-file\fR]
  25 .fi

  27 .LP
  28 .nf
  29 \fBuudecode\fR [\fB-o\fR \fIoutfile\fR] [\fIencoded-file\fR]
  30 .fi

  32 .SH DESCRIPTION
  33 .sp
  34 .LP
  35 These commands encode and decode files as follows:
  36 .SS "uuencode"
  37 .sp
  38 .LP
  39 The \fBuuencode\fR utility converts a binary file into an encoded
  40 representation that can be sent using \fBmail\fR(1). It encodes the contents of
  41 \fIsource-file\fR, or the standard input if no \fIsource-file\fR argument is
  42 given. The \fIdecode_pathname\fR argument is required. The
  43 \fIdecode_pathname\fR is included in the encoded file's header as the name of
  44 the file into which \fBuudecode\fR is to place the binary (decoded) data.
  45 \fBuuencode\fR also includes the permission modes of \fIsource-file\fR (except
  46 \fBsetuid\fR, \fBsetgid\fR, and sticky-bits), so that \fIdecode_pathname\fR is
  47 recreated with those same permission modes.
  48 .SS "uudecode"
  49 .sp
  50 .LP
  51 The \fBuudecode\fR utility reads an \fIencoded-file\fR, strips off any leading
  52 and trailing lines added by mailer programs, and recreates the original binary
  53 data with the filename and the mode specified in the header.
  54 .sp
  55 .LP
  56 The encoded file is an ordinary portable character set text file; it can be
  57 edited by any text editor. It is best only to change the mode or
  58 \fIdecode_pathname\fR in the header to avoid corrupting the decoded binary.
  59 .SH OPTIONS
  60 .sp
```



```
  61 .LP
  62 The following options are supported:
  63 .SS "uuencode"
  64 .sp
  65 .ne 2
  66 .na
  67 \fB\fB-m\fR\fR
  68 .ad
  69 .RS 6n
  70 Encodes \fIsource-file\fR using Base64 encoding and sends it to standard
  71 output.
  72 .RE

  74 .SS "uudecode"
  75 .sp
  76 .ne 2
  77 .na
  78 \fB\fB-o\fR \fIoutfile\fR\fR
  79 .ad
  80 .RS 14n
  81 Specifies a file pathname that should be used instead of any pathname contained
  82 in the input data. Specifying an \fIoutfile\fR option-argument of
  83 \fB/dev/stdout\fR indicates standard output.This allows \fBuudecode\fR to be
  84 used in a pipeline.
  85 .RE

  87 .sp
  88 .ne 2
  89 .na
  90 \fB\fB-p\fR\fR
  91 .ad
  92 .RS 14n
  93 Decodes \fIencoded-file\fR and sends it to standard output. This allows
  94 \fBuudecode\fR to be used in a pipeline.
  95 .RE

  97 .SH OPERANDS
  98 .sp
  99 .LP
 100 The following operands are supported by \fBuuencode\fR and \fBuudecode\fR:
 101 .SS "uuencode"
 102 .sp
 103 .ne 2
 104 .na
 105 \fB\fIdecode_pathname\fR\fR
 106 .ad
 107 .RS 19n
 108 The pathname of the file into which the \fBuudecode\fR utility will place the
 109 decoded file. If there are characters in \fIdecode_pathname\fR that are not in
 110 the portable filename character set, the results are unspecified.
 111 .RE

 113 .sp
 114 .ne 2
 115 .na
 116 \fB\fIsource-file\fR\fR
 117 .ad
 118 .RS 19n
 119 A pathname of the file to be encoded.
 120 .RE

 122 .SS "uudecode"
 123 .sp
 124 .ne 2
 125 .na
 126 \fB\fIencoded-file\fR\fR
```

```
.ad
.RS 16n
The pathname of a file containing the output of \fBuuencode\fR.
.RE

.SH USAGE
.sp
.LP
See \fBlargefile\fR(5) for the description of the behavior of \fBuuencode\fR
and \fBuudecode\fR when encountering files greater than or equal to 2 Gbyte (
2^31 bytes).
.SH ENVIRONMENT VARIABLES
.sp
.LP
See \fBenviron\fR(5) for descriptions of the following environment variables
that affect the execution of \fBuuencode\fR and \fBuudecode\fR: \fBLANG\fR,
\fBLC_ALL\fR, \fBLC_CTYPE\fR, \fBLC_MESSAGES\fR, and \fBNLSPATH\fR.
.SH OUTPUT
.sp
.LP
stdout
.SS "uuencode Base64 Algorithm"
.sp
.LP
The standard output is a text file, encoded in the character set of the current
locale, that begins with the line:
.sp
.in +2
.nf
begin-base64 %s %s\en, \fImode\fR, \fIdecode_pathname\fR
begin-base64 %s %s\n, \fImode\fR, \fIdecode_pathname\fR
.fi
.in -2
.sp

.sp
.LP
and ends with the line:
.sp
.in +2
.nf
====
.fi
.in -2
.sp

.sp
.LP
In both cases, the lines have no preceding or trailing blank characters.
.sp
.LP
The encoding process represents 24-bit groups of input bits as output strings
of  four encoded characters. Proceeding from left to right, a 24-bit input
group is formed by concatenating three 8-bit input groups. Each 24-bit input
group  is then treated as four concatenated 6-bit groups, each of which is
translated into a single digit in the Base64 alphabet. When encoding a bit
stream by means of the Base64 encoding, the bit stream is presumed to be
ordered with the most-significant bit first. That is, the first bit in the
stream is the high-order bit in the first byte, and the eighth bit is the
low-order bit in the first byte, and so on. Each 6-bit group is used as an
index into an array of 64 printable characters, as shown in the following
table.
.sp
.in +2
.nf
Value Encoding   Value Encoding   Value Encoding   Value Encoding
```



```
  0      A         17     R         34     i         51     z
  1      B         18     S         35     j         52     0
  2      C         19     T         36     k         53     1
  3      D         20     U         37     l         54     2
  4      E         21     V         38     m         55     3
  5      F         22     W         39     n         56     4
  6      G         23     X         40     o         57     5
  7      H         24     Y         41     p         58     6
  8      I         25     Z         42     q         59     7
  9      J         26     a         43     r         60     8
 10      K         27     b         44     s         61     9
 11      L         28     c         45     t         62     +
 12      M         29     d         46     u         63     /
 13      N         30     e         47     v
 14      O         31     f         48     w       (pad)    =
 15      P         32     g         49     x
 16      Q         33     h         50     y
.fi
.in -2
.sp

.sp
.LP
The character referenced by the index is placed in the output string.
.sp
.LP
The output stream (encoded bytes) is represented in lines of no more than 76
characters each. All line breaks or other characters not found in the table are
ignored by decoding software (see \fBuudecode\fR).
.sp
.LP
Special processing is performed if fewer than 24 bits are available at the end
of a message or encapsulated part of a message. A full encoding quantum is
always completed at the end of a message. When fewer than 24 input bits are
available in an input group, zero bits are added on the right to form an
integral number of 6-bit groups. Output character positions that are not
required to represent actual input data are set to the equals (\fB=\fR)
character. Since all Base64 input is an integral number of octets, only the
following cases can arise:
.RS +4
.TP
1.
The final quantum of encoding input is an integral multiple of 24 bits.
Here, the final unit of encoded output is an integral multiple of four
characters with no '\fB=\fR ' padding.
.RE
.RS +4
.TP
2.
The final quantum of encoding input is exactly 16 bits. Here, the final unit
of encoded output is three characters followed by one '\fB=\fR' padding
character.
.RE
.RS +4
.TP
3.
The final quantum of encoding input is exactly 8 bits. Here, the final unit
of encoded output is two characters followed by two '\fB=\fR' padding
characters.
.RE
.sp
.LP
A terminating "\fB====\fR" evaluates to nothing and denotes the end of the
encoded data.
.SS "uuencode Historical Algorithm"
.sp
```

```
.LP
The standard output is a text file (encoded in the character set of the current
locale) that begins with the line:
.sp
.in +2
.nf
begin %s %s\en, \fImode\fR, \fIdecode_pathname\fR
.fi
.in -2
.sp

.sp
.LP
and ends with the line:
.sp
.in +2
.nf
end\en
.fi
.in -2
.sp

.sp
.LP
In both cases, the lines have no preceding or trailing blank characters.
.sp
.LP
The algorithm that is used for lines between \fBbegin\fR and \fBend\fR takes
three octets as input and writes four characters of output by splitting the
input at six-bit intervals into four octets, containing data in the lower six
bits only. These octets are converted to characters by adding a value of
\fB0x20\fR to each octet, so that each octet is in the range
\fB0x20\(mi0x5f\fR, and each octet is assumed to represent a printable
character. Each octect is then translated into the corresponding character
codes for the codeset in use in the current locale. For example, the octet
\fB0x41\fR, representing '\fBA\fR', would be translated to '\fBA\fR' in the
current codeset, such as \fB0xc1\fR if the codeset were \fBEBCDIC\fR.
.sp
.LP
Where the bits of two octets are combined, the least significant bits of the
first octet are shifted left and combined with the most significant bits of the
second octet shifted right. Thus, the three octets \fBA\fR, \fBB\fR, \fBC\fR
are converted into the four octets:
.sp
.in +2
.nf
0x20 + (( A >> 2                          ) & 0x3F)
0x20 + (((A << 4)  ((B >> 4) & 0xF)) & 0x3F)
0x20 + (((B << 2)  ((C >> 6) & 0x3)) & 0x3F)
0x20 + (( C                                ) & 0x3F)
.fi
.in -2
.sp

.sp
.LP
These octets are then translated into the local character set.
.sp
.LP
Each encoded line contains a length character, equal to the number of
characters to be decoded plus \fB0x20\fR translated to the local character set
as described above, followed by the encoded characters. The maximum number of
octets to be encoded on each line is 45.
.SH EXIT STATUS
.sp
.LP
```



```
The following exit values are returned:
.sp
.ne 2
.na
\fB\fB0\fR\fR
.ad
.RS 6n
Successful completion.
.RE

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

.SH SEE ALSO
.sp
.LP
\fBmail\fR(1), \fBmailx\fR(1), \fBuucp\fR(1C), \fBuux\fR(1C),
\fBattributes\fR(5), \fBenviron\fR(5), \fBlargefile\fR(5), \fBstandards\fR(5)
.SH NOTES
.sp
.LP
The size of the encoded file is expanded by 35% (3 bytes become 4, plus control
information), causing it to take longer to transmit than the equivalent binary.
.sp
.LP
The user on the remote system who is invoking \fBuudecode\fR (typically
\fBuucp\fR) must have write permission on the file specified in the
\fIdecode_pathname\fR.
.sp
.LP
If you invoke \fBuuencode\fR and then execute \fBuudecode\fR on a file in the
same directory, you will overwrite the original file.
```

```
**********************************************************
   19042 Tue Sep 10 18:35:19 2013
new/usr/src/man/man1m/fdisk.1m
4023 - Typo in file(1) manpage and various others
**********************************************************
   1 '\" te
   2 .\" Copyright (c) 2008, Sun Microsystems, Inc. All Rights Reserved
   3 .\" The contents of this file are subject to the terms of the Common Development
   4 .\"  See the License for the specific language governing permissions and limitat
   5 .\" the fields enclosed by brackets "[]" replaced with your own identifying info
   6 .TH FDISK 1M "Sep 10, 2013"
   6 .TH FDISK 1M "Jul 2, 2009"
   7 .SH NAME
   8 fdisk \- create or modify fixed disk partition table
   9 .SH SYNOPSIS
  10 .LP
  11 .nf
  12 \fBfdisk\fR [\fB-o\fR \fIoffset\fR] [\fB-s\fR \fIsize\fR] [\fB-P\fR \fIfill_patt
  13      [\fB-w\fR | \fB-r\fR | \fB-d\fR | \fB-n\fR | \fB-I\fR | \fB-B\fR | \fB-t\fR
  14      [-\fB-F\fR \fIfdisk_file\fR] [ [\fB-v\fR] \fB-W\fR {\fIfdisk_file\fR | \(mi
  15      [\fB-h\fR] [\fB-b\fR \fImasterboot\fR]
  16      [\fB-A\fR \fIid\fR : \fIact\fR : \fIbhead\fR : \fIbsect\fR : \fIbcyl\fR : \
  17          \fIecyl\fR : \fIrsect\fR : \fInumsect\fR]
  18      [\fB-D\fR \fIid\fR : \fIact\fR : \fIbhead:\fR \fIbsect\fR : \fIbcyl\fR : \f
  19          \fIecyl\fR : \fIrsect\fR : \fInumsect\fR] \fIrdevice\fR
  20 .fi

  22 .SH DESCRIPTION
  23 .sp
  24 .LP
  25 This command is used to do the following:
  26 .RS +4
  27 .TP
  28 .ie t \(bu
  29 .el o
  30 Create and modify an \fBfdisk\fR partition table on x86 systems
  31 .RE
  32 .RS +4
  33 .TP
  34 .ie t \(bu
  35 .el o
  36 Create and modify an \fBfdisk\fR partition table on removable media on SPARC or
  37 x86 systems
  38 .RE
  39 .RS +4
  40 .TP
  41 .ie t \(bu
  42 .el o
  43 Install the master boot record that is put in the first sector of the fixed
  44 disk on x86 systems only
  45 .RE
  46 .sp
  47 .LP
  48 This table is used by the first-stage bootstrap (or firmware) to identify parts
  49 of the disk reserved for different operating systems, and to identify the
  50 partition containing the second-stage bootstrap (the \fIactive\fR Solaris
  51 partition). The \fIrdevice\fR argument must be used to specify the raw device
  52 associated with the fixed disk, for example, \fB/dev/rdsk/c0t0d0p0\fR.
  53 .sp
  54 .LP
  55 The program can operate in three different modes. The first is interactive
  56 mode. In interactive mode, the program displays the partition table as it
  57 exists on the disk, and then presents a menu allowing the user to modify the
  58 table. The menu, questions, warnings, and error messages are intended to be
  59 self-explanatory.
  60 .sp
```



```
  61 .LP
  62 In interactive mode, if there is no partition table on the disk, the user is
  63 given the options of creating a default partitioning or specifying the initial
  64 table values. The default partitioning allocates the entire disk for the
  65 Solaris system and makes the Solaris system partition active. In either case,
  66 when the initial table is created, \fBfdisk\fR also writes out the first-stage
  67 bootstrap (x86 only) code along with the partition table. In this mode, (x86
  68 only) when creating an  entry for a non-EFI partition on a disk that is larger
  69 than 2 TB (terabytes), \fBfdisk\fR warns that the maximum size of the partition
  70 is 2 TB. Under these conditions percentages displayed by \fBfdisk\fR are based
  71 on 2 TB.
  72 .sp
  73 .LP
  74 The second mode of operation is used for automated entry addition, entry
  75 deletion, or replacement of the entire \fBfdisk\fR table. This mode can add or
  76 delete an entry described on the command line. In this mode the entire
  77 \fBfdisk\fR table can be read in from a file replacing the original table.
  78 \fBfdisk\fR can also be used to create this file. There is a command line
  79 option that will cause \fBfdisk\fR to replace any \fBfdisk\fR table with the
  80 default of the whole disk for the Solaris system.
  81 .sp
  82 .LP
  83 The third mode of operation is used for disk diagnostics. In this mode, a
  84 section of the disk can be filled with a user-specified pattern and mode
  85 sections of the disk can also be read or written.
  86 .LP
  87 Note -
  88 .sp
  89 .RS 2
  90 The third mode of operation is not currently supported for extended partitions
  91 .RE
  92 .sp
  93 .LP
  94 When \fBfdisk\fR creates a partition, the space is allocated in the \fBfdisk\fR
  95 partition table, but the allocated disk space is not initialized.
  96 \fBnewfs\fR(1M) is required to create and write file system metadata to the new
  97 partition, and \fBformat\fR(1M) is required to write the VTOC or EFI/GPT
  98 metadata.
  99 .SS "Menu Options"
 100 .sp
 101 .LP
 102 The menu options for interactive mode given by the \fBfdisk\fR program are:
 103 .sp
 104 .ne 2
 105 .na
 106 \fB\fBCreate a partition\fR\fR
 107 .ad
 108 .sp .6
 109 .RS 4n
 110 This option allows the user to create a new partition. The maximum number of
 111 partitions is 4. The program will ask for the type of the partition (SOLARIS,
 112 MS-DOS, UNIX, or other). It will then ask for the size of the partition as a
 113 percentage of the disk. The user may also enter the letter \fBc\fR at this
 114 point, in which case the program will ask for the starting cylinder number and
 115 size of the partition in cylinders. If a \fBc\fR is not entered, the program
 116 will determine the starting cylinder number where the partition will fit. In
 117 either case, if the partition would overlap an existing partition or will not
 118 fit, a message is displayed and the program returns to the original menu.
 119 .RE

 121 .sp
 122 .ne 2
 123 .na
 124 \fB\fBChange Active (Boot from) partition\fR\fR
 125 .ad
 126 .sp .6
```

```
.RS 4n
This option allows the user to specify the partition where the first-stage
bootstrap will look for the second-stage bootstrap, otherwise known as the
\fIactive\fR partition.
.RE

.sp
.ne 2
.na
\fB\fBDelete a partition\fR\fR
.ad
.sp .6
.RS 4n
This option allows the user to delete a previously created partition. Note that
this will destroy all data in that partition.
.RE

.sp
.ne 2
.na
\fB\fBChange between Solaris and Solaris2 Partition IDs\fR\fR
.ad
.sp .6
.RS 4n
This option allows the user to switch between the current \fBfdisk\fR operating
system partition identifier and the previous one. This does not affect any data
in the disk partition and is provided for compatibility with older software.
.RE

.sp
.ne 2
.na
\fB\fBEdit/View extended partitions\fR\fR
.ad
.sp .6
.RS 4n
This option provides the extended partition menu to the user. Use the extended
partition menu to add and delete logical drives, change the sysid of the
logical drives, and display logical drive information. To commit the changes
made in the extended partition, you must return to the main menu using the
extended partition submenu option \fBr\fR. There is also an option to display
the list of options that the extended partition submenu supports. Given below
is the list:
.sp
.ne 2
.na
\fB\fBa\fR\fR
.ad
.RS 5n
Add a logical drive.
.sp
Use this submenu option to add a logical drive. There are three pieces of
information that are required: The beginning cylinder, the size (in cylinders
or in human readable form - KB, MB, or GB), and the partition ID. While
specifying the partition ID, there is an option (\fBI\fR) that you can use to
list the supported partitions.
.RE

.sp
.ne 2
.na
\fB\fBd\fR\fR
.ad
.RS 5n
Delete a logical drive.
.sp
```



```
Use this submenu option to delete a logical drive. The only input required is
the number of the logical drive that is to be deleted.
.RE

.sp
.ne 2
.na
\fB\fBh\fR\fR
.ad
.RS 5n
Display the help menu.
.sp
This submenu option displays the supported operations in the extended partition
submenu.
.RE

.sp
.ne 2
.na
\fB\fBi\fR\fR
.ad
.RS 5n
Change the id of the logical drive.
.sp
Use this submenu option to change the system ID of the existing logical drives.
A list of supported system IDs is displayed when you use the \fBI\fR option
when in this submenu.
.RE

.sp
.ne 2
.na
\fB\fBp\fR\fR
.ad
.RS 5n
Display the logical drive layout.
.sp
Displays the logical drive information to stdout. This output reflects any
changes made during the current run of the \fBfdisk\fR program. The changes are
not committed to the disk until return to the main menu (using the submenu
\fBr\fR) and choose the option to commit the changes to the disk.
.RE

.sp
.ne 2
.na
\fB\fBr\fR\fR
.ad
.RS 5n
Return to the main \fBfdisk\fR menu.
.sp
Exit the extended partition submenu and return to the main menu.
.RE

.RE

.sp
.LP
Use the following options to include your modifications to the partition table
at this time or to cancel the session without modifying the table:
.sp
.ne 2
.na
\fB\fBExit\fR\fR
.ad
.RS 10n
```

```
259 This option writes the new version of the table created during this session
260 with \fBfdisk\fR out to the fixed disk, and exits the program.
261 .RE

263 .sp
264 .ne 2
265 .na
266 \fB\fBCancel\fR\fR
267 .ad
268 .RS 10n
269 This option exits without modifying the partition table.
270 .RE

272 .SH OPTIONS
273 .sp
274 .LP
275 The following options apply to \fBfdisk\fR:
276 .sp
277 .ne 2
278 .na
279 \fB\fB-A\fR \fIid:act:bhead:bsect:bcyl:ehead:esect:ecyl:rsect:numsect\fR\fR
280 .ad
281 .sp .6
282 .RS 4n
283 Add a partition as described by the argument (see the \fB-F\fR option below for
284 the format). Use of this option will zero out the \fBVTOC\fR on the Solaris
285 partition if the \fBfdisk\fR table changes.
286 .RE

288 .sp
289 .ne 2
290 .na
291 \fB\fB-b\fR \fImaster_boot\fR\fR
292 .ad
293 .sp .6
294 .RS 4n
295 Specify the file \fImaster_boot\fR as the master boot program. The default
296 master boot program is \fB/usr/lib/fs/ufs/mboot\fR.
297 .RE

299 .sp
300 .ne 2
301 .na
302 \fB\fB-B\fR\fR
303 .ad
304 .sp .6
305 .RS 4n
306 Default to one Solaris partition that uses the whole disk. On an x86 machine,
307 if the disk is larger than 2 TB (terabytes), the default size of the Solaris
308 partition will be limited to 2 TB.
309 .RE

311 .sp
312 .ne 2
313 .na
314 \fB\fB-d\fR\fR
315 .ad
316 .sp .6
317 .RS 4n
318 Turn on verbose \fIdebug\fR mode. This will cause \fBfdisk\fR to print its
319 state on stderr as it is used. The output from this option should not be used
320 with \fB-F\fR.
321 .RE

323 .sp
324 .ne 2
```



```
325 .na
326 \fB\fB-D\fR \fIid:act:bhead:bsect:bcyl:ehead:esect:ecyl:rsect:numsect\fR\fR
327 .ad
328 .sp .6
329 .RS 4n
330 Delete a partition as described by the argument (see the \fB-F\fR option below
331 for the format). Note that the argument must be an exact match or the entry
332 will not be deleted! Use of this option will zero out the \fBVTOC\fR on the
333 Solaris partition if the \fBfdisk\fR table changes.
334 .RE

336 .sp
337 .ne 2
338 .na
339 \fB\fB-E\fR\fR
340 .ad
341 .sp .6
342 .RS 4n
343 Create an \fBEFI\fR partition that uses the entire disk.
344 .RE

346 .sp
347 .ne 2
348 .na
349 \fB\fB-F\fR \fIfdisk_file\fR\fR
350 .ad
351 .sp .6
352 .RS 4n
353 Use fdisk file \fIfdisk_file\fR to initialize table. Use of this option will
354 zero out the \fBVTOC\fR on the Solaris partition if the \fBfdisk\fR table
355 changes.
356 .sp
357 The \fIfdisk_file\fR contains four specification lines for the primary
358 partitions followed by specification lines for the logical drives. You must
359 have four lines for the primary partitions if there is at least one logical
360 drive. In this case, if the number of primary partitions to be configured is
361 less than four, the remaining lines should be filled with zeros.
362 .sp
363 Each line is composed of entries that are position-dependent, are separated by
364 whitespace or colons, and have the following format:
365 .sp
366 \fIid act bhead bsect bcyl ehead esect ecyl rsect numsect\fR
367 .sp
368 \&...where the entries have the following values:
369 .sp
370 .ne 2
371 .na
372 \fB\fIid\fR\fR
373 .ad
374 .RS 11n
375 This is the type of partition and the correct numeric values may be found in
376 \fBfdisk.h\fR.
377 .RE

379 .sp
380 .ne 2
381 .na
382 \fB\fIact\fR\fR
383 .ad
384 .RS 11n
385 This is the active partition flag; \fB0\fR means not active and \fB128\fR means
386 active. For logical drives, this flag will always be set to 0 even if specified
387 as 128 by the user.
388 .RE

390 .sp
```

```
391 .ne 2
392 .na
393 \fB\fIbhead\fR\fR
394 .ad
395 .RS 11n
396 This is the head where the partition starts. If this is set to \fB0\fR,
397 \fBfdisk\fR will correctly fill this in from other information.
398 .RE

400 .sp
401 .ne 2
402 .na
403 \fB\fIbsect\fR\fR
404 .ad
405 .RS 11n
406 This is the sector where the partition starts. If this is set to \fB0\fR,
407 \fBfdisk\fR will correctly fill this in from other information.
408 .RE

410 .sp
411 .ne 2
412 .na
413 \fB\fIbcyl\fR\fR
414 .ad
415 .RS 11n
416 This is the cylinder where the partition starts. If this is set to \fB0\fR,
417 \fBfdisk\fR will correctly fill this in from other information.
418 .RE

420 .sp
421 .ne 2
422 .na
423 \fB\fIehead\fR\fR
424 .ad
425 .RS 11n
426 This is the head where the partition ends. If this is set to \fB0\fR,
427 \fBfdisk\fR will correctly fill this in from other information.
428 .RE

430 .sp
431 .ne 2
432 .na
433 \fB\fIesect\fR\fR
434 .ad
435 .RS 11n
436 This is the sector where the partition ends. If this is set to \fB0\fR,
437 \fBfdisk\fR will correctly fill this in from other information.
438 .RE

440 .sp
441 .ne 2
442 .na
443 \fB\fIecyl\fR\fR
444 .ad
445 .RS 11n
446 This is the cylinder where the partition ends. If this is set to \fB0\fR,
447 \fBfdisk\fR will correctly fill this in from other information.
448 .RE

450 .sp
451 .ne 2
452 .na
453 \fB\fIrsect\fR\fR
454 .ad
455 .RS 11n
456 The relative sector from the beginning of the disk where the partition starts.
```



```
457 This must be specified and can be used by \fBfdisk\fR to fill in other fields.
458 For logical drives, you must make sure that there are at least 63 free sectors
459 before the \fIrsect\fR specified for a logical drive.
460 .RE

462 .sp
463 .ne 2
464 .na
465 \fB\fInumsect\fR\fR
466 .ad
467 .RS 11n
468 The size in sectors of this disk partition. This must be specified and can be
469 used by \fBfdisk\fR to fill in other fields.
470 .RE

472 .RE

474 .sp
475 .ne 2
476 .na
477 \fB\fB-g\fR\fR
478 .ad
479 .sp .6
480 .RS 4n
481 Get the label geometry for disk and display on stdout (see the \fB-S\fR option
482 for the format).
483 .RE

485 .sp
486 .ne 2
487 .na
488 \fB\fB-G\fR\fR
489 .ad
490 .sp .6
491 .RS 4n
492 Get the physical geometry for disk and display on stdout (see the \fB-S\fR
493 option for the format).
494 .RE

496 .sp
497 .ne 2
498 .na
499 \fB\fB-h\fR\fR
500 .ad
501 .sp .6
502 .RS 4n
503 Issue verbose message; message will list all options and supply an explanation
504 for each.
505 .RE

507 .sp
508 .ne 2
509 .na
510 \fB\fB-I\fR\fR
511 .ad
512 .sp .6
513 .RS 4n
514 Forgo device checks. This is used to generate a file image of what would go on
515 a disk without using the device. Note that you must use \fB-S\fR with this
516 option (see above).
517 .RE

519 .sp
520 .ne 2
521 .na
522 \fB\fB-n\fR\fR
```

```
523 .ad
524 .sp .6
525 .RS 4n
526 Don’t update \fBfdisk\fR table unless explicitly specified by another option.
527 If no other options are used, \fB-n\fR will only write the master boot record
528 to the disk. In addition, note that \fBfdisk\fR will not come up in interactive
529 mode if the \fB-n\fR option is specified.
530 .RE

532 .sp
533 .ne 2
534 .na
535 \fB\fB-o\fR \fIoffset\fR\fR
536 .ad
537 .sp .6
538 .RS 4n
539 Block offset from start of disk. This option is used for \fB-P\fR, \fB-r\fR,
540 and \fB-w\fR. Zero is assumed when this option is not used.
541 .RE

543 .sp
544 .ne 2
545 .na
546 \fB\fB-P\fR \fIfill_patt\fR\fR
547 .ad
548 .sp .6
549 .RS 4n
550 Fill disk with pattern \fIfill_patt\fR. \fIfill_patt\fR can be decimal or hex
551 and is used as number for constant long word pattern. If \fIfill_patt\fR is
552 \fB#\fR, then pattern is block # for each block. Pattern is put in each block
553 as long words and fills each block (see \fB-o\fR and \fB-s\fR).
554 .RE

556 .sp
557 .ne 2
558 .na
559 \fB\fB-r\fR\fR
560 .ad
561 .sp .6
562 .RS 4n
563 Read from disk and write to stdout. See \fB-o\fR and \fB-s\fR, which specify
564 the starting point and size of the operation.
565 .RE

567 .sp
568 .ne 2
569 .na
570 \fB\fB-R\fR\fR
571 .ad
572 .sp .6
573 .RS 4n
574 Treat disk as read-only. This is for testing purposes.
575 .RE

577 .sp
578 .ne 2
579 .na
580 \fB\fB-s\fR \fIsize\fR\fR
581 .ad
582 .sp .6
583 .RS 4n
584 Number of blocks to perform operation on (see \fB-o\fR).
585 .RE

587 .sp
588 .ne 2
```



```
589 .na
590 \fB\fB-S\fR \fIgeom_file\fR\fR
591 .ad
592 .sp .6
593 .RS 4n
594 Set the label geometry to the content of the \fIgeom_file\fR. The
595 \fIgeom_file\fR contains one specification line. Each line is delimited by a
596 new-line character (\fB\en\fR). If the first character of a line is an asterisk
596 new-line character (\fB\n\fR). If the first character of a line is an asterisk
597 (*), the line is treated as a comment. Each line is composed of entries that
598 are position-dependent, are separated by white space, and have the following
599 format:
600 .sp
601 .in +2
602 .nf
603 \fIpcyl ncyl acyl bcyl nheads nsectors sectsiz\fR
604 .fi
605 .in -2
606 .sp

608 where the entries have the following values:
609 .sp
610 .ne 2
611 .na
612 \fB\fIpcyl\fR\fR
613 .ad
614 .RS 12n
615 This is the number of physical cylinders for the drive.
616 .RE

618 .sp
619 .ne 2
620 .na
621 \fB\fIncyl\fR\fR
622 .ad
623 .RS 12n
624 This is the number of usable cylinders for the drive.
625 .RE

627 .sp
628 .ne 2
629 .na
630 \fB\fIacyl\fR\fR
631 .ad
632 .RS 12n
633 This is the number of alt cylinders for the drive.
634 .RE

636 .sp
637 .ne 2
638 .na
639 \fB\fIbcyl\fR\fR
640 .ad
641 .RS 12n
642 This is the number of offset cylinders for the drive (should be zero).
643 .RE

645 .sp
646 .ne 2
647 .na
648 \fB\fInheads\fR\fR
649 .ad
650 .RS 12n
651 The number of heads for this drive.
652 .RE
```

```
654 .sp
655 .ne 2
656 .na
657 \fB\fInsectors\fR\fR
658 .ad
659 .RS 12n
660 The number of sectors per track.
661 .RE

663 .sp
664 .ne 2
665 .na
666 \fB\fIsectsiz\fR\fR
667 .ad
668 .RS 12n
669 The size in bytes of a sector.
670 .RE

672 .RE

674 .sp
675 .ne 2
676 .na
677 \fB\fB-t\fR\fR
678 .ad
679 .sp .6
680 .RS 4n
681 Adjust incorrect slice table entries so that they will not cross partition
682 table boundaries.
683 .RE

685 .sp
686 .ne 2
687 .na
688 \fB\fB-T\fR\fR
689 .ad
690 .sp .6
691 .RS 4n
692 Remove incorrect slice table entries that span partition table boundaries.
693 .RE

695 .sp
696 .ne 2
697 .na
698 \fB\fB-v\fR\fR
699 .ad
700 .sp .6
701 .RS 4n
702 Output the HBA (virtual) geometry dimensions. This option must be used in
703 conjunction with the \fB-W\fR flag. This option will work for platforms which
704 support virtual geometry. (x86 only)
705 .RE

707 .sp
708 .ne 2
709 .na
710 \fB\fB-w\fR\fR
711 .ad
712 .sp .6
713 .RS 4n
714 Write to disk and read from stdin. See \fB-o\fR and \fB-s\fR, which specify the
715 starting point and size of the operation.
716 .RE

718 .sp
719 .ne 2
```



```
720 .na
721 \fB\fB-W\fR \fB\(mi\fR\fR
722 .ad
723 .sp .6
724 .RS 4n
725 Output the disk table to \fBstdout\fR.
726 .RE

728 .sp
729 .ne 2
730 .na
731 \fB\fB-W\fR \fIfdisk_file\fR\fR
732 .ad
733 .sp .6
734 .RS 4n
735 Create an \fBfdisk\fR file \fIfdisk_file\fR from disk table. This can be used
736 with the \fB-F\fR option below.
737 .RE

739 .SH FILES
740 .sp
741 .ne 2
742 .na
743 \fB\fB/dev/rdsk/c0t0d0p0\fR\fR
744 .ad
745 .RS 25n
746 Raw device associated with the fixed disk.
747 .RE

749 .sp
750 .ne 2
751 .na
752 \fB\fB/usr/lib/fs/ufs/mboot\fR\fR
753 .ad
754 .RS 25n
755 Default master boot program.
756 .RE

758 .SH ATTRIBUTES
759 .sp
760 .LP
761 See \fBattributes\fR(5) for descriptions of the following attributes:
762 .sp

764 .sp
765 .TS
766 box;
767 c | c
768 l | l .
769 ATTRIBUTE TYPE  ATTRIBUTE VALUE
770 _
771 Architecture    x86 and SPARC
772 .TE

774 .SH SEE ALSO
775 .sp
776 .LP
777 \fBuname\fR(1), \fBfmthard\fR(1M), \fBformat\fR(1M), \fBnewfs\fR(1M),
778 \fBprtvtoc\fR(1M), \fBattributes\fR(5)
779 .SH DIAGNOSTICS
780 .sp
781 .LP
782 Most messages will be self-explanatory. The following may appear immediately
783 after starting the program:
784 .sp
785 .ne 2
```

```
.na
\fB\fBFdisk\fR: \fBcannot open\fR <\fBdevice\fR>\fR
.ad
.sp .6
.RS 4n
This indicates that the device name argument is not valid.
.RE

.sp
.ne 2
.na
\fB\fBFdisk\fR: \fBunable to get device parameters for device\fR
<\fBdevice\fR>\fR
.ad
.sp .6
.RS 4n
This indicates a problem with the configuration of the fixed disk, or an error
in the fixed disk driver.
.RE

.sp
.ne 2
.na
\fB\fBFdisk\fR: \fBerror reading partition table\fR\fR
.ad
.sp .6
.RS 4n
This indicates that some error occurred when trying initially to read the fixed
disk. This could be a problem with the fixed disk controller or driver, or with
the configuration of the fixed disk.
.RE

.sp
.ne 2
.na
\fB\fBFdisk\fR: \fBerror writing boot record\fR\fR
.ad
.sp .6
.RS 4n
This indicates that some error occurred when trying to write the new partition
table out to the fixed disk. This could be a problem with the fixed disk
controller, the disk itself, the driver, or the configuration of the fixed
disk.
.RE
```

```
**********************************************************
    9251 Tue Sep 10 18:35:20 2013
new/usr/src/man/man1m/id.1m
4023 - Typo in file(1) manpage and various others
**********************************************************
'\" te
.\"  Copyright (c) 1992, X/Open Company Limited  All Rights Reserved  Portions C
.\" Sun Microsystems, Inc. gratefully acknowledges The Open Group for permission
.\" http://www.opengroup.org/bookstore/.
.\" The Institute of Electrical and Electronics Engineers and The Open Group, ha
.\"  This notice shall appear on any product containing this material.
.\" 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 ID 1M "Sep 10, 2013"
.TH ID 1M "Nov 28, 2006"
.SH NAME
id \- return user identity
.SH SYNOPSIS
.LP
.nf
\fB/usr/bin/id\fR [\fB-p\fR] [\fIuser\fR]
.fi

.LP
.nf
\fB/usr/bin/id\fR \fB-a\fR [\fB-p\fR] [\fIuser\fR]
.fi

.LP
.nf
\fB/usr/bin/id\fR \fB-G\fR [\fB-n\fR] [\fIuser\fR]
.fi

.LP
.nf
\fB/usr/bin/id\fR \fB-g\fR [\fB-nr\fR] [\fIuser\fR]
.fi

.LP
.nf
\fB/usr/bin/id\fR \fB-u\fR [\fB-nr\fR] [\fIuser\fR]
.fi

.LP
.nf
\fB/usr/xpg4/bin/id\fR [\fB-p\fR] [\fIuser\fR]
.fi

.LP
.nf
\fB/usr/xpg4/bin/id\fR \fB-a\fR [\fB-p\fR] [\fIuser\fR]
.fi

.LP
.nf
\fB/usr/xpg4/bin/id\fR \fB-G\fR [\fB-n\fR] [\fIuser\fR]
.fi

.LP
.nf
\fB/usr/xpg4/bin/id\fR \fB-g\fR [\fB-nr\fR] [\fIuser\fR]
.fi

.LP
.nf
```



```
\fB/usr/xpg4/bin/id\fR \fB-u\fR [\fB-nr\fR] [\fIuser\fR]
.fi

.SH DESCRIPTION
.sp
.LP
If no \fIuser\fR operand is provided, the \fBid\fR utility writes the user and
group \fBID\fRs and the corresponding user and group names of the invoking
process to standard output. If the effective and real \fBID\fRs do not match,
both are written. If multiple groups are supported by the underlying system,
\fB/usr/xpg4/bin/id\fR also writes the supplementary group affiliations of the
invoking process.
.sp
.LP
If a \fIuser\fR operand is provided and the process has the appropriate
privileges, the user and group \fBID\fRs of the selected user are written. In
this case, effective \fBID\fRs are assumed to be identical to real \fBID\fRs.
If the selected user has more than one allowable group membership listed in the
group database, \fB/usr/xpg4/bin/id\fR writes them in the same manner as the
supplementary groups described in the preceding paragraph.
.SS "Formats"
.sp
.LP
The following formats are used when the \fBLC_MESSAGES\fR locale category
specifies the "C" locale. In other locales, the strings \fBuid\fR, \fBgid\fR,
\fBeuid\fR, \fBegid\fR, and \fBgroups\fR may be replaced with more appropriate
strings corresponding to the locale.
.sp
.in +2
.nf
"uid=%u(%s) gid=%u(%s)\en" <\fIreal user ID\fR>, <\fIuser-name\fR>,
    <\fIreal group ID\fR>, <\fIgroup-name\fR>
.fi
.in -2
.sp

.sp
.LP
If the effective and real user \fBID\fRs do not match, the following are
inserted immediately before the \fB\en\fR character in the previous format:
.sp
.in +2
.nf
" euid=%u(%s)"
.fi
.in -2
.sp

.sp
.LP
with the following arguments added at the end of the argument list:
.sp
.in +2
.nf
<\fIeffective user ID\fR>, <\fIeffective user-name\fR>
.fi
.in -2
.sp

.sp
.LP
If the effective and real group \fBID\fRs do not match, the following is
inserted directly before the \fB\en\fR character in the format string (and
after any addition resulting from the effective and real user \fBID\fRs not
matching):
.sp
```

```
.in +2
.nf
" egid=%u(%s)"
.fi
.in -2
.sp

.sp
.LP
with the following arguments added at the end of the argument list:
.sp
.in +2
.nf
<\fIeffectivegroup-ID\fR>, <\fIeffectivegroupname\fR>
.fi
.in -2
.sp

.sp
.LP
If the process has supplementary group affiliations or the selected user is
allowed to belong to multiple groups, the first is added directly before the
\fBNEWLINE\fR character in the format string:
.sp
.in +2
.nf
" groups=%u(%s)"
.fi
.in -2
.sp

.sp
.LP
with the following arguments added at the end of the argument list:
.sp
.in +2
.nf
<\fIsupplementary group ID\fR>, <\fIsupplementary group name\fR>
.fi
.in -2
.sp

.sp
.LP
and the necessary number of the following added after that for any remaining
supplementary group \fBID\fRs:
.sp
.in +2
.nf
",%u(%s)"
.fi
.in -2
.sp

.sp
.LP
and the necessary number of the following arguments added at the end of the
argument list:
.sp
.in +2
.nf
<\fIsupplementary group ID\fR>, <\fIsupplementary group name\fR>
.fi
.in -2
.sp
```



```
.sp
.LP
If any of the user \fBID\fR, group \fBID\fR, effective user \fBID\fR, effective
group \fBID\fR or supplementary/multiple group \fBID\fRs cannot be mapped by
the system into printable user or group names, the corresponding (\fB%s\fR) and
name argument is omitted from the corresponding format string.
.sp
.LP
When any of the options are specified, the output format is as described under
OPTIONS.
.SH OPTIONS
.sp
.LP
The following options are supported by both \fB/usr/bin/id\fR and
\fB/usr/xpg4/bin/id\fR. The \fB-p\fR and \fB-a\fR options are invalid if
specified with any of the \fB-G\fR, \fB-g\fR, or \fB-u\fR options.
.sp
.ne 2
.na
\fB\fB-p\fR\fR
.ad
.RS 6n
Reports additionally the current project membership of the invoking process.
The project is reported using the           format:
.sp
.in +2
.nf
"projid=%u(%s)"
.fi
.in -2
.sp

which is inserted prior to the \fB\en\fR character of the default format
which is inserted prior to the \fB\n\fR character of the default format
described in the \fBFormats\fR section. The arguments
.sp
.in +2
.nf
<\fIproject ID\fR>,<\fIproject name\fR>
.fi
.in -2
.sp

are appended to the end of the argument list.  If the project \fBID\fR cannot
be mapped by the system into a printable project name, the corresponding
\fB(%s)\fR and name argument is omitted from the corresponding format string.
.RE

.sp
.ne 2
.na
\fB\fB-a\fR\fR
.ad
.RS 6n
Reports user name, user  \fBID\fR and all the groups to which the user belongs.
.RE

.sp
.ne 2
.na
\fB\fB-G\fR\fR
.ad
.RS 6n
Outputs all different group \fBID\fRs (effective, real and supplementary) only,
using the format \fB"%u\en"\fR. If there is more than one distinct group
affiliation, output each such affiliation, using the format \fB" %u"\fR, before
```

```
the \fBNEWLINE\fR character is output.
.RE

.sp
.ne 2
.na
\fB\fB-g\fR\fR
.ad
.RS 6n
Outputs only the effective group \fBID\fR, using the format \fB"%u\en"\fR.
.RE

.sp
.ne 2
.na
\fB\fB-n\fR\fR
.ad
.RS 6n
Outputs the name in the format \fB"%s"\fR instead of the numeric \fBID\fR using
the format \fB"%u"\fR.
.RE

.sp
.ne 2
.na
\fB\fB-r\fR\fR
.ad
.RS 6n
Outputs the real \fBID\fR instead of the effective \fBID\fR.
.RE

.sp
.ne 2
.na
\fB\fB-u\fR\fR
.ad
.RS 6n
Outputs only the effective user \fBID\fR, using the format \fB"%u\en"\fR.
.RE

.SH OPERANDS
.sp
.LP
The following operand is supported:
.sp
.ne 2
.na
\fB\fIuser\fR\fR
.ad
.RS 8n
The user (login) name for which information is to be written.
.RE

.SH ENVIRONMENT VARIABLES
.sp
.LP
See \fBenviron\fR(5) for descriptions of the following environment variables
that affect the execution of \fBid\fR: \fBLANG\fR, \fBLC_ALL\fR,
\fBLC_CTYPE\fR, \fBLC_MESSAGES\fR, and \fBNLSPATH\fR.
.SH EXIT STATUS
.sp
.LP
The following exit values are returned:
.sp
.ne 2
.na
```



```
\fB\fB0\fR\fR
.ad
.RS 6n
Successful completion.
.RE

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

.SH ATTRIBUTES
.sp
.LP
See \fBattributes\fR(5) for descriptions of the following attributes:
.SS "/usr/bin/id"
.sp

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

.SS "/usr/xpg4/bin/id"
.sp

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

.SH SEE ALSO
.sp
.LP
\fBfold\fR(1), \fBlogname\fR(1), \fBwho\fR(1), \fBgetgid\fR(2),
\fBgetgroups\fR(2), \fBgetprojid\fR(2), \fBgetuid\fR(2), \fBattributes\fR(5),
\fBenviron\fR(5), \fBstandards\fR(5)
.SH NOTES
.sp
.LP
Output produced by the \fB-G\fR option and by the default case could
potentially produce very long lines on systems that support large numbers of
supplementary groups.
```

```
**********************************************************
   15223 Tue Sep 10 18:35:20 2013
new/usr/src/man/man3cpc/cpc_bind_event.3cpc
4023 - Typo in file(1) manpage and various others
**********************************************************
```

```
   1 '\" te
   2 .\" Copyright (c) 2007, Sun Microsystems, Inc. All Rights Reserved.
   3 .\" The contents of this file are subject to the terms of the Common Development
   4 .\" You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE or http:
   5 .\" When distributing Covered Code, include this CDDL HEADER in each file and in
   6 .TH CPC_BIND_EVENT 3CPC "Sep 10, 2013"
   6 .TH CPC_BIND_EVENT 3CPC "Mar 02, 2007"
   7 .SH NAME
   8 cpc_bind_event, cpc_take_sample, cpc_rele \- use CPU performance counters on
   9 lwps
  10 .SH SYNOPSIS
  11 .LP
  12 .nf
  13 cc [ \fIflag\fR... ] \fIfile\fR... \(milcpc [ \fIlibrary\fR... ]
  14 #include <libcpc.h>

  16 \fBint\fR \fBcpc_bind_event\fR(\fBcpc_event_t *\fR\fIevent\fR, \fBint\fR \fIflag
  17 .fi

  19 .LP
  20 .nf
  21 \fBint\fR \fBcpc_take_sample\fR(\fBcpc_event_t *\fR\fIevent\fR);
  22 .fi

  24 .LP
  25 .nf
  26 \fBint\fR \fBcpc_rele\fR(\fBvoid\fR);
  27 .fi

  29 .SH DESCRIPTION
  30 .sp
  31 .LP
  32 Once the events to be sampled have been selected using, for example,
  33 \fBcpc_strtoevent\fR(3CPC), the event selections can be bound to the calling
  34 \fBLWP\fR using \fBcpc_bind_event()\fR. If \fBcpc_bind_event()\fR returns
  35 successfully, the system has associated performance counter context with the
  36 calling \fBLWP\fR. The context allows the system to virtualize the hardware
  37 counters to that specific \fBLWP\fR, and the counters are enabled.
  38 .sp
  39 .LP
  40 Two flags are defined that can be passed into the routine to allow the behavior
  41 of the interface to be modified, as described below.
  42 .sp
  43 .LP
  44 Counter values can be sampled at any time by calling \fBcpc_take_sample()\fR,
  45 and dereferencing the fields of the \fBce_pic\fR\fB[]\fR array returned. The
  46 \fBce_hrt\fR field contains the timestamp at which the kernel last sampled the
  47 counters.
  48 .sp
  49 .LP
  50 To immediately remove the performance counter context on an \fBLWP\fR, the
  51 \fBcpc_rele()\fR interface should be used. Otherwise, the context will be
  52 destroyed after the \fBLWP\fR or process exits.
  53 .sp
  54 .LP
  55 The caller should take steps to ensure that the counters are sampled often
  56 enough to avoid the 32-bit counters wrapping. The events most prone to wrap are
  57 those that count processor clock cycles. If such an event is of interest,
  58 sampling should occur frequently so that less than 4 billion clock cycles can
  59 occur between samples. Practically speaking, this is only likely to be a
  60 problem for otherwise idle systems, or when processes are bound to processors,
```



```
  61 since normal context switching behavior will otherwise hide this problem.
  62 .SH RETURN VALUES
  63 .sp
  64 .LP
  65 Upon successful completion, \fBcpc_bind_event()\fR and \fBcpc_take_sample()\fR
  66 return \fB0\fR. Otherwise, these functions return \fB\(mi1\fR, and set
  67 \fBerrno\fR to indicate the error.
  68 .SH ERRORS
  69 .sp
  70 .LP
  71 The \fBcpc_bind_event()\fR and \fBcpc_take_sample()\fR functions will fail if:
  72 .sp
  73 .ne 2
  74 .na
  75 \fB\fBEACCES\fR\fR
  76 .ad
  77 .RS 11n
  78 For \fBcpc_bind_event()\fR, access to the requested hypervisor event was
  79 denied.
  80 .RE

  82 .sp
  83 .ne 2
  84 .na
  85 \fB\fBEAGAIN\fR\fR
  86 .ad
  87 .RS 11n
  88 Another process may be sampling system-wide CPU statistics. For
  89 \fBcpc_bind_event()\fR, this implies that no new contexts can be created. For
  90 \fBcpc_take_sample()\fR, this implies that the performance counter context has
  91 been invalidated and must be released with \fBcpc_rele()\fR. Robust programs
  92 should be coded to expect this behavior and recover from it by releasing the
  93 now invalid context by calling \fBcpc_rele()\fR sleeping for a while, then
  94 attempting to bind and sample the event once more.
  95 .RE

  97 .sp
  98 .ne 2
  99 .na
 100 \fB\fBEINVAL\fR\fR
 101 .ad
 102 .RS 11n
 103 The \fBcpc_take_sample()\fR function has been invoked before the context is
 104 bound.
 105 .RE

 107 .sp
 108 .ne 2
 109 .na
 110 \fB\fBENOTSUP\fR\fR
 111 .ad
 112 .RS 11n
 113 The caller has attempted an operation that is illegal or not supported on the
 114 current platform, such as attempting to specify signal delivery on counter
 115 overflow on a CPU that doesn't generate an interrupt on counter overflow.
 116 .RE

 118 .SH USAGE
 119 .sp
 120 .LP
 121 Prior to calling \fBcpc_bind_event()\fR, applications should call
 122 \fBcpc_access\fR(3CPC) to determine if the counters are accessible on the
 123 system.
 124 .SH EXAMPLES
 125 .LP
 126 \fBExample 1 \fRUse hardware performance counters to measure events in a
```

```
process.
.sp
.LP
The example below shows how a standalone program can be instrumented with the
\fBlibcpc\fR routines to use hardware performance counters to measure events in
a process.  The program performs 20 iterations of a computation, measuring the
counter values for each iteration.  By default, the example makes the counters
measure external cache references and external cache hits; these options are
only appropriate for UltraSPARC processors. By setting the \fBPERFEVENTS\fR
environment variable to other strings (a list of which can be gleaned from the
\fB-h\fR flag of the \fBcpustat\fR or \fBcputrack\fR utilities), other events
can be counted.  The \fBerror()\fR routine below is assumed to be a
user-provided routine analogous to the familiar \fBprintf\fR(3C) routine from
the C library but which also performs an \fBexit\fR(2) after printing the
message.

.sp
.in +2
.nf
\fB#include <inttypes.h>
#include <stdlib.h>
#include <stdio.h>
#include <unistd.h>
#include <libcpc.h>
int
main(int argc, char *argv[])
{
int cpuver, iter;
char *setting = NULL;
cpc_event_t event;

if (cpc_version(CPC_VER_CURRENT) != CPC_VER_CURRENT)
    error("application:library cpc version mismatch!");

if ((cpuver = cpc_getcpuver()) == -1)
    error("no performance counter hardware!");

if ((setting = getenv("PERFEVENTS")) == NULL)
    setting = "pic0=EC_ref,pic1=EC_hit";

if (cpc_strtoevent(cpuver, setting, &event) != 0)
    error("can't measure '%s' on this processor", setting);
setting = cpc_eventtostr(&event);

if (cpc_access() == -1)
    error("can't access perf counters: %s", strerror(errno));

if (cpc_bind_event(&event, 0) == -1)
    error("can't bind lwp%d: %s", _lwp_self(), strerror(errno));

for (iter = 1; iter <= 20; iter++) {
    cpc_event_t before, after;

    if (cpc_take_sample(&before) == -1)
        break;

    /* ==> Computation to be measured goes here <== */

    if (cpc_take_sample(&after) == -1)
        break;
    (void) printf("%3d: %" PRId64 " %" PRId64 "\en", iter,
    (void) printf("%3d: %" PRId64 " %" PRId64 "\n", iter,
        after.ce_pic[0] - before.ce_pic[0],
        after.ce_pic[1] - before.ce_pic[1]);
}
```

unchanged_portion_omitted_


```
    (void) printf("lwp%d - si_addr %p ucontext: %%pc %p %%sp %p\en",
    (void) printf("lwp%d - si_addr %p ucontext: %%pc %p %%sp %p\n",
        _lwp_self(), (void *)sip->si_addr,
        (void *)uap->uc_mcontext.gregs[PC],
        (void *)uap->uc_mcontext.gregs[USP]);

if (cpc_take_sample(&sample) == -1)
    error("can't sample: %s", strerror(errno));

    (void) printf("0x%" PRIx64 " 0x%" PRIx64 "\en",
    (void) printf("0x%" PRIx64 " 0x%" PRIx64 "\n",
    sample.ce_pic[0], sample.ce_pic[1]);
(void) fflush(stdout);

sample.ce_pic[0] = PRESET0;
sample.ce_pic[1] = PRESET1;
if (cpc_bind_event(&sample, CPC_BIND_EMT_OVF) == -1)
    error("cannot bind lwp%d: %s", _lwp_self(), strerror(errno));
}
```

unchanged_portion_omitted_

```
**********************************************************
   16398 Tue Sep 10 18:35:20 2013
new/usr/src/man/man3nsl/gethostbyname.3nsl
4023 - Typo in file(1) manpage and various others
**********************************************************
```

```
'\" te
.\" Copyright (C) 2008, Sun Microsystems, Inc. All Rights Reserved.
.\" Copyright 1989 AT&T.
.\" Portions Copyright (c) 1992, X/Open Company Limited. All Rights Reserved
.\" Sun Microsystems, Inc. gratefully acknowledges The Open Group for permission
.\" http://www.opengroup.org/bookstore/.
.\" The Institute of Electrical and Electronics Engineers and The Open Group, ha
.\"  This notice shall appear on any product containing this material.
.\" 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 GETHOSTBYNAME 3NSL "Sep 10, 2013"
.TH GETHOSTBYNAME 3NSL "Aug 24, 2007"
.SH NAME
gethostbyname, gethostbyname_r, gethostbyaddr, gethostbyaddr_r, gethostent,
gethostent_r, sethostent, endhostent \- get network host entry
.SH SYNOPSIS
.LP
.nf
\fBcc\fR [ \fIflag\fR... ] \fIfile\fR... \fB-lnsl\fR [ \fIlibrary\fR... ]
#include <netdb.h>

\fBstruct hostent *\fR\fBgethostbyname\fR(\fBconst char *\fR\fIname\fR);
.fi

.LP
.nf
\fBstruct hostent *\fR\fBgethostbyname_r\fR(\fBconst char *\fR\fIname\fR,
     \fBstruct hostent *\fR\fIresult\fR, \fBchar *\fR\fIbuffer\fR, \fBint\fR \fI
     \fBint *\fR\fIh_errnop\fR);
.fi

.LP
.nf
\fBstruct hostent *\fR\fBgethostbyaddr\fR(\fBconst char *\fR\fIaddr\fR, \fBint\f
     \fBint\fR \fItype\fR);
.fi

.LP
.nf
\fBstruct hostent *\fR\fBgethostbyaddr_r\fR(\fBconst char *\fR\fIaddr\fR, \fBint
     \fBint\fR \fItype\fR, \fBstruct hostent *\fR\fIresult\fR, \fBchar *\fR\fIbu
     \fBint\fR \fIbuflen\fR, \fBint *\fR\fIh_errnop\fR);
.fi

.LP
.nf
\fBstruct hostent *\fR\fBgethostent\fR(\fBvoid\fR);
.fi

.LP
.nf
\fBstruct hostent *\fR\fBgethostent_r\fR(\fBstruct hostent *\fR\fIresult\fR,
     \fBchar *\fR\fIbuffer\fR, \fBint\fR \fIbuflen\fR, \fBint *\fR\fIh_errnop\fR
.fi

.LP
.nf
\fBint\fR \fBsethostent\fR(\fBint\fR \fIstayopen\fR);
.fi
```



```
.LP
.nf
\fBint\fR \fBendhostent\fR(\fBvoid\fR);
.fi

.SH DESCRIPTION
.sp
.LP
These functions are used to obtain entries describing hosts. An entry can come
from any of the sources for \fBhosts\fR specified in the
\fB/etc/nsswitch.conf\fR file. See \fBnsswitch.conf\fR(4). These functions have
been superseded by \fBgetipnodebyname\fR(3SOCKET),
\fBgetipnodebyaddr\fR(3SOCKET), and \fBgetaddrinfo\fR(3SOCKET), which provide
greater portability to applications when multithreading is performed or
technologies such as IPv6 are used. For example, the functions described in the
following cannot be used with applications targeted to work with IPv6.
.sp
.LP
The \fBgethostbyname()\fR function searches for information for a host with the
hostname specified by the character-string parameter \fIname\fR.
.sp
.LP
The \fBgethostbyaddr()\fR function searches for information for a host with a
given host address. The parameter \fBtype\fR specifies the family of the
address. This should be one of the address families defined in
\fB<sys/socket.h>\fR\&. See the \fBNOTES\fR section for more information. Also
see the \fBEXAMPLES\fR section for information on how to convert an Internet
\fBIP\fR address notation that is separated by periods (.) into an \fIaddr\fR
parameter. The parameter \fIlen\fR specifies the length of the buffer indicated
by \fIaddr\fR.
.sp
.LP
All addresses are returned in network order. In order to interpret the
addresses, \fBbyteorder\fR(3SOCKET) must be used for byte order conversion.
.sp
.LP
The \fBsethostent()\fR, \fBgethostent()\fR, and \fBendhostent()\fR functions
are used to enumerate host entries from the database.
.sp
.LP
The \fBsethostent()\fR function sets or resets the enumeration to the beginning
of the set of host entries. This function should be called before the first
call to \fBgethostent()\fR. Calls to \fBgethostbyname()\fR and
\fBgethostbyaddr()\fR leave the enumeration position in an indeterminate state.
If the \fIstayopen\fR flag is non-zero, the system can keep allocated resources
such as open file descriptors until a subsequent call to \fBendhostent()\fR.
.sp
.LP
Successive calls to the \fBgethostent()\fR function return either successive
entries or \fINULL,\fR indicating the end of the enumeration.
.sp
.LP
The \fBendhostent()\fR function can be called to indicate that the caller
expects to do no further host entry retrieval operations; the system can then
deallocate resources it was using. It is still allowed, but possibly less
efficient, for the process to call more host retrieval functions after calling
\fBendhostent()\fR.
.SS "Reentrant Interfaces"
.sp
.LP
The \fBgethostbyname()\fR, \fBgethostbyaddr()\fR, and \fBgethostent()\fR
functions use static storage that is reused in each call, making these
functions unsafe for use in multithreaded applications.
.sp
.LP
The \fBgethostbyname_r()\fR, \fBgethostbyaddr_r()\fR, and \fBgethostent_r()\fR
```

functions provide reentrant interfaces for these operations.
.sp
.LP
Each reentrant interface performs the same operation as its non-reentrant counterpart, named by removing the \fB_r\fR suffix. The reentrant interfaces, however, use buffers supplied by the caller to store returned results and the interfaces are safe for use in both single-threaded and multithreaded applications.
.sp
.LP
Each reentrant interface takes the same parameters as its non-reentrant counterpart, as well as the following additional parameters. The parameter \fIresult\fR must be a pointer to a \fBstruct hostent\fR structure allocated by the caller. On successful completion, the function returns the host entry in this structure. The parameter \fIbuffer\fR must be a pointer to a buffer supplied by the caller. This buffer is used as storage space for the host data. All of the pointers within the returned \fBstruct hostent\fR \fIresult\fR point to data stored within this buffer. See the \fBRETURN VALUES\fR section for more information. The buffer must be large enough to hold all of the data associated with the host entry. The parameter \fIbuflen\fR should give the size in bytes of the buffer indicated by \fIbuffer\fR. The parameter \fIh_errnop\fR should be a pointer to an integer. An integer error status value is stored there on certain error conditions. See the \fBERRORS\fR section for more information.
.sp
.LP
For enumeration in multithreaded applications, the position within the enumeration is a process-wide property shared by all threads. The \fBsethostent()\fR function can be used in a multithreaded application but resets the enumeration position for all threads. If multiple threads interleave calls to \fBgethostent_r()\fR, the threads will enumerate disjoint subsets of the host database.
.sp
.LP
Like their non-reentrant counterparts, \fBgethostbyname_r()\fR and \fBgethostbyaddr_r()\fR leave the enumeration position in an indeterminate state.
.SH RETURN VALUES
.sp
.LP
Host entries are represented by the \fBstruct hostent\fR structure defined in \fB<netdb.h>\fR:
.sp
.in +2
.nf
struct hostent {
    char   *h_name;   /* canonical name of host */
    char   **h_aliases;   /* alias list */
    int   h_addrtype;   /* host address type */
    int   h_length;   /* length of address */
    char   **h_addr_list;   /* list of addresses */
};
.fi
.in -2

.sp
.LP
See the \fBEXAMPLES\fR section for information about how to retrieve a ‘‘.’’ separated Internet \fBIP\fR address string from the \fIh_addr_list\fR field of \fBstruct hostent\fR.
.sp
.LP
The \fBgethostbyname()\fR, \fBgethostbyname_r()\fR, \fBgethostbyaddr()\fR, and \fBgethostbyaddr_r()\fR functions each return a pointer to a \fBstruct hostent\fR if they successfully locate the requested entry; otherwise they return \fINULL\fR.
.sp

.LP
The \fBgethostent()\fR and \fBgethostent_r()\fR functions each return a pointer to a \fBstruct hostent\fR if they successfully enumerate an entry; otherwise they return \fINULL\fR, indicating the end of the enumeration.
.sp
.LP
The \fBgethostbyname()\fR, \fBgethostbyaddr()\fR, and \fBgethostent()\fR functions use static storage, so returned data must be copied before a subsequent call to any of these functions if the data is to be saved.
.sp
.LP
When the pointer returned by the reentrant functions \fBgethostbyname_r()\fR, \fBgethostbyaddr_r()\fR, and \fBgethostent_r()\fR is not \fINULL\fR, it is always equal to the \fIresult\fR pointer that was supplied by the caller.
.sp
.LP
The \fBsethostent()\fR and \fBendhostent()\fR functions return \fB0\fR on success.
.SH ERRORS
.sp
.LP
The reentrant functions \fBgethostbyname_r()\fR, \fBgethostbyaddr_r()\fR, and \fBgethostent_r()\fR will return \fINULL\fR and set \fIerrno\fR to \fBERANGE\fR if the length of the buffer supplied by caller is not large enough to store the result. See \fBIntro\fR(2) for the proper usage and interpretation of \fBerrno\fR in multithreaded applications.
.sp
.LP
The reentrant functions \fBgethostbyname_r()\fR and \fBgethostbyaddr_r()\fR set the integer pointed to by \fIh_errnop\fR to one of these values in case of error.
.sp
.LP
On failures, the non-reentrant functions \fBgethostbyname()\fR and \fBgethostbyaddr()\fR set a global integer \fIh_errno\fR to indicate one of these error codes (defined in \fB<netdb.h>\fR): \fBHOST_NOT_FOUND\fR, \fBTRY_AGAIN\fR, \fBNO_RECOVERY\fR, \fBNO_DATA\fR, and \fBNO_ADDRESS\fR.
.sp
.LP
If a resolver is provided with a malformed address, or if any other error occurs before \fBgethostbyname()\fR is resolved, then \fBgethostbyname()\fR returns an internal error with a value of \(mi1.
.sp
.LP
The \fBgethostbyname()\fR function will set \fIh_errno\fR to \fBNETDB_INTERNAL\fR when it returns a \fINULL\fR value.
.SH EXAMPLES
.LP
\fBExample 1 \fRUsing \fBgethostbyaddr()\fR
.sp
.LP
Here is a sample program that gets the canonical name, aliases, and ‘‘.’’ separated Internet \fBIP\fR addresses for a given ‘‘.’’ separated \fBIP\fR address:

.sp
.in +2
.nf
#include <stdio.h>
#include <stdlib.h
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <netdb.h>

```
259 int main(int argc, const char **argv)
260 {
261         in_addr_t addr;
262         struct hostent *hp;
263         char **p;
264         if (argc != 2) {
265             (void) printf("usage: %s IP-address\en", argv[0]);
266             exit (1);
267         }
268         if ((int)(addr = inet_addr(argv[1])) == -1) {
269             (void) printf("IP-address must be of the form a.b.c.d\en");
270             exit (2);
271         }
272         hp = gethostbyaddr((char *)&addr, 4, AF_INET);
273         if (hp == NULL) {
274             (void) printf("host information for %s not found\en", argv[1]);
275             exit (3);
276         }
277         for (p = hp->h_addr_list; *p != 0; p++) {
278             struct in_addr in;
279             char **q;
280             (void) memcpy(&in.s_addr, *p, sizeof (in.s_addr));
281          (void) printf("%s\t%s", inet_ntoa(in), hp\(mi>h_name);
282             for (q = hp->h_aliases; *q != 0; q++)
283                 (void) printf(" %s", *q);
284             (void) putchar('\en');
284             (void) putchar('\n');
285         }
286         exit (0);
287 }
```

_____***unchanged_portion_omitted_***

```
**********************************************************
   19326 Tue Sep 10 18:35:20 2013
new/usr/src/man/man3papi/papiJobSubmit.3papi
4023 - Typo in file(1) manpage and various others
**********************************************************
```

```
   1 '\" te
   2 .\" Copyright (c) 2007, Sun Microsystems, Inc.  All Rights Reserved.
   3 .\" The contents of this file are subject to the terms of the Common Development
   4 .\" You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE or http:
   5 .\" When distributing Covered Code, include this CDDL HEADER in each file and in
   6 .TH PAPIJOBSUBMIT 3PAPI "Sep 10, 2013"
   6 .TH PAPIJOBSUBMIT 3PAPI "Jan 17, 2007"
   7 .SH NAME
   8 papiJobSubmit, papiJobSubmitByReference, papiJobValidate, papiJobStreamOpen,
   9 papiJobStreamWrite, papiJobStreamClose, papiJobQuery, papiJobModify,
  10 papiJobMove, papiJobCancel, papiJobHold, papiJobRelease, papiJobRestart,
  11 papiJobPromote, papiJobGetAttributeList, papiJobGetPrinterName, papiJobGetId,
  12 papiJobGetJobTicket, papiJobFree, papiJobListFree \- job object manipulation
  13 .SH SYNOPSIS
  14 .LP
  15 .nf
  16 cc [ \fIflag\fR\&.\|.\|. ] \fIfile\fR\&.\|.\|. \fB-lpapi\fR [ \fIlibrary\fR\&.\|
  17 #include <papi.h>

  19 \fBpapi_status_t\fR \fBpapiJobSubmit\fR(\fBpapi_service_t\fR \fIhandle\fR,
  20      \fBchar *\fR\fIprinter\fR, \fBpapi_attribute_t **\fR\fIjob_attributes\fR,
  21      \fBpapi_job_ticket_t *\fR\fIjob_ticket\fR, \fBchar **\fR\fIfiles\fR,
  22      \fBpapi_job_t *\fR\fIjob\fR);
  23 .fi

  25 .LP
  26 .nf
  27 \fBpapi_status_t\fR \fBpapiJobSubmitByReference\fR(\fBpapi_service_t\fR \fIhandl
  28      \fBchar *\fR\fIprinter\fR, \fBpapi_attribute_t **\fR\fIjob_attributes\fR,
  29      \fBpapi_job_ticket_t *\fR\fIjob_ticket\fR, \fBchar **\fR\fIfiles\fR,
  30      \fBpapi_job_t *\fR\fIjob\fR);
  31 .fi

  33 .LP
  34 .nf
  35 \fBpapi_status_t\fR \fBpapiJobValidate\fR(\fBpapi_service_t\fR \fIhandle\fR,
  36      \fBchar *\fR\fIprinter\fR, \fBpapi_attribute_t **\fR\fIjob_attributes\fR,
  37      \fBpapi_job_ticket_t *\fR\fIjob_ticket\fR, \fBchar **\fR\fIfiles\fR,
  38      \fBpapi_job_t *\fR\fIjob\fR);
  39 .fi

  41 .LP
  42 .nf
  43 \fBpapi_status_t\fR \fBpapiJobStreamOpen\fR(\fBpapi_service_t\fR \fIhandle\fR,
  44      \fBchar *\fR\fIprinter\fR, \fBpapi_attribute_t **\fR\fIjob_attributes\fR,
  45      \fBpapi_job_ticket_t *\fR\fIjob_ticket\fR, \fBpapi_stream_t *\fR\fIstream\f
  46 .fi

  48 .LP
  49 .nf
  50 \fBpapi_status_t\fR \fBpapiJobStreamWrite\fR(\fBpapi_service_t\fR \fIhandle\fR,
  51      \fBpapi_stream_t\fR \fIstream\fR, \fBvoid *\fR\fIbuffer\fR, \fBsize_t\fR \f
  52 .fi

  54 .LP
  55 .nf
  56 \fBpapi_status_t\fR \fBpapiJobStreamClose\fR(\fBpapi_service_t\fR \fIhandle\fR,
  57      \fBpapi_stream_t\fR \fIstream\fR, \fBpapi_job_t *\fR\fIjob\fR);
  58 .fi

  60 .LP
```



```
  61 .nf
  62 \fBpapi_status_t\fR \fBpapiJobQuery\fR(\fBpapi_service_t\fR \fIhandle\fR,
  63      \fBchar *\fR\fIprinter\fR, \fBint32_t\fR \fIjob_id\fR, \fBchar **\fR\fIrequ
  64      \fBpapi_job_t *\fR\fIjob\fR);
  65 .fi

  67 .LP
  68 .nf
  69 \fBpapi_status_t\fR \fBpapiJobModify\fR(\fBpapi_service_t\fR \fIhandle\fR,
  70      \fBchar *\fR\fIprinter\fR, \fBint32_t\fR \fIjob_id\fR,
  71      \fBpapi_attribute_t **\fR\fIattributes\fR, \fBpapi_job_t *\fR\fIjob\fR);
  72 .fi

  74 .LP
  75 .nf
  76 \fBpapi_status_t\fR \fBpapiJobMove\fR(\fBpapi_service_t\fR \fIhandle\fR,
  77      \fBchar *\fR\fIprinter\fR, \fBint32_t\fR \fIjob_id\fR, \fBchar *\fR\fIdesti
  78 .fi

  80 .LP
  81 .nf
  82 \fBpapi_status_t\fR \fBpapiJobCancel\fR(\fBpapi_service_t\fR \fIhandle\fR,
  83      \fBchar *\fR\fIprinter\fR, \fBint32_t\fR \fIjob_id\fR);
  84 .fi

  86 .LP
  87 .nf
  88 \fBpapi_status_t\fR \fBpapiJobHold\fR(\fBpapi_service_t\fR \fIhandle\fR,
  89      \fBchar *\fR\fIprinter\fR, \fBint32_t\fR \fIjob_id\fR);
  90 .fi

  92 .LP
  93 .nf
  94 \fBpapi_status_t\fR \fBpapiJobRelease\fR(\fBpapi_service_t\fR \fIhandle\fR,
  95      \fBchar *\fR\fIprinter\fR, \fBint32_t\fR \fIjob_id\fR);
  96 .fi

  98 .LP
  99 .nf
 100 \fBpapi_status_t\fR \fBpapiJobRestart\fR(\fBpapi_service_t\fR \fIhandle\fR,
 101      \fBchar *\fR\fIprinter\fR, \fBint32_t\fR \fIjob_id\fR);
 102 .fi

 104 .LP
 105 .nf
 106 \fBpapi_status_t\fR \fBpapiJobPromote\fR(\fBpapi_service_t\fR \fIhandle\fR,
 107      \fBchar *\fR\fIprinter\fR, \fBint32_t\fR \fIjob_id\fR);
 108 .fi

 110 .LP
 111 .nf
 112 \fBpapi_attribute_t **\fR\fBpapiJobGetAttributeList\fR(\fBpapi_job_t\fR \fIjob\f
 113 .fi

 115 .LP
 116 .nf
 117 \fBchar *\fR\fBpapiJobGetPrinterName\fR(\fBpapi_job_t\fR \fIjob\fR);
 118 .fi

 120 .LP
 121 .nf
 122 \fBint32_t\fR \fBpapiJobGetId\fR(\fBpapi_job_t\fR \fIjob\fR);
 123 .fi

 125 .LP
 126 .nf
```

```
\fBpapi_job_ticket_t *\fR\fBpapiJobGetJobTicket\fR(\fBpapi_job_t\fR \fIjob\fR);
.fi

.LP
.nf
\fBvoid\fR \fBpapiJobFree\fR(\fBpapi_job_t\fR \fIjob\fR);
.fi

.LP
.nf
\fBvoid\fR \fBpapiJobListFree\fR(\fBpapi_job_t *\fR\fIjobs\fR);
.fi

.SH PARAMETERS
.sp
.ne 2
.na
\fB\fIattributes\fR\fR
.ad
.RS 19n
a set of attributes to be applied to a printer object
.RE

.sp
.ne 2
.na
\fB\fIbuffer\fR\fR
.ad
.RS 19n
a buffer of data to be written to the job stream
.RE

.sp
.ne 2
.na
\fB\fIbufflen\fR\fR
.ad
.RS 19n
the size of the supplied buffer
.RE

.sp
.ne 2
.na
\fB\fIdestination\fR\fR
.ad
.RS 19n
the name of the printer where a print job should be relocated, which must
reside within the same print services as the job is currently queued
.RE

.sp
.ne 2
.na
\fB\fIfiles\fR\fR
.ad
.RS 19n
files to use during job submission
.RE

.sp
.ne 2
.na
\fB\fIhandle\fR\fR
.ad
.RS 19n
```



```
a pointer to a handle to be used for all PAPI operations that is created by
calling \fBpapiServiceCreate()\fR
.RE

.sp
.ne 2
.na
\fB\fIjob\fR\fR
.ad
.RS 19n
a pointer to a printer object (initialized to NULL) to be filled in by
\fBpapiJobQuery()\fR, \fBpapiJobSubmit()\fR, \fBpapiJobSubmitByReference()\fR,
\fBpapiJobValidate()\fR, \fBpapiJobStreamClose()\fR, and \fBpapiJobModify()\fR
.RE

.sp
.ne 2
.na
\fB\fIjob_attributes\fR\fR
.ad
.RS 19n
attributes to apply during job creation or modification
.RE

.sp
.ne 2
.na
\fB\fIjob_id\fR\fR
.ad
.RS 19n
ID number of the job reported on or manipulated
.RE

.sp
.ne 2
.na
\fB\fIjob_ticket\fR\fR
.ad
.RS 19n
unused
.RE

.sp
.ne 2
.na
\fB\fIjobs\fR\fR
.ad
.RS 19n
a list of job objects returned by \fBpapiPrinterListJobs()\fR or
\fBpapiPrinterPurgeJobs()\fR
.RE

.sp
.ne 2
.na
\fB\fIprinter\fR\fR
.ad
.RS 19n
name of the printer where the job is or should reside
.RE

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

```
.RS 19n
a null-terminated array of pointers to attribute names requested during job
enumeration (\fBpapiPrinterListJobs()\fR) or job query (\fBpapiJobQuery()\fR)
.RE

.sp
.ne 2
.na
\fB\fIstream\fR\fR
.ad
.RS 19n
a communication endpoint for sending print job data
.RE

.SH DESCRIPTION
.sp
.LP
The \fBpapiJobSubmit()\fR function creates a print job containing the passed in
files with the supplied attributes.  When the function returns, the data in the
passed files will have been copied by the print service. A job object is
returned that reflects the state of the job.
.sp
.LP
The \fBpapiJobSubmitByReference()\fR function creates a print job containing
the passed in files with the supplied attributes.  When the function returns,
the data in the passed files might have been copied by the print service. A job
object is returned that reflects the state of the job.
.sp
.LP
The \fBpapiJobStreamOpen()\fR, \fBpapiJobStreamWrite()\fR,
\fBpapiJobStreamClose()\fR functions create a print job by opening a stream,
writing to the stream, and closing it.
.sp
.LP
The \fBpapiJobValidate()\fR function validates that the supplied attributes and
files will result in a valid print job.
.sp
.LP
The \fBpapiJobQuery()\fR function retrieves job information from the print
service.
.sp
.LP
The \fBpapiJobModify()\fR function modifies a queued job according to the
attribute list passed into the call.  A job object is returned that reflects
the state of the job after the modification has been applied.
.sp
.LP
The \fBpapiJobMove()\fR function moves a job from its current queue to the
named destination within the same print service.
.sp
.LP
The \fBpapiJobCancel()\fR function removes a job from the queue.
.sp
.LP
The \fBpapiJobHold()\fR and \fBpapiJobRelease()\fR functions set the job state
to "held" or "idle" to indicate whether the job is eligible for processing.
.sp
.LP
The \fBpapiJobRestart()\fR function restarts processing of a currently queued
print job.
.sp
.LP
The \fBpapiJobGetAttributeList()\fR function returns a list of attributes
describing the job. This list can be searched and/or enumerated using
\fBpapiAttributeList*()\fR calls. See \fBpapiAttributeListAddValue\fR(3PAPI).
.sp
```



```
.LP
The \fBpapiJobGetPrinterName()\fR function returns the name of the queue where
the job is currently queued.
.sp
.LP
The \fBpapiJobGetId()\fR function returns a job identifier number from the job
object passed in.
.sp
.LP
The \fBpapiJobPromote()\fR function moves a job to the head of the print queue.
.sp
.LP
The \fBpapiJobGetJobTicket()\fR function retrieves a pointer to a job ticket
associated with the job object.
.sp
.LP
The \fBpapiJobFree()\fR and \fBpapiJobListFree()\fR functions deallocate memory
allocated for the return of printer object(s) from functions that return
printer objects.
.SH RETURN VALUES
.sp
.LP
Upon successful completion, all \fBpapiJob*()\fR functions that return a value
return \fBPAPI_OK\fR. Otherwise, they return an appropriate \fBpapi_status_t\fR
indicating the type of failure.
.sp
.LP
Upon successful completion, \fBpapiJobGetAttributeList()\fR returns a pointer
to the requested data. Otherwise, it returns \fINULL\fR.
.SH EXAMPLES
.LP
\fBExample 1 \fREnumerate all jobs in a queue
.sp
.in +2
.nf
/*
 * program to enumerate queued jobs using PAPI interfaces.
 */
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <libintl.h>
#include <pwd.h>
#include <papi.h>

static int
authCB(papi_service_t svc, void *app_data)
{
    char prompt[BUFSIZ];
    char *user, *svc_name, *passphrase;

    /* get the name of the service we are contacting */
    if ((svc_name = papiServiceGetServiceName(svc)) == NULL)
            return (-1);

    /* find out who we are supposed to be */
    if ((user = papiServiceGetUserName(svc)) == NULL) {
            struct passwd *pw;

            if ((pw = getpwuid(getuid())) != NULL)
                    user = pw->pw_name;
            else
                    user = "nobody";
    }

    /* build the prompt string */
```

```
391     snprintf(prompt, sizeof (prompt),
392             gettext("passphrase for %s to access %s: "), user,
393                   svc_name);

395     /* ask for the passphrase */
396     if ((passphrase = getpassphrase(prompt)) != NULL)
397             papiServiceSetPassword(svc, passphrase);

399     return (0);
400 }

402 /*ARGSUSED*/
403 int
404 main(int ac, char *av[])
405 {
406     papi_status_t status;
407     papi_service_t svc = NULL;
408     papi_job_t *jobs = NULL;
409     char *svc_name = NULL;
410     char *pname = "unknown";
411     int c;

413     while ((c = getopt(ac, av, "s:p:")) != EOF)
414         switch (c) {
415         case 's':
416             svc_name = optarg;
417             break;
418         case 'p':
419             pname = optarg;
420             break;
421         }

423     status = papiServiceCreate(&svc, svc_name, NULL, NULL, authCB,
424                             PAPI_ENCRYPT_NEVER, NULL);

426     if (status != PAPI_OK) {
427         printf("papiServiceCreate(%s): %s\en", svc_name ? svc_name :
427         printf("papiServiceCreate(%s): %s\n", svc_name ? svc_name :
428                         "NULL", papiStatusString(status));
429         papiServiceDestroy(svc);
430         exit(1);
431     }

433     status = papiPrinterListJobs(svc, pname, NULL, 0, 0, &jobs);
434     if (status != PAPI_OK) {
435         printf("papiPrinterListJobs(%s): %s\en", pname,
435         printf("papiPrinterListJobs(%s): %s\n", pname,
436                         papiStatusString(status));
437         papiServiceDestroy(svc);
438         exit(1);
439     }

441     if (jobs != NULL) {
442         int i;

444         for (i = 0; jobs[i] != NULL; i++) {
445             papi_attribute_t **list = papiJobGetAttributeList(jobs[i]);

447             if (list != NULL) {
448                 char *name = "unknown";
449                             int32_t id = 0;
450                 char *buffer = NULL;
451                 size_t size = 0;

453                (void) papiAttributeListGetString(list, NULL,
454                                     "printer-name", &name);
```



```
455                (void) papiAttributeListGetInteger(list, NULL,
456                                     "job-id", &id);
457                 while (papiAttributeListToString(list, "\n\t", buffer,
458                             size) != PAPI_OK)
459                     buffer = realloc(buffer, size += BUFSIZ);

461                 printf("%s-%d:\en\t%s\en", name, id, buffer);
461                 printf("%s-%d:\n\t%s\n", name, id, buffer);
462                 free(buffer);
463             }
464         }
465         papiJobListFree(jobs);
466     }

468     papiServiceDestroy(svc);

470     exit(0);
471 }
```

*unchanged_portion_omitted_*

```
522 /*ARGSUSED*/
523 int
524 main(int ac, char *av[])
525 {
526     papi_status_t status;
527     papi_service_t svc = NULL;
528     papi_job_t job = NULL;
529     char *svc_name = NULL;
530     char *pname = "unknown";
531     int id = 0;
532     int c;

534     while ((c = getopt(ac, av, "s:p:j:")) != EOF)
535         switch (c) {
536         case 's':
537             svc_name = optarg;
538             break;
539         case 'p':
540             pname = optarg;
541             break;
542         case 'j':
543             id = atoi(optarg);
544             break;
545         }

547     status = papiServiceCreate(&svc, svc_name, NULL, NULL, authCB,
548                             PAPI_ENCRYPT_NEVER, NULL);

550     if (status != PAPI_OK) {
551         printf("papiServiceCreate(%s): %s\en", svc_name ? svc_name :
551         printf("papiServiceCreate(%s): %s\n", svc_name ? svc_name :
552                         "NULL", papiStatusString(status));
553         papiServiceDestroy(svc);
554         exit(1);
555     }

557     status = papiJobQuery(svc, pname, id, NULL, &job);
558     if ((status == PAPI_OK) && (job != NULL)) {
559         papi_attribute_t **list = papiJobGetAttributeList(job);

561         if (list != NULL) {
562             char *name = "unknown";
563             int32_t id = 0;
564             char *buffer = NULL;
565             size_t size = 0;
```

```
567             (void) papiAttributeListGetString(list, NULL,
568                                     "printer-name", &name);
569             (void) papiAttributeListGetInteger(list, NULL,
570                                     "job-id", &id);
571             while (papiAttributeListToString(list, "\en\et", buffer, size)
571             while (papiAttributeListToString(list, "\n\t", buffer, size)
572                         != PAPI_OK)
573                 buffer = realloc(buffer, size += BUFSIZ);

575             printf("%s-%d:\en\et%s\n", name, id, buffer);
575             printf("%s-%d:\n\t%s\n", name, id, buffer);
576             free(buffer);
577         }
578     } else
579         printf("papiJobQuery(%s-%d): %s\en", pname, id,
579         printf("papiJobQuery(%s-%d): %s\n", pname, id,
580                     papiStatusString(status));

582     papiJobFree(job);
583     papiServiceDestroy(svc);

585     exit(0);
586 }
```

*unchanged_portion_omitted_*

```
637 /*ARGSUSED*/
638 int
639 main(int ac, char *av[])
640 {
641     papi_status_t status;
642     papi_service_t svc = NULL;
643     papi_stream_t stream = NULL;
644     papi_job_t job = NULL;
645     papi_attribute_t **attrs = NULL;
646     char *svc_name = NULL;
647     char *pname = "unknown";
648     int id = 0;
649     int c;
650     int rc;
651     char buf[BUFSIZ];

653     while ((c = getopt(ac, av, "s:p:")) != EOF)
654         switch (c) {
655         case 's':
656             svc_name = optarg;
657             break;
658         case 'p':
659             pname = optarg;
660             break;
661         }

663     status = papiServiceCreate(&svc, svc_name, NULL, NULL, authCB,
664                             PAPI_ENCRYPT_NEVER, NULL);

666     if (status != PAPI_OK) {
667         printf("papiServiceCreate(%s): %s\en", svc_name ? svc_name :
667         printf("papiServiceCreate(%s): %s\n", svc_name ? svc_name :
668                     "NULL", papiStatusString(status));
669         papiServiceDestroy(svc);
670         exit(1);
671     }

673     papiAttributeListAddInteger(&attrs, PAPI_ATTR_EXCL, "copies", 1);
674     papiAttributeListAddString(&attrs, PAPI_ATTR_EXCL,
675                 "document-format", "application/octet-stream");
676     papiAttributeListAddString(&attrs, PAPI_ATTR_EXCL,
```



```
677                 "job-title", "Standard Input");

679     status = papiJobStreamOpen(svc, pname, attrs, NULL, &stream);
680     while ((status == PAPI_OK) && ((rc = read(0, buf,
681                     sizeof (buf))) > 0))
682             status = papiJobStreamWrite(svc, stream, buf, rc);

684     if (status == PAPI_OK)
685             status = papiJobStreamClose(svc, stream, &job);


688     if ((status == PAPI_OK) && (job != NULL)) {
689         papi_attribute_t **list = papiJobGetAttributeList(job);

691         if (list != NULL) {
692             char *name = "unknown";
693             int32_t id = 0;
694             char *buffer = NULL;
695             size_t size = 0;

697             (void) papiAttributeListGetString(list, NULL,
698                                     "printer-name", &name);
699             (void) papiAttributeListGetInteger(list, NULL,
700                                     "job-id", &id);
701             while (papiAttributeListToString(list, "\n\t", buffer, size)
702                         != PAPI_OK)
703                 buffer = realloc(buffer, size += BUFSIZ);

705             printf("%s-%d:\en\t%s\en", name, id, buffer);
705             printf("%s-%d:\n\t%s\n", name, id, buffer);
706             free(buffer);
707         }
708     } else
709         printf("papiJobStream*(%s-%d): %s\en", pname, id,
709         printf("papiJobStream*(%s-%d): %s\n", pname, id,
710                     papiStatusString(status));

712     papiJobFree(job);
713     papiServiceDestroy(svc);

715     exit(0);
716 }
```

*unchanged_portion_omitted_*

```
**********************************************************
   13904 Tue Sep 10 18:35:20 2013
new/usr/src/man/man3papi/papiPrintersList.3papi
4023 - Typo in file(1) manpage and various others
**********************************************************
'\" te
.\" Copyright (c) 2007, 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 PAPIPRINTERSLIST 3PAPI "Sep 10, 2013"
.TH PAPIPRINTERSLIST 3PAPI "Jan 17, 2007"
.SH NAME
papiPrintersList, papiPrinterQuery, papiPrinterAdd, papiPrinterModify,
papiPrinterRemove, papiPrinterDisable, papiPrinterEnable, papiPrinterPause,
papiPrinterResume, papiPrinterPurgeJobs, papiPrinterListJobs,
papiPrinterGetAttributeList, papiPrinterFree, papiPrinterListFree \- print
object manipulation
.SH SYNOPSIS
.LP
.nf
cc [ \fIflag\fR\&.\|.\|. ] \fIfile\fR\&.\|.\|. \fB-lpapi\fR [ \fIlibrary\fR\&.\|
#include <papi.h>

\fBpapi_status_t\fR \fBpapiPrintersList\fR(\fBpapi_service_t\fR \fIhandle\fR,
     \fBchar **\fR\fIrequested_attrs\fR, \fBpapi_filter_t *\fR\fIfilter\fR,
     \fBpapi_printer_t **\fR\fIprinters\fR);
.fi

.LP
.nf
\fBpapi_status_t\fR \fBpapiPrinterQuery\fR(\fBpapi_service_t\fR \fIhandle\fR, \f
     \fBchar **\fR\fIrequested_attrs\fR, \fBpapi_attribute_t **\fR\fIjob_attribu
     \fBpapi_printer_t *\fR\fIprinter\fR);
.fi

.LP
.nf
\fBpapi_status_t\fR \fBpapiPrinterAdd\fR(\fBpapi_service_t\fR \fIhandle\fR, \fBc
     \fBpapi_attribute_t **\fR\fIattributes\fR, \fBpapi_printer_t *\fR\fIprinter
.fi

.LP
.nf
\fBpapi_status_t\fR \fBpapiPrinterModify\fR(\fBpapi_service_t\fR \fIhandle\fR, \
     \fBpapi_attribute_t **\fR\fIattributes\fR, \fBpapi_printer_t *\fR\fIprinter
.fi

.LP
.nf
\fBpapi_status_t\fR \fBpapiPrinterRemove\fR(\fBpapi_service_t\fR \fIhandle\fR, \
.fi

.LP
.nf
\fBpapi_status_t\fR \fBpapiPrinterDisable\fR(\fBpapi_service_t\fR \fIhandle\fR,
     \fBchar *\fR\fImessage\fR);
.fi

.LP
.nf
\fBpapi_status_t\fR \fBpapiPrinterEnable\fR(\fBpapi_service_t\fR \fIhandle\fR, \
.fi

.LP
.nf
```



```
\fBpapi_status_t\fR \fBpapiPrinterPause\fR(\fBpapi_service_t\fR \fIhandle\fR, \f
     \fBchar *\fR\fImessage\fR);
.fi

.LP
.nf
\fBpapi_status_t\fR \fBpapiPrinterResume\fR(\fBpapi_service_t\fR \fIhandle\fR, \
.fi

.LP
.nf
\fBpapi_status_t\fR \fBpapiPrinterPurgeJobs\fR(\fBpapi_service_t\fR \fIhandle\fR
     \fBpapi_job_t **\fR\fIjobs\fR);
.fi

.LP
.nf
\fBpapi_status_t\fR \fBpapiPrinterListJobs\fR(\fBpapi_service_t\fR \fIhandle\fR,
     \fBchar **\fR\fIrequested_attrs\fR, \fBint\fR \fItype_mask\fR, \fBint\fR \f
     \fBpapi_job_t **\fR\fIjobs\fR);
.fi

.LP
.nf
\fBpapi_attribute_t **\fR\fBpapiPrinterGetAttributeList\fR(\fBpapi_printer_t\fR
.fi

.LP
.nf
\fBvoid\fR \fBpapiPrinterFree\fR(\fBpapi_printer_t\fR \fIprinter\fR);
.fi

.LP
.nf
\fBvoid\fR \fBpapiPrinterListFree\fR(\fBpapi_printer_t *\fR\fIprinters\fR);
.fi

.SH PARAMETERS
.sp
.ne 2
.na
\fB\fIattributes\fR\fR
.ad
.RS 19n
a set of attributes to be applied to a printer object
.RE

.sp
.ne 2
.na
\fB\fIfilter\fR\fR
.ad
.RS 19n
a filter to be applied during printer enumeration
.RE

.sp
.ne 2
.na
\fB\fIhandle\fR\fR
.ad
.RS 19n
a pointer to a handle to be used for all PAPI operations, created by calling
\fBpapiServiceCreate()\fR
.RE
```

```
.sp
.ne 2
.na
\fB\fIjob_attributes\fR\fR
.ad
.RS 19n
unused
.RE

.sp
.ne 2
.na
\fB\fIjobs\fR\fR
.ad
.RS 19n
a pointer to a list to return job objects (initialized to \fINULL\fR)
enumerated by \fBpapiPrinterGetJobs()\fR
.RE

.sp
.ne 2
.na
\fB\fImax_num_jobs\fR\fR
.ad
.RS 19n
the maximum number of jobs to return from a \fBpapiPrinterGetJobs()\fR request
.RE

.sp
.ne 2
.na
\fB\fImessage\fR\fR
.ad
.RS 19n
a message to be associated with a printer while disabled or paused
.RE

.sp
.ne 2
.na
\fB\fIname\fR\fR
.ad
.RS 19n
the name of the printer object being operated on
.RE

.sp
.ne 2
.na
\fB\fIprinter\fR\fR
.ad
.RS 19n
a pointer to a printer object (initialized to \fINULL\fR) to be filled in by
\fBpapiPrinterQuery()\fR, \fBpapiPrinterAdd()\fR, and \fBpapiPrinterModify()\fR
.RE

.sp
.ne 2
.na
\fB\fIprinters\fR\fR
.ad
.RS 19n
a pointer to a list to return printer objects (initialized to \fINULL\fR)
enumerated by \fBpapiPrintersList()\fR
.RE
```



```
.sp
.ne 2
.na
\fB\fIrequested_attrs\fR\fR
.ad
.RS 19n
a null-terminated array of pointers to attribute names requested during printer
enumeration (\fBpapiPrintersList()\fR), printer query
(\fBpapiPrinterQuery()\fR), or job enumeration (\fBpapiPrinterListJobs()\fR)
.RE

.sp
.ne 2
.na
\fB\fItype_mask\fR\fR
.ad
.RS 19n
a bit field indicating which type of jobs to return \fBPAPI_LIST_JOBS_OTHERS\fR
include jobs submitted by others.  The default is to report only on your own
jobs
.sp
.ne 2
.na
\fB\fBPAPI_LIST_JOBS_COMPLETED\fR\fR
.ad
.sp .6
.RS 4n
include completed jobs
.RE

.sp
.ne 2
.na
\fB\fBPAPI_LIST_JOBS_NOT_COMPLETED\fR\fR
.ad
.sp .6
.RS 4n
include jobs not complete
.RE

.sp
.ne 2
.na
\fB\fBPAPI_LIST_JOBS_ALL\fR\fR
.ad
.sp .6
.RS 4n
report on all jobs
.RE

.RE

.SH DESCRIPTION
.sp
.LP
The \fBpapiPrintersList()\fR function retrieves the requested attributes from
the print service(s) for all available printers.  Because the Solaris
implementation is name service-enabled, applications should retrieve only the
\fBprinter-name\fR and \fBprinter-uri-supported\fR attributes using this
function, thereby reducing the overhead involved in generating a printer list.
Further integration of printer state and capabilities can be performed with
\fBpapiPrinterQuery()\fR.
.sp
.LP
The \fBpapiPrinterAdd()\fR, \fBpapiPrinterModify()\fR, and
\fBpapiPrinterRemove()\fR functions allow for creation, modification, and
```

```
259 removal of print queues.  Print queues are added or modified according to the
260 attribute list passed into the call. A printer object is returned that reflects
261 the configuration of the printer after the addition or modification has been
262 applied.  At this time, they provide only minimal functionality and only for
263 the LP print service.
264 .sp
265 .LP
266 The \fBpapiPrinterDisable()\fR and \fBpapiPrinterEnable()\fR functions allow
267 applications to turn off and on queueing (accepting print requests) for a print
268 queue. The \fBpapiPrinterEnable()\fR and \fBpapiPrinterDisable()\fR functions
269 allow applications to turn on and off print job processing for a print queue.
270 .sp
271 .LP
272 The \fBpapiPrinterPause()\fR function stops queueing of print jobs on the named
273 print queue.
274 .sp
275 .LP
276 The \fBpapiPrinterResume()\fR function resumes queueing of print jobs on the
277 named print queue.
278 .sp
279 .LP
280 The \fBpapiPrinterPurgeJobs()\fR function allows applications to delete all
281 print jobs that it has privilege to remove.  A list of cancelled jobs is
282 returned in the jobs argument.
283 .sp
284 .LP
285 The \fBpapiPrinterListJobs()\fR function enumerates print jobs on a particular
286 queue. \fBpapiPrinterGetAttributeList()\fR retrieves an attribute list from a
287 printer object.
288 .sp
289 .LP
290 The \fBpapiPrinterGetAttributeList()\fR function retrieves an attribute list
291 from a printer object returned from \fBpapiPrinterQuery()\fR,
292 \fBpapiPrintersList()\fR, \fBpapiPrinterModify()\fR, and
293 \fBpapiPrinterAdd()\fR.  This attribute list can be searched for various
294 information about the printer object.
295 .sp
296 .LP
297 The \fBpapiPrinterFree()\fR and \fBpapiPrinterListFree()\fR functions
298 deallocate memory allocated for the return of printer object(s) from functions
299 that return printer objects.
300 .SH RETURN VALUES
301 .sp
302 .LP
303 Upon successful completion, all functions that return a value return
304 \fBPAPI_OK\fR. Otherwise, they return an appropriate \fBpapi_status_t()\fR
305 indicating the type of failure.
306 .sp
307 .LP
308 Upon successful completion, \fBpapiPrinterGetAttributeList()\fR returns a
309 pointer to the requested data. Otherwise, it returns \fINULL\fR.
310 .SH EXAMPLES
311 .LP
312 \fBExample 1 \fREnumerate all available printers.
313 .sp
314 .in +2
315 .nf
316 #include <stdio.h>
317 #include <stdlib.h>
318 #include <unistd.h>
319 #include <libintl.h>
320 #include <pwd.h>
321 #include <papi.h>

323 static int
324 authCB(papi_service_t svc, void *app_data)
```



```
325 {
326     char prompt[BUFSIZ];
327     char *user, *svc_name, *passphrase;

329     /* get the name of the service we are contacting */
330     if ((svc_name = papiServiceGetServiceName(svc)) == NULL)
331             return (-1);

333     /* find out who we are supposed to be */
334     if ((user = papiServiceGetUserName(svc)) == NULL) {
335             struct passwd *pw;

337             if ((pw = getpwuid(getuid())) != NULL)
338                     user = pw->pw_name;
339             else
340                     user = "nobody";
341     }

343     /* build the prompt string */
344     snprintf(prompt, sizeof (prompt),
345             gettext("passphrase for %s to access %s: "), user,
346                    svc_name);

348     /* ask for the passphrase */
349     if ((passphrase = getpassphrase(prompt)) != NULL)
350             papiServiceSetPassword(svc, passphrase);

352     return (0);
353 }

355 /*ARGSUSED*/
356 int
357 main(int ac, char *av[])
358 {
359     papi_status_t status;
360     papi_service_t svc = NULL;
361     papi_printer_t *printers = NULL;
362     char *attrs[] = { "printer-name", "printer-uri-supported", NULL };
363     char *svc_name = NULL;
364     int c;

366     while ((c = getopt(ac, av, "s:")) != EOF)
367         switch (c) {
368         case 's':
369             svc_name = optarg;
370             break;
371         }

373     status = papiServiceCreate(&svc, svc_name, NULL, NULL, authCB,
374                                PAPI_ENCRYPT_NEVER, NULL);

376     if (status != PAPI_OK) {
377         printf("papiServiceCreate(%s): %s\en", svc_name ? svc_name :
378                     "NULL", papiStatusString(status));
379         papiServiceDestroy(svc);
380         exit(1);
381     }

383     status = papiPrintersList(svc, attrs, NULL, &printers);
384     if (status != PAPI_OK) {
385         printf("papiPrintersList(%s): %s\en", svc_name ? svc_name :
386                     "NULL", papiStatusString(status));
387         papiServiceDestroy(svc);
388         exit(1);
389     }
```

```
391     if (printers != NULL) {
392         int i;

394         for (i = 0; printers[i] != NULL; i++) {
395             papi_attribute_t **list =
396                 papiPrinterGetAttributeList(printers[i]);

398             if (list != NULL) {
399                 char *name = "unknown";
400                 char *uri = "unknown";

402                 (void) papiAttributeListGetString(list, NULL,
403                                 "printer-name", &name);

405                 (void) papiAttributeListGetString(list, NULL,
406                                 "printer-uri-supported", &uri);
407                 printf("%s is %s\en", name, uri);
407                 printf("%s is %s\n", name, uri);
408             }
409         }
410         papiPrinterListFree(printers);
411     }

413     papiServiceDestroy(svc);

415     exit(0);
416 }
```

unchanged_portion_omitted_

```
**********************************************************
   14726 Tue Sep 10 18:35:20 2013
new/usr/src/man/man3socket/getipnodebyname.3socket
4023 - Typo in file(1) manpage and various others
**********************************************************
'\" te
.\" Copyright (c) 2007, 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 GETIPNODEBYNAME 3SOCKET "Sep 10, 2013"
.TH GETIPNODEBYNAME 3SOCKET "Aug 22, 2007"
.SH NAME
getipnodebyname, getipnodebyaddr, freehostent \- get IP node entry
.SH SYNOPSIS
.LP
.nf
\fBcc\fR [ \fIflag\fR... ] \fIfile\fR... \fB-lsocket\fR \fB -lnsl \fR [ \fIlibra
#include <sys/socket.h>
#include <netdb.h>

\fBstruct hostent *\fR\fBgetipnodebyname\fR(\fBconst char *\fR\fIname\fR, \fBint
     \fBint\fR \fIflags\fR, \fBint *\fR\fIerror_num\fR);
.fi

.LP
.nf
\fBstruct hostent *\fR\fBgetipnodebyaddr\fR(\fBconst void *\fR\fIsrc\fR, \fBsize
     \fBint\fR \fIaf\fR, \fBint *\fR\fIerror_num\fR);
.fi

.LP
.nf
\fBvoid\fR \fBfreehostent\fR(\fBstruct hostent *\fR\fIptr\fR);
.fi

.SH PARAMETERS
.sp
.ne 2
.na
\fB\fIaf\fR\fR
.ad
.RS 13n
Address family
.RE

.sp
.ne 2
.na
\fB\fIflags\fR\fR
.ad
.RS 13n
Various flags
.RE

.sp
.ne 2
.na
\fB\fIname\fR\fR
.ad
.RS 13n
Name of host
.RE

.sp
.ne 2
```



```
.na
\fB\fIerror_num\fR\fR
.ad
.RS 13n
Error storage
.RE

.sp
.ne 2
.na
\fB\fIsrc\fR\fR
.ad
.RS 13n
Address for lookup
.RE

.sp
.ne 2
.na
\fB\fIlen\fR\fR
.ad
.RS 13n
Length of address
.RE

.sp
.ne 2
.na
\fB\fIptr\fR\fR
.ad
.RS 13n
Pointer to \fBhostent\fR structure
.RE

.SH DESCRIPTION
.sp
.LP
The \fBgetipnodebyname()\fR function searches the \fBipnodes\fR database from
the beginning. The function finds the first \fBh_name\fR member that matches
the hostname specified by \fIname\fR. The function takes an \fIaf\fR argument
that specifies the address family. The address family can be \fBAF_INET\fR for
IPv4 addresses or \fBAF_INET6\fR for IPv6 addresses. The \fIflags\fR argument
determines what results are returned based on the value of \fIflags\fR. If the
\fIflags\fR argument is set to \fB0\fR (zero), the default operation of the
function is specified as follows:
.RS +4
.TP
.ie t \(bu
.el o
If the \fIaf\fR argument is \fBAF_INET\fR, a query is made for an IPv4 address.
If successful, IPv4 addresses are returned and the \fBh_length\fR member of the
\fBhostent\fR structure is 4. Otherwise, the function returns a \fINULL\fR
pointer.
.RE
.RS +4
.TP
.ie t \(bu
.el o
If the \fIaf\fR argument is \fBAF_INET6\fR, a query is made for an IPv6
address. If successful, IPv6 addresses are returned and the \fBh_length\fR
member of the \fBhostent\fR structure is 16. Otherwise, the function returns a
\fINULL\fR pointer.
.RE
.sp
.LP
The \fIflags\fR argument changes the default actions of the function. Set the
```

\fIflags\fR argument with a logical \fBOR\fR operation on any of combination of the following values:

- \fBAI_V4MAPPED\fR
- \fBAI_ALL\fR
- \fBAI_ADDRCONFIG\fR

The special flags value, \fBAI_DEFAULT\fR, should handle most applications. Porting simple applications to use IPv6 replaces the call

```
hptr = gethostbyname(name);
```

with

```
hptr = getipnodebyname(name, AF_INET6, AI_DEFAULT, &error_num);
```

The \fIflags\fR value \fB0\fR (zero) implies a strict interpretation of the \fIaf\fR argument:

- If \fIflags\fR is \fB0\fR and \fIaf\fR is \fBAF_INET\fR, the caller wants only IPv4 addresses. A query is made for \fBA\fR records. If successful, IPv4 addresses are returned and the \fBh_length\fR member of the \fBhostent\fR structure is 4. Otherwise, the function returns a \fINULL\fR pointer.
- If \fIflags\fR is \fB0\fR and \fIaf\fR is \fBAF_INET6\fR, the caller wants only IPv6 addresses. A query is made for \fBAAAA\fR records. If successful, IPv6 addresses are returned and the \fBh_length\fR member of the \fBhostent\fR structure is 16. Otherwise, the function returns a \fINULL\fR pointer.

Logically \fBOR\fR other constants into the \fIflags\fR argument to modify the

behavior of the \fBgetipnodebyname()\fR function.

- If the \fBAI_V4MAPPED\fR flag is specified with \fIaf\fR set to \fBAF_INET6\fR, the caller can accept IPv4-mapped IPv6 addresses. If no \fBAAAA\fR records are found, a query is made for \fBA\fR records. Any \fBA\fR records found are returned as IPv4-mapped IPv6 addresses and the \fBh_length\fR is 16. The \fBAI_V4MAPPED\fR flag is ignored unless \fIaf\fR equals \fBAF_INET6\fR.
- The \fBAI_ALL\fR flag is used in conjunction with the \fBAI_V4MAPPED\fR flag, exclusively with the IPv6 address family. When \fBAI_ALL\fR is logically \fBOR\fRed with \fBAI_V4MAPPED\fR flag, the caller wants all addresses: IPv6 and IPv4-mapped IPv6 addresses. A query is first made for \fBAAAA\fR records and, if successful, IPv6 addresses are returned. Another query is then made for \fBA\fR records. Any \fBA\fR records found are returned as IPv4-mapped IPv6 addresses and the \fBh_length\fR is 16. Only when both queries fail does the function return a \fINULL\fR pointer. The \fBAI_ALL\fR flag is ignored unless \fIaf\fR is set to \fBAF_INET6\fR.
- The \fBAI_ADDRCONFIG\fR flag specifies that a query for \fBAAAA\fR records should occur only when the node is configured with at least one IPv6 source address. A query for \fBA\fR records should occur only when the node is configured with at least one IPv4 source address. For example, if a node is configured with no IPv6 source addresses, \fIaf\fR equals \fBAF_INET6\fR, and the node name queried has both \fBAAAA\fR and \fBA\fR records, then:
  - A \fINULL\fR pointer is returned when only the \fBAI_ADDRCONFIG\fR value is specified.
  - The \fBA\fR records are returned as IPv4-mapped IPv6 addresses when the \fBAI_ADDRCONFIG\fR and \fBAI_V4MAPPED\fR values are specified.

The special flags value, \fBAI_DEFAULT\fR, is defined as

```
#define  AI_DEFAULT  (AI_V4MAPPED | AI_ADDRCONFIG)
```

The \fBgetipnodebyname()\fR function allows the \fIname\fR argument to be a node name or a literal address string: a dotted-decimal IPv4 address or an IPv6 hex address. Applications do not have to call \fBinet_pton\fR(3SOCKET) to handle literal address strings.

```
.sp
.LP
Four scenarios arise based on the type of literal address string and the value
of the \fIaf\fR argument. The two simple cases occur when \fIname\fR is a
dotted-decimal IPv4 address and \fIaf\fR equals \fBAF_INET\fR and when
\fIname\fR is an IPv6 hex address and \fIaf\fR equals \fBAF_INET6\fR. The
members of the returned \fBhostent\fR structure are:
.sp
.ne 2
.na
\fB\fBh_name\fR\fR
.ad
.RS 15n
Pointer to a copy of the name argument
.RE

.sp
.ne 2
.na
\fB\fBh_aliases\fR\fR
.ad
.RS 15n
\fINULL\fR pointer.
.RE

.sp
.ne 2
.na
\fB\fBh_addrtype\fR\fR
.ad
.RS 15n
Copy of the \fIaf\fR argument.
.RE

.sp
.ne 2
.na
\fB\fBh_length\fR\fR
.ad
.RS 15n
4 for \fBAF_INET\fR or 16 for \fBAF_INET6\fR.
.RE

.sp
.ne 2
.na
\fB\fBh_addr_list\fR\fR
.ad
.RS 15n
Array of pointers to 4-byte or 16-byte binary addresses. The array is
terminated by a \fINULL\fR pointer.
.RE

.SH RETURN VALUES
.sp
.LP
Upon successful completion, \fBgetipnodebyname()\fR and \fBgetipnodebyaddr()\fR
return a \fBhostent\fR structure. Otherwise they return \fINULL\fR.
.sp
.LP
The \fBhostent\fR structure does not change from the existing definition when
used with \fBgethostbyname\fR(3NSL). For example, host entries are represented
by the \fBstruct hostent\fR structure defined in <\fBnetdb.h\fR>:
.sp
.in +2
.nf
```



```
struct  hostent {
        char    *h_name;        /* canonical name of host */
        char    **h_aliases;    /* alias list */
        int     h_addrtype;     /* host address type */
        int     h_length;       /* length of address */
        char    **h_addr_list;  /* list of addresses */
};
.fi
.in -2

.sp
.LP
An error occurs when \fIname\fR is an IPv6 hex address and \fIaf\fR equals
\fBAF_INET\fR. The return value of the function is a \fINULL\fR pointer and
\fBerror_num\fR equals \fBHOST_NOT_FOUND\fR.
.sp
.LP
The \fBgetipnodebyaddr()\fR function has the same arguments as the existing
\fBgethostbyaddr\fR(3NSL) function, but adds an error number. As with
\fBgetipnodebyname()\fR, \fBgetipnodebyaddr()\fR is thread-safe. The
\fBerror_num\fR value is returned to the caller with the appropriate error code
to support thread-safe error code returns. The following error conditions can
be returned for \fBerror_num\fR:
.sp
.ne 2
.na
\fB\fBHOST_NOT_FOUND\fR\fR
.ad
.RS 18n
Host is unknown.
.RE

.sp
.ne 2
.na
\fB\fBNO_DATA\fR\fR
.ad
.RS 18n
No address is available for the \fIname\fR specified in the server request.
This error is not a soft error. Another type of \fIname\fR server request might
be successful.
.RE

.sp
.ne 2
.na
\fB\fBNO_RECOVERY\fR\fR
.ad
.RS 18n
An unexpected server failure occurred, which is a non-recoverable error.
.RE

.sp
.ne 2
.na
\fB\fBTRY_AGAIN\fR\fR
.ad
.RS 18n
This error is a soft error that indicates that the local server did not receive
a response from an authoritative server. A retry at some later time might be
successful.
.RE

.sp
.LP
One possible source of confusion is the handling of IPv4-mapped IPv6 addresses
```

```
391 and IPv4-compatible IPv6 addresses, but the following logic should apply:
392 .RS +4
393 .TP
394 1.
395 If \fIaf\fR is \fBAF_INET6\fR, and if \fIlen\fR equals 16, and if the IPv6
396 address is an IPv4-mapped IPv6 address or an IPv4-compatible IPv6 address, then
397 skip over the first 12 bytes of the IPv6 address, set \fIaf\fR to
398 \fBAF_INET\fR, and set \fIlen\fR to 4.
399 .RE
400 .RS +4
401 .TP
402 2.
403 If \fIaf\fR is \fBAF_INET\fR, lookup the \fIname\fR for the given IPv4
404 address.
405 .RE
406 .RS +4
407 .TP
408 3.
409 If \fIaf\fR is \fBAF_INET6\fR, lookup the \fIname\fR for the given IPv6
410 address.
411 .RE
412 .RS +4
413 .TP
414 4.
415 If the function is returning success, then the single address that is
416 returned in the \fBhostent\fR structure is a copy of the first argument to the
417 function with the same address family that was passed as an argument to this
418 function.
419 .RE
420 .sp
421 .LP
422 All four steps listed are performed in order.
423 .sp
424 .LP
425 This structure, and the information pointed to by this structure, are
426 dynamically allocated by \fBgetipnodebyname()\fR and \fBgetipnodebyaddr()\fR.
427 The \fBfreehostent()\fR function frees this memory.
428 .SH EXAMPLES
429 .LP
430 \fBExample 1 \fRGetting the Canonical Name, Aliases, and Internet IP Addresses
431 for a Given Hostname
432 .sp
433 .LP
434 The following is a sample program that retrieves the canonical name, aliases,
435 and all Internet IP addresses, both version 6 and version 4, for a given
436 hostname.

438 .sp
439 .in +2
440 .nf
441      #include <stdio.h>
442      #include <string.h>
443      #include <sys/types.h>
444      #include <sys/socket.h>
445      #include <netinet/in.h>
446      #include <arpa/inet.h>
447      #include <netdb.h>

449      main(int argc, const char **argv)
450      {
451      char abuf[INET6_ADDRSTRLEN];
452      int error_num;
453      struct hostent *hp;
454      char **p;

456          if (argc != 2) {
```



```
457              (void) printf("usage: %s hostname\en", argv[0]);
457              (void) printf("usage: %s hostname\n", argv[0]);
458              exit (1);
459          }

461      /* argv[1] can be a pointer to a hostname or literal IP address */
462      hp = getipnodebyname(argv[1], AF_INET6, AI_ALL | AI_ADDRCONFIG |
463         AI_V4MAPPED, &error_num);
464      if (hp == NULL) {
465         if (error_num == TRY_AGAIN) {
466             printf("%s: unknown host or invalid literal address "
467                 "(try again later)\en", argv[1]);
467                 "(try again later)\n", argv[1]);
468         } else {
469             printf("%s: unknown host or invalid literal address\en",
469             printf("%s: unknown host or invalid literal address\n",
470                 argv[1]);
471         }
472         exit (1);
473      }
474      for (p = hp->h_addr_list; *p != 0; p++) {
475         struct in6_addr in6;
476         char **q;

478         bcopy(*p, (caddr_t)&in6, hp->h_length);
479         (void) printf("%s\t%s", inet_ntop(AF_INET6, (void *)&in6,
480             abuf, sizeof(abuf)), hp->h_name);
481         for (q = hp->h_aliases; *q != 0; q++)
482         (void) printf(" %s", *q);
483         (void) putchar('\en');
483         (void) putchar('\n');
484      }
485      freehostent(hp);
486      exit (0);
487      }
488 .fi
489 .in -2

491 .SH ATTRIBUTES
492 .sp
493 .LP
494 See \fBattributes\fR(5) for descriptions of the following attributes:
495 .sp

497 .sp
498 .TS
499 box;
500 c | c
501 l | l .
502 ATTRIBUTE TYPE  ATTRIBUTE VALUE
503 _
504 Interface Stability     Committed
505 _
506 MT-Level        Safe
507 .TE

509 .SH SEE ALSO
510 .sp
511 .LP
512 \fBgetaddrinfo\fR(3SOCKET), \fBgethostbyname\fR(3NSL), \fBhtonl\fR(3SOCKET),
513 \fBinet\fR(3SOCKET), \fBnetdb.h\fR(3HEAD), \fBhosts\fR(4),
514 \fBnsswitch.conf\fR(4), \fBattributes\fR(5)
515 .SH NOTES
516 .sp
517 .LP
518 No enumeration functions are provided for IPv6. Existing enumeration functions
```

```
 519 such as \fBsethostent\fR(3NSL) do not work in combination with the
 520 \fBgetipnodebyname()\fR and \fBgetipnodebyaddr()\fR functions.
 521 .sp
 522 .LP
 523 All the functions that return a \fBstruct hostent\fR must always return the
 524 canonical in the \fBh_name\fR field. This name, by definition, is the
 525 well-known and official hostname shared between all aliases and all addresses.
 526 The underlying source that satisfies the request determines the mapping of the
 527 input name or address into the set of names and addresses in \fBhostent\fR.
 528 Different sources might make such as determination in different ways. If more
 529 than one alias and more than one address in \fBhostent\fR exist, no pairing is
 530 implied between the alias and address.
 531 .sp
 532 .LP
 533 The current implementations of these functions return or accept only addresses
 534 for the Internet address family (type \fBAF_INET\fR) or the Internet address
 535 family Version 6 (type \fBAF_INET6\fR).
 536 .sp
 537 .LP
 538 IPv4-mapped addresses are not recommended. The \fBgetaddrinfo\fR(3SOCKET)
 539 function is preferred over \fBgetipnodebyaddr()\fR because it allows
 540 applications to lookup IPv4 and IPv6 addresses without relying on IPv4-mapped
 541 addresses.
 542 .sp
 543 .LP
 544 The form for an address of type \fBAF_INET\fR is a \fBstruct in_addr\fR defined
 545 in <\fBnetinet/in.h\fR>. The form for an address of type \fBAF_INET6\fR is a
 546 \fBstruct in6_addr\fR, also defined in <\fBnetinet/in.h\fR>. The functions
 547 described in \fBinet_ntop\fR(3SOCKET) and \fBinet_pton\fR(3SOCKET) that are
 548 illustrated in the EXAMPLES section are helpful in constructing and
 549 manipulating addresses in either of these forms.
```

```
**********************************************************
   85863 Tue Sep 10 18:35:21 2013
new/usr/src/man/man3tecla/gl_get_line.3tecla
4023 - Typo in file(1) manpage and various others
**********************************************************
```

```
'\" te
.\" Copyright (c) 2000, 2001, 2002, 2003, 2004 by Martin C. Shepherd.
.\" All Rights Reserved.
.\" 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) 2007, Sun Microsystems, Inc. All Rights Reserved.
.TH GL_GET_LINE 3TECLA "Sep 10, 2013"
.TH GL_GET_LINE 3TECLA "Nov 28, 2007"
.SH NAME
gl_get_line, new_GetLine, del_GetLine, gl_customize_completion,
gl_change_terminal, gl_configure_getline, gl_load_history, gl_save_history,
gl_group_history, gl_show_history, gl_watch_fd, gl_inactivity_timeout,
gl_terminal_size, gl_set_term_size, gl_resize_history, gl_limit_history,
gl_clear_history, gl_toggle_history, gl_lookup_history, gl_state_of_history,
gl_range_of_history, gl_size_of_history, gl_echo_mode, gl_replace_prompt,
gl_prompt_style, gl_ignore_signal, gl_trap_signal, gl_last_signal,
gl_completion_action, gl_register_action, gl_display_text, gl_return_status,
gl_error_message, gl_catch_blocked, gl_list_signals, gl_bind_keyseq,
gl_erase_terminal, gl_automatic_history, gl_append_history, gl_query_char,
gl_read_char \- allow the user to compose an input line
.SH SYNOPSIS
.LP
.nf
cc [ \fIflag\fR\&.\|.\|. ] \fIfile\fR\&.\|.\|. \fB-ltecla\fR [ \fIlibrary\fR\&.\
#include <stdio.h>
#include <libtecla.h>

\fBGetLine *\fR\fBnew_GetLine\fR(\fBsize_t\fR \fIlinelen\fR, \fBsize_t\fR \fIhis
.fi

.LP
.nf
\fBGetLine *\fR\fBdel_GetLine\fR(\fBGetLine *\fR\fIgl\fR);
.fi

.LP
.nf
\fBchar *\fR\fBgl_get_line\fR(\fBGetLine *\fR\fIgl\fR, \fBconst char *\fR\fIprom
     \fBconst char *\fR\fIstart_line\fR, \fBint\fR \fIstart_pos\fR);
.fi
```



```
.LP
.nf
\fBint\fR \fBgl_query_char\fR(\fBGetLine *\fR\fIgl\fR, \fBconst char *\fR\fIprom
.fi

.LP
.nf
\fBint\fR \fBgl_read_char\fR(\fBGetLine *\fR\fIgl\fR);
.fi

.LP
.nf
\fBint\fR \fBgl_customize_completion\fR(\fBGetLine *\fR\fIgl\fR, \fBvoid *\fR\fI
     \fBCplMatchFn *\fR\fImatch_fn\fR);
.fi

.LP
.nf
\fBint\fR \fBgl_change_terminal\fR(\fBGetLine *\fR\fIgl\fR, \fBFILE *\fR\fIinput
     \fBFILE *\fR\fIoutput_fp\fR, \fBconst char *\fR\fIterm\fR);
.fi

.LP
.nf
\fBint\fR \fBgl_configure_getline\fR(\fBGetLine *\fR\fIgl\fR, \fBconst char *\fR
     \fBconst char *\fR\fIapp_file\fR,\ \fBconst char *\fR\fIuser_file\fR);
.fi

.LP
.nf
\fBint\fR \fBgl_bind_keyseq\fR(\fBGetLine *\fR\fIgl\fR, \fBGlKeyOrigin\fR \fIori
     \fBconst char *\fR\fIkeyseq\fR, \fBconst char *\fR\fIaction\fR);
.fi

.LP
.nf
\fBint\fR \fBgl_save_history\fR(\fBGetLine *\fR\fIgl\fR, \fBconst char *\fR\fIfi
     \fBconst char *\fR\fIcomment\fR, \fBint\fR \fImax_lines\fR);
.fi

.LP
.nf
\fBint\fR \fBgl_load_history\fR(\fBGetLine *\fR\fIgl\fR, \fBconst char *\fR\fIfi
     \fBconst char *\fR\fIcomment\fR);
.fi

.LP
.nf
\fBint\fR \fBgl_watch_fd\fR(\fBGetLine *\fR\fIgl\fR, \fBint\fR \fIfd\fR, \fBGlFd
     \fBGlFdEventFn *\fR\fIcallback\fR, \fBvoid *\fR\fIdata\fR);
.fi

.LP
.nf
\fBint\fR \fBgl_inactivity_timeout\fR(\fBGetLine *\fR\fIgl\fR, \fBGlTimeoutFn *\
     \fBvoid *\fR\fIdata\fR, \fBunsigned long\fR \fIsec\fR, \fBunsigned long\fR
.fi

.LP
.nf
\fBint\fR \fBgl_group_history\fR(\fBGetLine *\fR\fIgl\fR, \fBunsigned\fR \fIstre
.fi

.LP
.nf
```

```
\fBint\fR \fBgl_show_history\fR(\fBGetLine *\fR\fIgl\fR, \fBFILE *\fR\fIfp\fR, \
     \fBint\fR \fIall_groups\fR, \fBint\fR \fImax_lines\fR);
.fi

.LP
.nf
\fBint\fR \fBgl_resize_history\fR(\fBGetLine *\fR\fIgl\fR, \fBsize_t\fR \fIbufsi
.fi

.LP
.nf
\fBvoid\fR \fBgl_limit_history\fR(\fBGetLine *\fR\fIgl\fR, \fBint\fR \fImax_line
.fi

.LP
.nf
\fBvoid\fR \fBgl_clear_history\fR(\fBGetLine *\fR\fIgl\fR, \fBint\fR \fIall_grou
.fi

.LP
.nf
\fBvoid\fR \fBgl_toggle_history\fR(\fBGetLine *\fR\fIgl\fR, \fBint\fR \fIenable\
.fi

.LP
.nf
\fBGlTerminalSize\fR \fBgl_terminal_size\fR(\fBGetLine *\fR\fIgl\fR, \fBint\fR \
     \fBint\fR \fIdef_nline\fR);
.fi

.LP
.nf
\fBint\fR \fBgl_set_term_size\fR(\fBGetLine *\fR\fIgl\fR, \fBint\fR \fIncolumn\f
.fi

.LP
.nf
\fBint\fR \fBgl_lookup_history\fR(\fBGetLine *\fR\fIgl\fR, \fBunsigned long\fR \
     \fBGlHistoryLine *\fR\fIhline\fR);
.fi

.LP
.nf
\fBvoid\fR \fBgl_state_of_history\fR(\fBGetLine *\fR\fIgl\fR, \fBGlHistoryState
.fi

.LP
.nf
\fBvoid\fR \fBgl_range_of_history\fR(\fBGetLine *\fR\fIgl\fR, \fBGlHistoryRange
.fi

.LP
.nf
\fBvoid\fR \fBgl_size_of_history\fR(\fBGetLine *\fR\fIgl\fR, \fBGlHistorySize *\
.fi

.LP
.nf
\fBvoid\fR \fBgl_echo_mode\fR(\fBGetLine *\fR\fIgl\fR, \fBint\fR \fIenable\fR);
.fi

.LP
.nf
\fBvoid\fR \fBgl_replace_prompt\fR(\fBGetLine *\fR\fIgl\fR, \fBconst char *\fR\f
.fi
```



```
.LP
.nf
\fBvoid\fR \fBgl_prompt_style\fR(\fBGetLine *\fR\fIgl\fR, \fBGlPromptStyle\fR \f
.fi

.LP
.nf
\fBint\fR \fBgl_ignore_signal\fR(\fBGetLine *\fR\fIgl\fR, \fBint\fR \fIsigno\fR)
.fi

.LP
.nf
\fBint\fR \fBgl_trap_signal\fR(\fBGetLine *\fR\fIgl\fR, \fBint\fR \fIsigno\fR, \
     \fBGlAfterSignal\fR \fIafter\fR, \fBint\fR \fIerrno_value\fR);
.fi

.LP
.nf
\fBint\fR \fBgl_last_signal\fR(\fBGetLine *\fR\fIgl\fR);
.fi

.LP
.nf
\fBint\fR \fBgl_completion_action\fR(\fBGetLine *\fR\fIgl\fR, \fBvoid *\fR\fIdat
     \fBCplMatchFn *\fR\fImatch_fn\fR, \fBint\fR \fIlist_only\fR, \fBconst char
     \fBconst char *\fR\fIkeyseq\fR);
.fi

.LP
.nf
\fBint\fR \fBgl_register_action\fR(\fBGetLine *\fR\fIgl\fR, \fBvoid *\fR\fIdata\
     \fBconst char *\fR\fIname\fR, \fBconst char *\fR\fIkeyseq\fR);
.fi

.LP
.nf
\fBint\fR \fBgl_display_text\fR(\fBGetLine *\fR\fIgl\fR, \fBint\fR \fIindentatio
     \fBconst char *\fR\fIprefix\fR, \fBconst char *\fR\fIsuffix\fR, \fBint\fR \
     \fBint\fR \fIdef_width\fR, \fBint\fR \fIstart\fR, \fBconst char *\fR\fIstri
.fi

.LP
.nf
\fBGlReturnStatus\fR \fBgl_return_status\fR(\fBGetLine *\fR\fIgl\fR);
.fi

.LP
.nf
\fBconst char *\fR\fBgl_error_message\fR(\fBGetLine *\fR\fIgl\fR, \fBchar *\fR\f
.fi

.LP
.nf
\fBvoid\fR \fBgl_catch_blocked\fR(\fBGetLine *\fR\fIgl\fR);
.fi

.LP
.nf
\fBint\fR \fBgl_list_signals\fR(\fBGetLine *\fR\fIgl\fR, \fBsigset_t *\fR\fIset\
.fi

.LP
.nf
\fBint\fR \fBgl_append_history\fR(\fBGetLine *\fR\fIgl\fR, \fBconst char *\fR\fI
.fi
```

```
.LP
.nf
\fBint\fR \fBgl_automatic_history\fR(\fBGetLine *\fR\fIgl\fR, \fBint\fR \fIenabl
.fi

.LP
.nf
\fBint\fR \fBgl_erase_terminal\fR(\fBGetLine *\fR\fIgl\fR);
.fi

.SH DESCRIPTION
.sp
.LP
The \fBgl_get_line()\fR function is part of the \fBlibtecla\fR(3LIB) library.
If the user is typing at a terminal, each call prompts them for an line of
input, then provides interactive editing facilities, similar to those of the
UNIX \fBtcsh\fR shell. In addition to simple command-line editing, it supports
recall of previously entered command lines, TAB completion of file names, and
in-line wild-card expansion of filenames. Documentation of both the user-level
command-line editing features and all user configuration options can be found
on the \fBtecla\fR(5) manual page.
.SS "An Example"
.sp
.LP
The following shows a complete example of how to use the \fBgl_get_line()\fR
function to get input from the user:
.sp
.in +2
.nf
#include <stdio.h>
#include <locale.h>
#include <libtecla.h>

int main(int argc, char *argv[])
{
  char *line;    /* The line that the user typed */
  GetLine *gl;   /* The gl_get_line() resource object */

  setlocale(LC_CTYPE, ""); /* Adopt the user's choice */
                           /* of character set. */

  gl = new_GetLine(1024, 2048);
  if(!gl)
    return 1;
  while((line=gl_get_line(gl, "$ ", NULL, -1)) != NULL &&
        strcmp(line, "exit\en") != 0)
    printf("You typed: %s\en", line);

  gl = del_GetLine(gl);
  return 0;
}
.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"
.sp
.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"
.sp
.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:
```

```
391 .sp
392 .ne 2
393 .na
394 \fB\fBGLR_NEWLINE\fR\fR
395 .ad
396 .RS 15n
397 The last call to \fBgl_get_line()\fR successfully returned a completed input
398 line.
399 .RE

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

413 .sp
414 .ne 2
415 .na
416 \fB\fBGLR_SIGNAL\fR\fR
417 .ad
418 .RS 15n
419 A signal was caught by \fBgl_get_line()\fR that had an after-signal disposition
420 of \fBGLS_ABORT\fR. See \fBgl_trap_signal()\fR.
421 .RE

423 .sp
424 .ne 2
425 .na
426 \fB\fBGLR_TIMEOUT\fR\fR
427 .ad
428 .RS 15n
429 The inactivity timer expired while \fBgl_get_line()\fR was waiting for input,
430 and the timeout callback function returned \fBGLTO_ABORT\fR. See
431 \fBgl_inactivity_timeout()\fR for information about timeouts.
432 .RE

434 .sp
435 .ne 2
436 .na
437 \fB\fBGLR_FDABORT\fR\fR
438 .ad
439 .RS 15n
440 An application I/O callback returned \fBGLFD_ABORT\fR. Ssee
441 \fBgl_watch_fd()\fR.
442 .RE

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

455 .sp
456 .ne 2
```



```
457 .na
458 \fB\fBGLR_ERROR\fR\fR
459 .ad
460 .RS 15n
461 An unexpected error caused \fBgl_get_line()\fR to abort (consult \fBerrno\fR
462 and/or \fBgl_error_message()\fR for details.
463 .RE

465 .sp
466 .LP
467 When \fBgl_return_status()\fR returns \fBGLR_ERROR\fR and the value of
468 \fBerrno\fR is not sufficient to explain what happened, you can use the
469 \fBgl_error_message()\fR function to request a description of the last error
470 that occurred.
471 .sp
472 .LP
473 The return value of \fBgl_error_message()\fR is a pointer to the message that
474 occurred. If the \fIbuff\fR argument is \fINULL\fR, this will be a pointer to a
475 buffer within \fIgl\fR whose value will probably change on the next call to any
476 function associated with \fBgl_get_line()\fR. Otherwise, if a non-null
477 \fIbuff\fR argument is provided, the error message, including a '\e0'
478 terminator, will be written within the first \fIn\fR elements of this buffer,
479 and the return value will be a pointer to the first element of this buffer. If
480 the message will not fit in the provided buffer, it will be truncated to fit.
481 .SS "Optional Prompt Formatting"
482 .sp
483 .LP
484 Whereas by default the prompt string that you specify is displayed literally
485 without any special interpretation of the characters within it, the
486 \fBgl_prompt_style()\fR function can be used to enable optional formatting
487 directives within the prompt.
488 .sp
489 .LP
490 The \fIstyle\fR argument, which specifies the formatting style, can take any of
491 the following values:
492 .sp
493 .ne 2
494 .na
495 \fB\fBGL_FORMAT_PROMPT\fR\fR
496 .ad
497 .RS 21n
498 In this style, the formatting directives described below, when included in
499 prompt strings, are interpreted as follows:
500 .sp
501 .ne 2
502 .na
503 \fB\fB%B\fR\fR
504 .ad
505 .RS 6n
506 Display subsequent characters with a bold font.
507 .RE

509 .sp
510 .ne 2
511 .na
512 \fB\fB%b\fR\fR
513 .ad
514 .RS 6n
515 Stop displaying characters with the bold font.
516 .RE

518 .sp
519 .ne 2
520 .na
521 \fB\fB%F\fR\fR
522 .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"
.sp
.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
```

```
655 .LP
656 The \fBgl_configure_getline()\fR function allows the configuration commands
657 that would normally be read from a user's \fB~/.teclarc\fR file, to be read
658 from any or none of, a string, an application specific configuration file,
659 and/or a user-specific configuration file. If this function is called before
660 the first call to \fBgl_get_line()\fR, the default behavior of reading
661 \fB~/.teclarc\fR on the first call to \fBgl_get_line()\fR is disabled, so all
662 configurations must be achieved using the configuration sources specified with
663 this function.
664 .sp
665 .LP
666 If \fIapp_string\fR != \fINULL\fR, then it is interpreted as a string
667 containing one or more configuration commands, separated from each other in the
668 string by embedded newline  characters. If \fIapp_file\fR != \fINULL\fR then it
669 is interpreted as the full pathname of an application-specific configuration
670 file. If user_file != \fINULL\fR then it is interpreted as the full path name
671 of a user-specific configuration file, such as \fB~/.teclarc\fR. For example,
672 in the call
673 .sp
674 .in +2
675 .nf
676 gl_configure_getline(gl, "edit-mode vi \en nobeep",
676 gl_configure_getline(gl, "edit-mode vi \n nobeep",
677                      "/usr/share/myapp/teclarc", "~/.teclarc");
678 .fi
679 .in -2

681 .sp
682 .LP
683 The \fIapp_string\fR argument causes the calling application to start in
684 \fBvi\fR(1) edit-mode, instead of the default \fBemacs\fR mode, and turns off
685 the use of the terminal bell by the library. It then attempts to read
686 system-wide configuration commands from an optional file called
687 \fB/usr/share/myapp/teclarc\fR, then finally reads user-specific configuration
688 commands from an optional \fB\&.teclarc\fR file in the user's home directory.
689 Note that the arguments are listed in ascending order of priority, with the
690 contents of \fIapp_string\fR being potentially over riden by commands in
691 \fIapp_file\fR, and commands in \fIapp_file\fR potentially being overriden by
692 commands in \fIuser_file\fR.
693 .sp
694 .LP
695 You can call this function as many times as needed, the results being
696 cumulative, but note that copies of any file names specified with the
697 \fIapp_file\fR and \fIuser_file\fR arguments are recorded internally for
698 subsequent use by the read-init-files key-binding function, so if you plan to
699 call this function multiple times, be sure that the last call specifies the
700 filenames that you want re-read when the user requests that the configuration
701 files be re-read.
702 .sp
703 .LP
704 Individual key sequences can also be bound and unbound using the
705 \fBgl_bind_keyseq()\fR function. The \fIorigin\fR argument specifies the
706 priority of the binding, according to whom it is being established for, and
707 must be one of the following two values.
708 .sp
709 .ne 2
710 .na
711 \fB\fBGL_USER_KEY\fR\fR
712 .ad
713 .RS 15n
714 The user requested this key-binding.
715 .RE

717 .sp
718 .ne 2
719 .na
```



```
720 \fB\fBGL_APP_KEY\fR\fR
721 .ad
722 .RS 15n
723 This is a default binding set by the application.
724 .RE

726 .sp
727 .LP
728 When both user and application bindings for a given key sequence have been
729 specified, the user binding takes precedence. The application's binding is
730 subsequently reinstated if the user's binding is later unbound with either
731 another call to this function, or a call to \fBgl_configure_getline()\fR.
732 .sp
733 .LP
734 The \fIkeyseq\fR argument specifies the key sequence to be bound or unbound,
735 and is expressed in the same way as in a \fB~/.teclarc\fR configuration file.
736 The \fIaction\fR argument must either be a string containing the name of the
737 action to bind the key sequence to, or it must be \fINULL\fR or \fB""\fR to
738 unbind the key sequence.
739 .SS "Customized Word Completion"
740 .sp
741 .LP
742 If in your application you would like to have TAB completion complete other
743 things in addition to or instead of filenames, you can arrange this by
744 registering an alternate completion callback function with a call to the
745 \fBgl_customize_completion()\fR function.
746 .sp
747 .LP
748 The \fIdata\fR argument provides a way for your application to pass arbitrary,
749 application-specific information to the callback function. This is passed to
750 the callback every time that it is called. It might for example point to the
751 symbol table from which possible completions are to be sought. The
752 \fImatch_fn\fR argument specifies the callback function to be called. The
753 \fICplMatchFn\fR function type is defined in <\fBlibtecla.h\fR>, as is a
754 \fBCPL_MATCH_FN()\fR macro that you can use to declare and prototype callback
755 functions. The declaration and responsibilities of callback functions are
756 described in depth on the \fBcpl_complete_word\fR(3TECLA) manual page.
757 .sp
758 .LP
759 The callback function is responsible for looking backwards in the input line
760 from the point at which the user pressed TAB, to find the start of the word
761 being completed. It then must lookup possible completions of this word, and
762 record them one by one in the \fBWordCompletion\fR object that is passed to it
763 as an argument, by calling the \fBcpl_add_completion()\fR function. If the
764 callback function wants to provide filename completion in addition to its own
765 specific completions, it has the option of itself calling the builtin filename
766 completion callback. This also is documented on the
767 \fBcpl_complete_word\fR(3TECLA) manual page.
768 .sp
769 .LP
770 If you would like \fBgl_get_line()\fR to return the current input line when a
771 successful completion is been made, you can arrange this when you call
772 \fBcpl_add_completion()\fR by making the last character of the continuation
773 suffix a newline character. The input line will be updated to display the
774 completion, together with any contiuation suffix up to the newline character,
775 and \fBgl_get_line()\fR will return this input line.
776 .sp
777 .LP
778 If your callback function needs to write something to the terminal, it must
779 call \fBgl_normal_io()\fR before doing so. This will start a new line after the
780 input line that is currently being edited, reinstate normal terminal I/O, and
781 notify \fBgl_get_line()\fR that the input line will need to be redrawn when the
782 callback returns.
783 .SS "Adding Completion Actions"
784 .sp
785 .LP
```

```
786 In the previous section the ability to customize the behavior of the only
787 default completion action, complete-word, was described. In this section the
788 ability to install additional action functions, so that different types of word
789 completion can be bound to different key sequences, is described. This is
790 achieved by using the \fBgl_completion_action()\fR function.
791 .sp
792 .LP
793 The \fIdata\fR and \fImatch_fn\fR arguments are as described on the
794 \fBcpl_complete_word\fR(3TECLA) manual page, and specify the callback function
795 that should be invoked to identify possible completions. The \fIlist_only\fR
796 argument determines whether the action that is being defined should attempt to
797 complete the word as far as possible in the input line before displaying any
798 possible ambiguous completions, or whether it should simply display the list of
799 possible completions without touching the input line. The former option is
800 selected by specifying a value of 0, and the latter by specifying a value of 1.
801 The \fIname\fR argument specifies the name by which configuration files and
802 future invocations of this function should refer to the action. This must
803 either be the name of an existing completion action to be changed, or be a new
804 unused name for a new action. Finally, the \fIkeyseq\fR argument specifies the
805 default key sequence to bind the action to. If this is \fINULL\fR, no new key
806 sequence will be bound to the action.
807 .sp
808 .LP
809 Beware that in order for the user to be able to change the key sequence that is
810 bound to actions that are installed in this manner, you shouldcall
811 \fBgl_completion_action()\fR to install a given action for the first time
812 between calling \fBnew_GetLine()\fR and the first call to \fBgl_get_line()\fR.
813 Otherwise, when the user's configuration file is read on the first call to
814 \fBgl_get_line()\fR, the name of the your additional action will not be known,
815 and any reference to it in the configuration file will generate an error.
816 .sp
817 .LP
818 As discussed for \fBgl_customize_completion()\fR, if your callback function
819 needs to write anything to the terminal, it must call \fBgl_normal_io()\fR
820 before doing so.
821 .SS "Defining Custom Actions"
822 .sp
823 .LP
824 Although the built-in key-binding actions are sufficient for the needs of most
825 applications, occasionally a specialized application may need to define one or
826 more custom actions, bound to application-specific key sequences. For example,
827 a sales application would benefit from having a key sequence that displayed the
828 part name that corresponded to a part number preceding the cursor. Such a
829 feature is clearly beyond the scope of the built-in action functions. So for
830 such special cases, the \fBgl_register_action()\fR function is provided.
831 .sp
832 .LP
833 The \fBgl_register_action()\fR function lets the application register an
834 external function, \fIfn\fR, that will thereafter be called whenever either the
835 specified key sequence, \fIkeyseq\fR, is entered by the user, or the user
836 enters any other key sequence that the user subsequently binds to the specified
837 action name, \fIname\fR, in their configuration file. The \fIdata\fR argument
838 can be a pointer to anything that the application wants to have passed to the
839 action function, \fIfn\fR, whenever that function is invoked.
840 .sp
841 .LP
842 The action function, \fIfn\fR, should be declared using the
843 \fBGL_ACTION_FN()\fR macro, which is defined in <\fBlibtecla.h\fR>.
844 .sp
845 .in +2
846 .nf
847 #define GL_ACTION_FN(fn) GlAfterAction (fn)(GetLine *gl, \e
848                         void *data, int count, size_t curpos, \e
849                         const char *line)
850 .fi
851 .in -2
```



```
853 .sp
854 .LP
855 The \fIgl\fR and \fIdata\fR arguments are those that were previously passed to
856 \fBgl_register_action()\fR when the action function was registered. The
857 \fIcount\fR argument is a numeric argument which the user has the option of
858 entering using the digit-argument action, before invoking the action. If the
859 user does not enter a number, then the \fIcount\fR argument is set to 1.
860 Nominally this argument is interpreted as a repeat count, meaning that the
861 action should be repeated that many times. In practice however, for some
862 actions a repeat count makes little sense. In such cases, actions can either
863 simply ignore the \fIcount\fR argument, or use its value for a different
864 purpose.
865 .sp
866 .LP
867 A copy of the current input line is passed in the read-only \fIline\fR
868 argument. The current cursor position within this string is given by the index
869 contained in the \fIcurpos\fR argument. Note that direct manipulation of the
870 input line and the cursor position is not permitted because the rules dictated
871 by various modes (such as \fBvi\fR mode versus \fBemacs\fR mode, no-echo mode,
872 and insert mode versus overstrike mode) make it too complex for an application
873 writer to write a conforming editing action, as well as constrain future
874 changes to the internals of \fBgl_get_line()\fR. A potential solution to this
875 dilemma would be to allow the action function to edit the line using the
876 existing editing actions. This is currently under consideration.
877 .sp
878 .LP
879 If the action function wishes to write text to the terminal without this
880 getting mixed up with the displayed text of the input line, or read from the
881 terminal without having to handle raw terminal I/O, then before doing either of
882 these operations, it must temporarily suspend line editing by calling the
883 \fBgl_normal_io()\fR function. This function flushes any pending output to the
884 terminal, moves the cursor to the start of the line that follows the last
885 terminal line of the input line, then restores the terminal to a state that is
886 suitable for use with the C \fBstdio\fR facilities. The latter includes such
887 things as restoring the normal mapping of \en to \er\en, and, when in server
888 mode, restoring the normal blocking form of terminal I/O. Having called this
889 function, the action function can read from and write to the terminal without
890 the fear of creating a mess. It is not necessary for the action function to
891 restore the original editing environment before it returns. This is done
892 automatically by \fBgl_get_line()\fR after the action function returns. The
893 following is a simple example of an action function which writes the sentence
894 "Hello world" on a new terminal line after the line being edited. When this
895 function returns, the input line is redrawn on the line that follows the "Hello
896 world" line, and line editing resumes.
897 .sp
898 .in +2
899 .nf
900 static GL_ACTION_FN(say_hello_fn)
901 {
902     if(gl_normal_io(gl))   /* Temporarily suspend editing */
903         return GLA_ABORT;
904     printf("Hello world\en");
905     return GLA_CONTINUE;
906 }
_____unchanged_portion_omitted_
```

```
**********************************************************
   17944 Tue Sep 10 18:35:21 2013
new/usr/src/man/man5/acl.5
4023 - Typo in file(1) manpage and various others
**********************************************************
   1 '\" te
   2 .\" Copyright (c) 2008, Sun Microsystems, Inc. All Rights Reserved.
   3 .\" The contents of this file are subject to the terms of the Common Development
   4 .\" You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE or http:
   5 .\" When distributing Covered Code, include this CDDL HEADER in each file and in
   6 .TH ACL 5 "Sep 10, 2013"
   6 .TH ACL 5 "Sep 29, 2008"
   7 .SH NAME
   8 acl \- Access Control Lists
   9 .SH DESCRIPTION
  10 .sp
  11 .LP
  12 Access control lists (ACLs) are discretionary access control mechanisms that
  13 grant and deny access to files and directories. Two different ACL models are
  14 supported in the Solaris release:POSIX-draft ACLs and NFSv4 ACLs.
  15 .sp
  16 .LP
  17 The older, POSIX-draft model is supported by the UFS file system. This model is
  18 based on a withdrawn ACL POSIX specification that was never standardized. It
  19 was subsequently withdrawn by the POSIX committee.
  20 .sp
  21 .LP
  22 The other model is based on the standards of the NFSv4 working group and is an
  23 approved standard from the Internet Engineering Task Force (IETF). The ZFS file
  24 system uses the NFSv4 model, and provides richer semantics and finer grained
  25 permission capabilities than the POSIX-draft model.
  26 .SS "\fBPOSIX\fR-draft \fBACL\fRs"
  27 .sp
  28 .LP
  29 POSIX-draft ACLs provide an alternative security mechanism to basic UNIX file
  30 permissions in the Solaris release. Their purpose is to further restrict access
  31 to files and directories or to extend permissions to a particular user. ACLs
  32 can be used to change the permissions for the standard owner, group and other
  33 class bits of a file's mode. ACLs can give additional users and groups access
  34 to the file. A directory can also have a special kind of ACL called a
  35 \fBdefault\fR ACL, which defines ACL entries to be inherited by descendents of
  36 the directory. POSIX-draft ACLs have an ACL entry called \fBmask\fR. The mask
  37 defines the maximum permissions that can be granted to additional user and
  38 group entries. Whenever a file is created or its mode is changed by
  39 \fBchmod\fR(1) or \fBchmod\fR(2), the mask is recomputed. It is recomputed to
  40 be the group permission defined in the mode passed to \fBchmod\fR(2).
  41 .sp
  42 .LP
  43 The POSIX-draft ACL model uses the standard \fBrwx\fR model of traditional UNIX
  44 permissions.
  45 .sp
  46 .LP
  47 An ACL is represented as follows:
  48 .sp
  49 .in +2
  50 .nf
  51 \fIacl_entry\fR[,\fIacl_entry\fR]...
  52 .fi
  53 .in -2
  54 .sp

  56 .sp
  57 .LP
  58 Each \fIacl_entry\fR contains one ACL entry. An ACL entry is represented by two
  59 or three colon-separated(\fB:\fR) fields.
  60 .sp
```



```
  61 .ne 2
  62 .na
  63 \fB\fIuser\fR:[\fIuid\fR]:\fIperms\fR\fR
  64 .ad
  65 .RS 21n
  66 If \fIuid\fR blank, it represents the file owner.
  67 .RE

  69 .sp
  70 .ne 2
  71 .na
  72 \fB\fIgroup\fR:[\fIgid\fR]:\fIperms\fR\fR
  73 .ad
  74 .RS 21n
  75 If \fIgid\fR is blank, it represents the owning group.
  76 .RE

  78 .sp
  79 .ne 2
  80 .na
  81 \fB\fIother\fR:\fIperms\fR\fR
  82 .ad
  83 .RS 21n
  84 Represents the file other class.
  85 .RE

  87 .sp
  88 .ne 2
  89 .na
  90 \fB\fImask\fR:\fIperms\fR\fR
  91 .ad
  92 .RS 21n
  93 Defines the \fBMAX\fR permission to hand out.
  94 .RE

  96 .sp
  97 .LP
  98 For example to give user \fBjoe\fR read and write permissions, the ACL entry is
  99 specified as:
 100 .sp
 101 .in +2
 102 .nf
 103 user:joe:rw-
 104 .fi
 105 .in -2
 106 .sp

 108 .SS "\fBNFS\fRv4 \fBACL\fRs"
 109 .sp
 110 .LP
 111 NFSv4 ACL model is based loosely on the Windows NT ACL model. NFSv4 ACLs
 112 provide a much richer ACL model than POSIX-draft ACLs.
 113 .sp
 114 .LP
 115 The major differences between NFSv4 and POSIX-draft ACLs are as follows:
 116 .RS +4
 117 .TP
 118 .ie t \(bu
 119 .el o
 120 NFSv4 ACLs provide finer grained permissions than the \fBrwx\fR model.
 121 .RE
 122 .RS +4
 123 .TP
 124 .ie t \(bu
 125 .el o
 126 NFSv4 ACLs allow for both \fBALLOW\fR and \fBDENY\fR entries.
```

```
.RE
.RS +4
.TP
.ie t \(bu
.el o
NFSv4 ACLs provide a rich set of inheritance semantics. POSIX ACLs also have
inheritance, but with the NFSv4 model you can control the following inheritance
features:
.RS +4
.TP
.ie t \(bu
.el o
Whether inheritance cascades to both files and directories or only to files or
directories.
.RE
.RS +4
.TP
.ie t \(bu
.el o
In the case of directories, you can indicate whether inheritance is applied to
the directory itself, to just one level of subdirectories, or cascades to all
subdirectories of the directory.
.RE
.RE
.RS +4
.TP
.ie t \(bu
.el o
NFSv4 ACLs provide a mechanism for hooking into a system’s audit trail.
Currently, Solaris does not support this mechanism.
.RE
.RS +4
.TP
.ie t \(bu
.el o
NFSv4 ACLs enable adminstrators to specify the order in which ACL entries are
checked. With POSIX-draft ACLs the file system reorders ACL entries into a well
defined, strict access, checking order.
.RE
.sp
.LP
POSIX-draft ACL semantics can be achieved with NFSv4 ACLs. However, only some
NFSv4 ACLs can be translated to equivalent POSIX-draft ACLs.
.sp
.LP
Permissions can be specified in three different \fBchmod\fR ACL formats:
verbose, compact, or positional. The verbose format uses words to indicate that
the permissions are separated with a forward slash (\fB/\fR) character. Compact
format uses the permission letters and positional format uses the permission
letters or the hypen (\fB-\fR) to identify no permissions.
.sp
.LP
The permissions for verbose mode and their abbreviated form in parentheses for
compact and positional mode are described as follows:
.sp
.ne 2
.na
\fBread_data (\fBr\fR)\fR
.ad
.RS 24n
Permission to read the data of the file
.RE

.sp
.ne 2
.na
```



```
\fBlist_directory (\fBr\fR)\fR
.ad
.RS 24n
Permission to list the contents of a directory.
.RE

.sp
.ne 2
.na
\fBwrite_data (\fBw\fR)\fR
.ad
.RS 24n
Permission to modify a file’s data anywhere in the file’s offset range. This
includes the ability to grow the file or write to any arbitrary offset.
.RE

.sp
.ne 2
.na
\fBadd_file (\fBw\fR)\fR
.ad
.RS 24n
Permission to add a new file to a directory.
.RE

.sp
.ne 2
.na
\fBappend_data (\fBp\fR)\fR
.ad
.RS 24n
The ability to modify the file’s data, but only starting at EOF. Currently,
this permission is not supported.
.RE

.sp
.ne 2
.na
\fBadd_subdirectory (\fBp\fR)\fR
.ad
.RS 24n
Permission to create a subdirectory to a directory.
.RE

.sp
.ne 2
.na
\fBread_xattr (\fBR\fR)\fR
.ad
.RS 24n
The ability to read the extended attributes of a file or do a lookup in the
extended attributes directory.
.RE

.sp
.ne 2
.na
\fBwrite_xattr (\fBW\fR)\fR
.ad
.RS 24n
The ability to create extended attributes or write to the extended attributes
directory.
.RE

.sp
.ne 2
```

```
.na
\fBexecute (\fBx\fR)\fR
.ad
.RS 24n
Permission to execute a file.
.RE

.sp
.ne 2
.na
\fBread_attributes (\fBa\fR)\fR
.ad
.RS 24n
The ability to read basic attributes (non-ACLs) of a file. Basic attributes are
considered to be the stat level attributes. Allowing this access mask bit means
that the entity can execute \fBls\fR(1) and \fBstat\fR(2).
.RE

.sp
.ne 2
.na
\fBwrite_attributes (\fBA\fR)\fR
.ad
.RS 24n
Permission to change the times associated with a file or directory to an
arbitrary value.
.RE

.sp
.ne 2
.na
\fBdelete (\fBd\fR)\fR
.ad
.RS 24n
Permission to delete the file.
.RE

.sp
.ne 2
.na
\fBdelete_child (\fBD\fR)\fR
.ad
.RS 24n
Permission to delete a file within a directory.
.RE

.sp
.ne 2
.na
\fBread_acl (\fBc\fR)\fR
.ad
.RS 24n
Permission to read the ACL.
.RE

.sp
.ne 2
.na
\fBwrite_acl (\fBC\fR)\fR
.ad
.RS 24n
Permission to write the ACL or the ability to execute \fBchmod\fR(1) or
\fBsetfacl\fR(1).
.RE

.sp
```



```
.ne 2
.na
\fBwrite_owner (\fBo\fR)\fR
.ad
.RS 24n
Permission to change the owner or the ability to execute \fBchown\fR(1) or
\fBchgrp\fR(1).
.RE

.sp
.ne 2
.na
\fBsynchronize (\fBs\fR)\fR
.ad
.RS 24n
Permission to access a file locally at the server with synchronous reads and
writes. Currently, this permission is not supported.
.RE

.sp
.LP
The following inheritance flags are supported by NFSv4:
.sp
.ne 2
.na
\fBfile_inherit (\fBf\fR)\fR
.ad
.RS 26n
Inherit to all newly created files in a directory.
.RE

.sp
.ne 2
.na
\fBdir_inherit (\fBd\fR)\fR
.ad
.RS 26n
Inherit to all newly created directories in a directory.
.RE

.sp
.ne 2
.na
\fBinherit_only (\fBi\fR)\fR
.ad
.RS 26n
Placed on a directory, but does not apply to the directory itself, only to
newly created created files and directories. This flag requires file_inherit
and or dir_inherit to indicate what to inherit.
.RE

.sp
.ne 2
.na
\fBno_propagate (\fBn\fR)\fR
.ad
.RS 26n
Placed on directories and indicates that ACL entries should only be inherited
one level of the tree. This flag requires file_inherit and or dir_inherit to
indicate what to inherit.
.RE

.sp
.ne 2
.na
\fBsuccessful_access (\fBS)\fR)\fR
```

```
391 .ad
392 .RS 26n
393 Indicates if an alarm or audit record should be initiated upon successful
394 accesses. Used with audit/alarm ACE types.
395 .RE

397 .sp
398 .ne 2
399 .na
400 \fBfailed_access (\fBF\fR)\fR
401 .ad
402 .RS 26n
403 Indicates if an alarm or audit record should be initiated when access fails.
404 Used with audit/alarm ACE types.
405 .RE

407 .sp
408 .ne 2
409 .na
410 \fBinherited (\fBI\fR)\fR
411 .ad
412 .RS 26n
413 ACE was inherited.
414 .RE

416 .sp
417 .ne 2
418 .na
419 \fB\fB-\fR\fR
420 .ad
421 .RS 26n
422 No permission granted.
423 .RE

425 .sp
426 .LP
427 An NFSv4 ACL is expressed using the following syntax:
428 .sp
429 .in +2
430 .nf
431 \fIacl_entry\fR[,\fIacl_entry\fR]...

433     owner@:<perms>[:inheritance flags]:<allow|deny>
434     group@:<perms>[:inheritance flags]:<allow|deny>
435     everyone@:<perms>[:inheritance flags]:<allow|deny>
436     user:<username>[:inheritance flags]:<allow|deny>
437     group:<groupname>[:inheritance flags]:<allow|deny>
438 .fi
439 .in -2

441 .sp
442 .ne 2
443 .na
444 \fBowner@\fR
445 .ad
446 .RS 10n
447 File owner
448 .RE

450 .sp
451 .ne 2
452 .na
453 \fBgroup@\fR
454 .ad
455 .RS 10n
456 Group owner
```



```
457 .RE

459 .sp
460 .ne 2
461 .na
462 \fBuser\fR
463 .ad
464 .RS 10n
465 Permissions for a specific user
466 .RE

468 .sp
469 .ne 2
470 .na
471 \fBgroup\fR
472 .ad
473 .RS 10n
474 Permissions for a specific group
475 .RE

477 .sp
478 .LP
479 Permission and inheritance flags are separated by a \fB/\fR character.
480 .sp
481 .LP
482 ACL specification examples:
483 .sp
484 .in +2
485 .nf
486 user:fred:read_data/write_data/read_attributes:file_inherit:allow
487 owner@:read_data:allow,group@:read_data:allow,user:tom:read_data:deny
488 .fi
489 .in -2
490 .sp

492 .sp
493 .LP
494 Using the compact ACL format, permissions are specified by using 14 unique
495 letters to indicate permissions.
496 .sp
497 .LP
498 Using the positional ACL format, permissions are specified as positional
499 arguments similar to the \fBls -V\fR format. The hyphen (\fB-\fR), which
500 indicates that no permission is granted at that position, can be omitted and
501 only the required letters have to be specified.
502 .sp
503 .LP
504 The letters above are listed in the order they would be specified in positional
505 notation.
506 .sp
507 .LP
508 With these letters you can specify permissions in the following equivalent
509 ways.
510 .sp
511 .in +2
512 .nf
513 user:fred:rw------R------:file_inherit:allow
514 .fi
515 .in -2
516 .sp

518 .sp
519 .LP
520 Or you can remove the \fB-\fR and scrunch it together.
521 .sp
522 .in +2
```

```
.nf
user:fred:rwR:file_inherit:allow
.fi
.in -2
.sp

.sp
.LP
The inheritance flags can also be specified in a more compact manner, as
follows:
.sp
.in +2
.nf
user:fred:rwR:f:allow
user:fred:rwR:f------:allow
.fi
.in -2
.sp

.SS "Shell-level Solaris \fBAPI\fR"
.sp
.LP
The Solaris command interface supports the manipulation of ACLs. The following
Solaris utilities accommodate both ACL models:
.sp
.ne 2
.na
\fB\fBchmod\fR\fR
.ad
.RS 12n
The \fBchmod\fR utility has been enhanced to allow for the setting and deleting
of ACLs. This is achieved by extending the symbolic-mode argument to support
ACL manipulation. See \fBchmod\fR(1) for details.
.RE

.sp
.ne 2
.na
\fB\fBcompress\fR\fR
.ad
.RS 12n
When a file is compressed any ACL associated with the original file is
preserved with the compressed file.
.RE

.sp
.ne 2
.na
\fB\fBcp\fR\fR
.ad
.RS 12n
By default, \fBcp\fR ignores ACLs, unless the \fB-p\fR option is specified.
When \fB-p\fR is specified the owner and group id, permission modes,
modification and access times, ACLs, and extended attributes if applicable are
preserved.
.RE

.sp
.ne 2
.na
\fB\fBcpio\fR\fR
.ad
.RS 12n
ACLs are preserved when the \fB-P\fR option is specified.
.RE
```



```
.sp
.ne 2
.na
\fB\fBfind\fR\fR
.ad
.RS 12n
Find locates files with ACLs when the \fB-acl\fR flag is specified.
.RE

.sp
.ne 2
.na
\fB\fBls\fR\fR
.ad
.RS 12n
By default \fBls\fR does not display ACL information. When the \fB-v\fR option
is specified, a file's ACL is displayed.
.RE

.sp
.ne 2
.na
\fB\fBmv\fR\fR
.ad
.RS 12n
When a file is moved, all attributes are carried along with the renamed file.
When a file is moved across a file system boundary, the ACLs are replicated. If
the ACL information cannot be replicated, the move fails and the source file is
not removed.
.RE

.sp
.ne 2
.na
\fB\fBpack\fR\fR
.ad
.RS 12n
When a file is packed, any ACL associated with the original file is preserved
with the packed file.
.RE

.sp
.ne 2
.na
\fB\fBrcp\fR\fR
.ad
.RS 12n
\fBrcp\fR has been enhanced to support copying. A file's ACL is only preserved
when the remote host supports ACLs.
.RE

.sp
.ne 2
.na
\fB\fBtar\fR\fR
.ad
.RS 12n
ACLs are preserved when the \fB-p\fR option is specified.
.RE

.sp
.ne 2
.na
\fB\fBunpack\fR\fR
.ad
.RS 12n
```

```
655 When a file with an ACL is unpacked, the unpacked file retains the ACL
656 information.
657 .RE

659 .SS "Application-level \fBAPI\fR"
660 .sp
661 .LP
662 The primary interfaces required to access file system ACLs at the programmatic
663 level are the \fBacl_get()\fR and \fBacl_set()\fR functions. These functions
664 support both POSIX draft ACLs and NFSv4 ACLs.
665 .SS "Retrieving a file's \fBACL\fR"
666 .sp
667 .in +2
668 .nf
669 int acl_get(const char *path, int flag, acl_t **aclp);
670 int facl_get(int fd, int flag, acl_t **aclp);
671 .fi
672 .in -2

674 .sp
675 .LP
676 The \fBacl_get\fR(3SEC) and \fBfacl_get\fR(3SEC) functions retrieves an ACL on
677 a file whose name is given by path or referenced by the open file descriptor
678 fd. The flag argument specifies whether a trivial ACL should be retrieved. When
679 the flag argument equals \fBACL_NO_TRIVIAL\fR then only ACLs that are not
680 trivial are retrieved. The ACL is returned in the \fBaclp\fR argument.
681 .SS "Freeing \fBACL\fR structure"
682 .sp
683 .in +2
684 .nf
685 void acl_free(acl_t *aclp)s;
686 .fi
687 .in -2

689 .sp
690 .LP
691 The \fBacl_free()\fR function frees up memory allocated for the argument
692 \fBaclp;\fR.
693 .SS "Setting an \fBACL\fR on a file"
694 .sp
695 .in +2
696 .nf
697 int acl_set(const char *path, acl_t *aclp);
698 int facl_set(int fd, acl_t *aclp);
699 .fi
700 .in -2

702 .sp
703 .LP
704 The \fBacl_set\fR(3SEC) and \fBfacl_get\fR(3SEC) functions are used for setting
705 an ACL on a file whose name is given by path or referenced by the open file
706 descriptor \fBfd\fR. The \fBaclp\fR argument specifies the ACL to set. The
707 \fBacl_set\fR(3SEC) translates an POSIX-draft ACL into a NFSv4 ACL when the
708 target file systems supports NFSv4 ACLs. No translation is performed when
709 trying to set an NFSv4 ACL on a POSIX-draft ACL supported file system.
710 .SS "Determining an \fBACL\fR's trivialness"
711 .sp
712 .in +2
713 .nf
714 int acl_trivial(const char *path);
715 .fi
716 .in -2

718 .sp
719 .LP
720 The \fBacl_trivial()\fR function is used to determine whether a file has a
```



```
721 trivial ACL. The trivialness of a file's ACL depends on the type of ACL it is.
722 For POSIX-draft ACLs, it implies the ACL has greater than
723 \fBMIN_ACL_ENTRIES\fR. For NFSv4/ZFS style ACLs, it implies that the ACL has
724 entries other than \fBowner@\fR, \fBgroup@\fR and \fBeveryone@\fR, inheritance
725 flags are set, or the ACL is not ordered in a manner that meets POSIX access
726 control requirements.
727 .SS "Removing all \fBACL\fRs from a file"
728 .sp
729 .in +2
730 .nf
731 int acl_strip(const char *path, uid_t uid, gid_t gid, mode_t mode);
732 .fi
733 .in -2

735 .sp
736 .LP
737 The \fBacl_strip()\fR function removes all ACLs from a file and replaces them
738 with a trivial ACL based off of the passed in argument mode. After replacing
739 the ACL the owner and group of the file are set to the values specified in the
740 uid and gid parameters.
741 .SS "Converting \fBACL\fRs to/from external representation"
742 .sp
743 .in +2
744 .nf
745 int acl_fromtext(const char *path, acl_t **aclp);
746 char *acl_totext(acl_t *aclp, int flags);
747 .fi
748 .in -2

750 .sp
751 .LP
752 The \fBacl_text()\fR function converts an internal ACL representation pointed
753 to by aclp into an external representation. See \fBDESCRIPTION\fR for details
754 about external representation.
755 .sp
756 .LP
757 The \fBacl_fromtext()\fR functions converts and external representation into an
758 internal representation. See \fBDESCRIPTION\fR for details about external
759 representation.
760 .SH EXAMPLES
761 .sp
762 .LP
763 The following examples demonstrate how the API can be used to perform basic
764 operations on ACLs.
765 .LP
766 \fBExample 1 \fRRetrieving and Setting an ACL
767 .sp
768 .LP
769 Use the following to retrieve an ACL and set it on another file:

771 .sp
772 .in +2
773 .nf
774 error = acl_get("file", ACL_NO_TRIVIAL, &aclp);

776 if (error == 0 && aclp != NULL) {
777 error = acl_set("file2", aclp)
778 acl_free(aclp);
779 }
_____unchanged_portion_omitted_
799 \&...
800 .fi
801 .in -2

803 .LP
804 \fBExample 3 \fRDetermining if a File has a Trivial ACL
```

```
805 .sp
806 .LP
807 Use the following to determine if a file has a trivial ACL:

809 .sp
810 .in +2
811 .nf
812 istrivial = acl_trivial("file")

814 if (istrivial == 0)
815 printf("file %s has a trivial ACL\en", file);
815 printf("file %s has a trivial ACL\n", file);
816 else
817 printf("file %s has a NON-trivial ACL\en", file);
817 printf("file %s has a NON-trivial ACL\n", file);
818 \&...
819 .fi
820 .in -2

822 .LP
823 \fBExample 4 \fRRemoving all ACLs from a File
824 .sp
825 .LP
826 Use the following to remove all ACLs from a file, and set a new mode, owner,
827 and group:

829 .sp
830 .in +2
831 .nf
832 error = acl_strip("file", 10, 100, 0644);
833 \&...
834 .fi
835 .in -2

837 .SH SEE ALSO
838 .sp
839 .LP
840 \fBchgrp\fR(1), \fBchmod\fR(1), \fBchown\fR(1), \fBcp\fR(1), \fBcpio\fR(1),
841 \fBfind\fR(1), \fBls\fR(1), \fBmv\fR(1), \fBtar\fR(1), \fBsetfacl\fR(1),
842 \fBchmod\fR(2), \fBacl\fR(2),\fBstat\fR(2),\fBacl_get\fR(3SEC),
843 \fBaclsort\fR(3SEC), \fBacl_fromtext\fR(3SEC), \fBacl_free\fR(3SEC),
844 \fBacl_strip\fR(3SEC), \fBacl_trivial\fR(3SEC)
```

```
**********************************************************
   53557 Tue Sep 10 18:35:21 2013
new/usr/src/man/man5/tecla.5
4023 - Typo in file(1) manpage and various others
**********************************************************
```

```
   1 '\" te
   2 .\" Copyright (c) 2000, 2001, 2002, 2003, 2004 by Martin C. Shepherd. All Rights
   3 .\" Permission is hereby granted, free of charge, to any person obtaining a copy
   4 .\" "Software"), to deal in the Software without restriction, including
   5 .\" without limitation the rights to use, copy, modify, merge, publish,
   6 .\" distribute, and/or sell copies of the Software, and to permit persons
   7 .\" to whom the Software is furnished to do so, provided that the above
   8 .\" copyright notice(s) and this permission notice appear in all copies of
   9 .\" the Software and that both the above copyright notice(s) and this
  10 .\" permission notice appear in supporting documentation.
  11 .\"
  12 .\" THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
  13 .\" OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
  14 .\" MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT
  15 .\" OF THIRD PARTY RIGHTS. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR
  16 .\" HOLDERS INCLUDED IN THIS NOTICE BE LIABLE FOR ANY CLAIM, OR ANY SPECIAL
  17 .\" INDIRECT OR CONSEQUENTIAL DAMAGES, OR ANY DAMAGES WHATSOEVER RESULTING
  18 .\" FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT,
  19 .\" NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION
  20 .\" WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.
  21 .\"
  22 .\" Except as contained in this notice, the name of a copyright holder
  23 .\" shall not be used in advertising or otherwise to promote the sale, use
  24 .\" or other dealings in this Software without prior written authorization
  25 .\" of the copyright holder.
  26 .\" Portions Copyright (c) 2005, Sun Microsystems, Inc. All Rights Reserved.
  27 .TH TECLA 5 "Sep 10, 2013"
  27 .TH TECLA 5 "May 20, 2004"
  28 .SH NAME
  29 tecla, teclarc \- User interface provided by the tecla library.
  30 .SH DESCRIPTION
  31 .sp
  32 .LP
  33 This man page describes the command-line editing features that are available to
  34 users of programs that read keyboard input via the tecla library. Users of the
  35 \fBtcsh shell\fR will find the default key bindings very familiar. Users of the
  36 \fBbash\fR shell will also find it quite familiar, but with a few minor
  37 differences, most notably in how forward and backward searches through the list
  38 of historical commands are performed. There are two major editing modes, one
  39 with \fBemacs\fR-like key bindings and another with \fBvi\fR-like key bindings.
  40 By default \fBemacs\fR mode is enabled, but \fBvi\fR(1) mode can alternatively
  41 be selected via the user's configuration file. This file can also be used to
  42 change the bindings of individual keys to suit the user's preferences. By
  43 default, tab completion is provided. If the application hasn't reconfigured
  44 this to complete other types of symbols, then tab completion completes file
  45 names.
  46 .SS "Key Sequence Notation"
  47 .sp
  48 .LP
  49 In the rest of this man page, and also in all tecla configuration files, key
  50 sequences are expressed as follows.
  51 .sp
  52 .ne 2
  53 .na
  54 \fB\fB^A\fR or \fBC-a\fR\fR
  55 .ad
  56 .RS 13n
  57 This is a 'CONTROL-A', entered by pressing the CONTROL key at the same time as
  58 the 'A' key.
  59 .RE
```



```
  61 .sp
  62 .ne 2
  63 .na
  64 \fB\fB\E\fR or \fBM-\fR\fR
  65 .ad
  66 .RS 13n
  67 In key sequences, both of these notations can be entered either by pressing the
  68 ESCAPE key, then the following key, or by pressing the META key at the same
  69 time as the following key. Thus the key sequence \fBM-p\fR can be typed in two
  70 ways, by pressing the ESCAPE key, followed by pressing 'P', or by pressing the
  71 META key at the same time as 'P'.
  72 .RE

  74 .sp
  75 .ne 2
  76 .na
  77 \fBup\fR
  78 .ad
  79 .RS 13n
  80 This refers to the up-arrow key.
  81 .RE

  83 .sp
  84 .ne 2
  85 .na
  86 \fBdown\fR
  87 .ad
  88 .RS 13n
  89 This refers to the down-arrow key.
  90 .RE

  92 .sp
  93 .ne 2
  94 .na
  95 \fBleft\fR
  96 .ad
  97 .RS 13n
  98 This refers to the left-arrow key.
  99 .RE

 101 .sp
 102 .ne 2
 103 .na
 104 \fBright\fR
 105 .ad
 106 .RS 13n
 107 This refers to the right-arrow key.
 108 .RE

 110 .sp
 111 .ne 2
 112 .na
 113 \fBa\fR
 114 .ad
 115 .RS 13n
 116 This is just a normal 'A' key.
 117 .RE

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

```
 127 important item to go in this file is the following line:
 128 .sp
 129 .in +2
 130 .nf
 131 edit-mode vi
 132 .fi
 133 .in -2

 135 .sp
 136 .LP
 137 This will re-configure the default bindings for \fBvi\fR-mode. The complete set
 138 of arguments that this command accepts are:
 139 .sp
 140 .ne 2
 141 .na
 142 \fBvi\fR
 143 .ad
 144 .RS 9n
 145 Install key bindings like those of the \fBvi\fR editor.
 146 .RE

 148 .sp
 149 .ne 2
 150 .na
 151 \fBemacs\fR
 152 .ad
 153 .RS 9n
 154 Install key bindings like those of the \fBemacs\fR editor. This is the default.
 155 .RE

 157 .sp
 158 .ne 2
 159 .na
 160 \fBnone\fR
 161 .ad
 162 .RS 9n
 163 Use just the native line editing facilities provided by the terminal driver.
 164 .RE

 166 .sp
 167 .LP
 168 To prevent the terminal bell from being rung, such as when an unrecognized
 169 control-sequence is typed, place the following line in the configuration file:
 170 .sp
 171 .in +2
 172 .nf
 173 nobeep
 174 .fi
 175 .in -2

 177 .sp
 178 .LP
 179 An example of a key binding line in the configuration file is the following.
 180 .sp
 181 .in +2
 182 .nf
 183 bind M-[2~ insert-mode
 184 .fi
 185 .in -2

 187 .sp
 188 .LP
 189 On many keyboards, the above key sequence is generated when one presses the
 190 insert key, so with this key binding, one can toggle between the
 191 \fBemacs\fR-mode insert and overwrite modes by hitting one key. One could also
 192 do it by typing out the above sequence of characters one by one. As explained
```



```
 193 above, the \fBM-\fR part of this sequence can be typed either by pressing the
 194 ESCAPE key before the following key, or by pressing the META key at the same
 195 time as the following key. Thus if you had set the above key binding, and the
 196 insert key on your keyboard didn't generate the above key sequence, you could
 197 still type it in either of the following 2 ways.
 198 .RS +4
 199 .TP
 200 1.
 201 Hit the ESCAPE key momentarily, then press '[', then '2', then finally '~'.
 202 .RE
 203 .RS +4
 204 .TP
 205 2.
 206 Press the META key at the same time as pressing the '[' key, then press '2',
 207 then '~'.
 208 .RE
 209 .sp
 210 .LP
 211 If you set a key binding for a key sequence that is already bound to a
 212 function, the new binding overrides the old one. If in the new binding you omit
 213 the name of the new function to bind to the key sequence, the original binding
 214 becomes undefined.
 215 .sp
 216 .LP
 217 Starting with versions of \fBlibtecla\fR later than 1.3.3 it is now possible to
 218 bind key sequences that begin with a printable character. Previously key
 219 sequences were required to start with a CONTROL or META character.
 220 .sp
 221 .LP
 222 Note that the special keywords "up", "down", "left", and "right" refer to the
 223 arrow keys, and are thus not treated as key sequences. So, for example, to
 224 rebind the up and down arrow keys to use the history search mechanism instead
 225 of the simple history recall method, you could place the following in your
 226 configuration file:
 227 .sp
 228 .in +2
 229 .nf
 230 bind up history-search-backwards
 231 bind down history-search-backwards
 232 .fi
 233 .in -2

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

 248 .sp
 249 .LP
 250 If you create a \fB~/.teclarc\fR configuration file, but it appears to have no
 251 effect on the program, check the documentation of the program to see if the
 252 author chose a different name for this file.
 253 .SS "Filename and Tilde Completion"
 254 .sp
 255 .LP
 256 With the default key bindings, pressing the TAB key (aka. \fB^I\fR) results in
 257 tecla attempting to complete the incomplete file name that precedes the cursor.
 258 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"
.sp
.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"
.sp
.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 interpretted as a pattern to be matched. Thus, cotinuing 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"
.sp
.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"
.sp
.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

```
391 locale name. If none of these variables are defined, or the program neglects to
392 call \fBsetlocale\fR, then the default C locale is used, which is US 7-bit
393 ASCII. On most unix-like platforms, you can get a list of valid locales by
394 typing the command:
395 .sp
396 .in +2
397 .nf
398 locale -a
399 .fi
400 .in -2

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

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



```
457 xmodmap -e 'keycode 115 = Multi_key'
458 .fi
459 .in -2

461 .sp
462 .LP
463 Type this key followed by a " character to enter an 'I' with a umlaut over it.
464 .SS "The Available Key Binding Functions"
465 .sp
466 .LP
467 The following is a list of the editing functions provided by the tecla library.
468 The names in the leftmost column of the list can be used in configuration files
469 to specify which function a given key or combination of keys should invoke.
470 They are also used in the next two sections to list the default key bindings in
471 \fBemacs\fR and \fBvi\fR modes.
472 .sp
473 .ne 2
474 .na
475 \fBuser-interrupt\fR
476 .ad
477 .RS 30n
478 Send a SIGINT signal to the parent process.
479 .RE

481 .sp
482 .ne 2
483 .na
484 \fBsuspend\fR
485 .ad
486 .RS 30n
487 Suspend the parent process.
488 .RE

490 .sp
491 .ne 2
492 .na
493 \fBstop-output\fR
494 .ad
495 .RS 30n
496 Pause terminal output.
497 .RE

499 .sp
500 .ne 2
501 .na
502 \fBstart-output\fR
503 .ad
504 .RS 30n
505 Resume paused terminal output.
506 .RE

508 .sp
509 .ne 2
510 .na
511 \fBliteral-next\fR
512 .ad
513 .RS 30n
514 Arrange for the next character to be treated as a normal character. This allows
515 control characters to be entered.
516 .RE

518 .sp
519 .ne 2
520 .na
521 \fBcursor-right\fR
522 .ad
```

```
523 .RS 30n
524 Move the cursor one character right.
525 .RE

527 .sp
528 .ne 2
529 .na
530 \fBcursor-left\fR
531 .ad
532 .RS 30n
533 Move the cursor one character left.
534 .RE

536 .sp
537 .ne 2
538 .na
539 \fBinsert-mode\fR
540 .ad
541 .RS 30n
542 Toggle between insert mode and overwrite mode.
543 .RE

545 .sp
546 .ne 2
547 .na
548 \fBbeginning-of-line\fR
549 .ad
550 .RS 30n
551 Move the cursor to the beginning of the line.
552 .RE

554 .sp
555 .ne 2
556 .na
557 \fBend-of-line\fR
558 .ad
559 .RS 30n
560 Move the cursor to the end of the line.
561 .RE

563 .sp
564 .ne 2
565 .na
566 \fBdelete-line\fR
567 .ad
568 .RS 30n
569 Delete the contents of the current line.
570 .RE

572 .sp
573 .ne 2
574 .na
575 \fBkill-line\fR
576 .ad
577 .RS 30n
578 Delete everything that follows the cursor.
579 .RE

581 .sp
582 .ne 2
583 .na
584 \fBbackward-kill-line\fR
585 .ad
586 .RS 30n
587 Delete all characters between the cursor and the start of the line.
588 .RE
```



```
590 .sp
591 .ne 2
592 .na
593 \fBforward-word\fR
594 .ad
595 .RS 30n
596 Move to the end of the word which follows the cursor.
597 .RE

599 .sp
600 .ne 2
601 .na
602 \fBforward-to-word\fR
603 .ad
604 .RS 30n
605 Move the cursor to the start of the word that follows the cursor.
606 .RE

608 .sp
609 .ne 2
610 .na
611 \fBbackward-word\fR
612 .ad
613 .RS 30n
614 Move to the start of the word which precedes the cursor.
615 .RE

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

627 .sp
628 .ne 2
629 .na
630 \fBfind-parenthesis\fR
631 .ad
632 .RS 30n
633 If the cursor is currently over a parenthesis character, move it to the
634 matching parenthesis character. If not over a parenthesis character move right
635 to the next close parenthesis.
636 .RE

638 .sp
639 .ne 2
640 .na
641 \fBforward-delete-char\fR
642 .ad
643 .RS 30n
644 Delete the character under the cursor.
645 .RE

647 .sp
648 .ne 2
649 .na
650 \fBbackward-delete-char\fR
651 .ad
652 .RS 30n
653 Delete the character which precedes the cursor.
654 .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 commmand 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
commmand 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 '\n', 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 '\n'
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
occurence 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
occurence 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
occurence 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
occurence 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 occurence
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 occurence
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
occurence 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
occurence 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
```

```
1313 to insert mode.
1314 .RE

1316 .sp
1317 .ne 2
1318 .na
1319 \fBvi-append-at-eol\fR
1320 .ad
1321 .RS 30n
1322 From \fBvi\fR command mode, move the cursor to the end of the line and switch
1323 to append mode.
1324 .RE

1326 .sp
1327 .ne 2
1328 .na
1329 \fBvi-append\fR
1330 .ad
1331 .RS 30n
1332 From \fBvi\fR command mode, move the cursor one position right, and switch to
1333 insert mode.
1334 .RE

1336 .sp
1337 .ne 2
1338 .na
1339 \fBvi-replace-char\fR
1340 .ad
1341 .RS 30n
1342 From \fBvi\fR command mode, replace the character under the cursor with the
1343 next character entered.
1344 .RE

1346 .sp
1347 .ne 2
1348 .na
1349 \fBvi-forward-change-char\fR
1350 .ad
1351 .RS 30n
1352 From \fBvi\fR command mode, delete the next character then enter insert mode.
1353 .RE

1355 .sp
1356 .ne 2
1357 .na
1358 \fBvi-backward-change-char\fR
1359 .ad
1360 .RS 30n
1361 From vi command mode, delete the preceding character then enter insert mode.
1362 .RE

1364 .sp
1365 .ne 2
1366 .na
1367 \fBvi-forward-change-word\fR
1368 .ad
1369 .RS 30n
1370 From \fBvi\fR command mode, delete the next word then enter insert mode.
1371 .RE

1373 .sp
1374 .ne 2
1375 .na
1376 \fBvi-backward-change-word\fR
1377 .ad
1378 .RS 30n
```



```
1379 From vi command mode, delete the preceding word then enter insert mode.
1380 .RE

1382 .sp
1383 .ne 2
1384 .na
1385 \fBvi-change-rest-of-line\fR
1386 .ad
1387 .RS 30n
1388 From \fBvi\fR command mode, delete from the cursor to the end of the line, then
1389 enter insert mode.
1390 .RE

1392 .sp
1393 .ne 2
1394 .na
1395 \fBvi-change-line\fR
1396 .ad
1397 .RS 30n
1398 From \fBvi\fR command mode, delete the current line, then enter insert mode.
1399 .RE

1401 .sp
1402 .ne 2
1403 .na
1404 \fBvi-change-to-bol\fR
1405 .ad
1406 .RS 30n
1407 From \fBvi\fR command mode, delete all characters between the cursor and the
1408 beginning of the line, then enter insert mode.
1409 .RE

1411 .sp
1412 .ne 2
1413 .na
1414 \fBvi-change-to-column\fR
1415 .ad
1416 .RS 30n
1417 From \fBvi\fR command mode, delete the characters from the cursor up to the
1418 column that is specified by the repeat count, then enter insert mode.
1419 .RE

1421 .sp
1422 .ne 2
1423 .na
1424 \fBvi-change-to-parenthesis\fR
1425 .ad
1426 .RS 30n
1427 Delete the characters from the cursor up to and including the matching
1428 parenthesis, or next close parenthesis, then enter \fBvi\fR insert mode.
1429 .RE

1431 .sp
1432 .ne 2
1433 .na
1434 \fBvi-forward-change-find\fR
1435 .ad
1436 .RS 30n
1437 From \fBvi\fR command mode, delete the characters from the cursor up to and
1438 including the following occurence of the next character typed, then enter
1439 insert mode.
1440 .RE

1442 .sp
1443 .ne 2
1444 .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 occurence 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 occurence 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 occurence 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"
.sp
.LP
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 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\fB^\\fR\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 refered 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 setttings.
.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
```

```
1709 \fB\fB^D\fR\fR
1710 .ad
1711 .RS 21n
1712 del-char-or-list-or-eof
1713 .RE

1715 .sp
1716 .ne 2
1717 .na
1718 \fB\fB^H\fR\fR
1719 .ad
1720 .RS 21n
1721 backward-delete-char
1722 .RE

1724 .sp
1725 .ne 2
1726 .na
1727 \fB\fB^?\fR\fR
1728 .ad
1729 .RS 21n
1730 backward-delete-char
1731 .RE

1733 .sp
1734 .ne 2
1735 .na
1736 \fB\fBM-d\fR\fR
1737 .ad
1738 .RS 21n
1739 forward-delete-word
1740 .RE

1742 .sp
1743 .ne 2
1744 .na
1745 \fB\fBM-^H\fR\fR
1746 .ad
1747 .RS 21n
1748 backward-delete-word
1749 .RE

1751 .sp
1752 .ne 2
1753 .na
1754 \fB\fBM-^?\fR\fR
1755 .ad
1756 .RS 21n
1757 backward-delete-word
1758 .RE

1760 .sp
1761 .ne 2
1762 .na
1763 \fB\fBM-u\fR\fR
1764 .ad
1765 .RS 21n
1766 upcase-word
1767 .RE

1769 .sp
1770 .ne 2
1771 .na
1772 \fB\fBM-l\fR\fR
1773 .ad
1774 .RS 21n
```



```
1775 downcase-word
1776 .RE

1778 .sp
1779 .ne 2
1780 .na
1781 \fB\fBM-c\fR\fR
1782 .ad
1783 .RS 21n
1784 capitalize-word
1785 .RE

1787 .sp
1788 .ne 2
1789 .na
1790 \fB\fB^R\fR\fR
1791 .ad
1792 .RS 21n
1793 redisplay
1794 .RE

1796 .sp
1797 .ne 2
1798 .na
1799 \fB\fB^L\fR\fR
1800 .ad
1801 .RS 21n
1802 clear-screen
1803 .RE

1805 .sp
1806 .ne 2
1807 .na
1808 \fB\fB^T\fR\fR
1809 .ad
1810 .RS 21n
1811 transpose-chars
1812 .RE

1814 .sp
1815 .ne 2
1816 .na
1817 \fB\fB^@\fR\fR
1818 .ad
1819 .RS 21n
1820 set-mark
1821 .RE

1823 .sp
1824 .ne 2
1825 .na
1826 \fB\fB^X^X\fR\fR
1827 .ad
1828 .RS 21n
1829 exchange-point-and-mark
1830 .RE

1832 .sp
1833 .ne 2
1834 .na
1835 \fB\fB^W\fR\fR
1836 .ad
1837 .RS 21n
1838 kill-region
1839 .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
\fB\fB\n\fR\fR
.ad
```

```
1972 .RS 21n
1973 newline
1974 .RE

1976 .sp
1977 .ne 2
1978 .na
1979 \fB\fB\er\fR\fR
1979 \fB\fB\r\fR\fR
1980 .ad
1981 .RS 21n
1982 newline
1983 .RE

1985 .sp
1986 .ne 2
1987 .na
1988 \fB\fBM-o\fR\fR
1989 .ad
1990 .RS 21n
1991 repeat-history
1992 .RE

1994 .sp
1995 .ne 2
1996 .na
1997 \fB\fBM-^V\fR\fR
1998 .ad
1999 .RS 21n
2000 \fBvi\fR-mode
2001 .RE

2003 .sp
2004 .ne 2
2005 .na
2006 \fB\fBM-0, M-1, ... M-9\fR\fR
2007 .ad
2008 .RS 21n
2009 digit-argument (see below)
2010 .RE

2012 .sp
2013 .LP
2014 Note that \fB^I\fR is what the TAB key generates, and that \fB^@\fR can be
2015 generated not only by pressing the CONTROL key and the @ key simultaneously,
2016 but also by pressing the CONTROL key and the space bar at the same time.
2017 .SS "Default Key Bindings in \fBvi\fR Mode"
2018 .sp
2019 .LP
2020 The following default key bindings are designed to mimic the \fBvi\fR style of
2021 editing as closely as possible. This means that very few editing functions are
2022 provided in the initial character input mode, editing functions instead being
2023 provided by the \fBvi\fR command mode. The \fBvi\fR command mode is entered
2024 whenever the ESCAPE character is pressed, or whenever a key sequence that
2025 starts with a meta character is entered. In addition to mimicing \fBvi\fR,
2026 \fBlibtecla\fR provides bindings for tab completion, wild-card expansion of
2027 file names, and historical line recall.
2028 .sp
2029 .LP
2030 To learn how to tell the tecla library to use \fBvi\fR mode instead of the
2031 default \fBemacs\fR editing mode, see the earlier section entitled The Tecla
2032 Configuration File.
2033 .sp
2034 .LP
2035 Under UNIX the terminal driver sets a number of special keys for certain
2036 functions. The tecla library attempts to use the same key bindings to maintain
```



```
2037 consistency, binding them both in input mode and in command mode. The key
2038 sequences shown for the following 6 bindings are thus just examples of what
2039 they will probably be set to. If you have used the \fBstty\fR command to change
2040 these keys, then the default bindings should match.
2041 .sp
2042 .ne 2
2043 .na
2044 \fB\fB^C\fR\fR
2045 .ad
2046 .RS 8n
2047 user-interrupt
2048 .RE

2050 .sp
2051 .ne 2
2052 .na
2053 \fB\fB^\\fR\fR
2054 .ad
2055 .RS 8n
2056 abort
2057 .RE

2059 .sp
2060 .ne 2
2061 .na
2062 \fB\fB^Z\fR\fR
2063 .ad
2064 .RS 8n
2065 suspend
2066 .RE

2068 .sp
2069 .ne 2
2070 .na
2071 \fB\fB^Q\fR\fR
2072 .ad
2073 .RS 8n
2074 start-output
2075 .RE

2077 .sp
2078 .ne 2
2079 .na
2080 \fB\fB^S\fR\fR
2081 .ad
2082 .RS 8n
2083 stop-output
2084 .RE

2086 .sp
2087 .ne 2
2088 .na
2089 \fB\fB^V\fR\fR
2090 .ad
2091 .RS 8n
2092 literal-next
2093 .RE

2095 .sp
2096 .ne 2
2097 .na
2098 \fB\fBM-^C\fR\fR
2099 .ad
2100 .RS 8n
2101 user-interrupt
2102 .RE
```

```
.sp
.ne 2
.na
\fB\fBM-^\\fR\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 refered 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
\fB\fB\r\fR\fR
.ad
.RS 8n
newline
.RE

.sp
.ne 2
.na
\fB\fB\en\fR\fR
\fB\fB\n\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
\fB\fBM-\\fR\fR
.ad
.RS 21n
cursor-right (META-space)
.RE
```

```
2365 .sp
2366 .ne 2
2367 .na
2368 \fB\fBM-$\fR\fR
2369 .ad
2370 .RS 21n
2371 end-of-line
2372 .RE

2374 .sp
2375 .ne 2
2376 .na
2377 \fB\fBM-*\fR\fR
2378 .ad
2379 .RS 21n
2380 expand-filename
2381 .RE

2383 .sp
2384 .ne 2
2385 .na
2386 \fB\fBM-+\fR\fR
2387 .ad
2388 .RS 21n
2389 down-history
2390 .RE

2392 .sp
2393 .ne 2
2394 .na
2395 \fB\fBM--\fR\fR
2396 .ad
2397 .RS 21n
2398 up-history
2399 .RE

2401 .sp
2402 .ne 2
2403 .na
2404 \fB\fBM-<\fR\fR
2405 .ad
2406 .RS 21n
2407 beginning-of-history
2408 .RE

2410 .sp
2411 .ne 2
2412 .na
2413 \fB\fBM->\fR\fR
2414 .ad
2415 .RS 21n
2416 end-of-history
2417 .RE

2419 .sp
2420 .ne 2
2421 .na
2422 \fB\fBM-^\fR\fR
2423 .ad
2424 .RS 21n
2425 beginning-of-line
2426 .RE

2428 .sp
2429 .ne 2
2430 .na
```



```
2431 \fB\fBM-\fR\fR
2432 .ad
2433 .RS 21n
2434 repeat-find-char
2435 .RE

2437 .sp
2438 .ne 2
2439 .na
2440 \fB\fBM-,\fR\fR
2441 .ad
2442 .RS 21n
2443 invert-refind-char
2444 .RE

2446 .sp
2447 .ne 2
2448 .na
2449 \fB\fBM-|\fR\fR
2450 .ad
2451 .RS 21n
2452 goto-column
2453 .RE

2455 .sp
2456 .ne 2
2457 .na
2458 \fB\fBM-~\fR\fR
2459 .ad
2460 .RS 21n
2461 change-case
2462 .RE

2464 .sp
2465 .ne 2
2466 .na
2467 \fB\fBM-.\fR\fR
2468 .ad
2469 .RS 21n
2470 vi-repeat-change
2471 .RE

2473 .sp
2474 .ne 2
2475 .na
2476 \fB\fBM-%\fR\fR
2477 .ad
2478 .RS 21n
2479 find-parenthesis
2480 .RE

2482 .sp
2483 .ne 2
2484 .na
2485 \fB\fBM-a\fR\fR
2486 .ad
2487 .RS 21n
2488 vi-append
2489 .RE

2491 .sp
2492 .ne 2
2493 .na
2494 \fB\fBM-A\fR\fR
2495 .ad
2496 .RS 21n
```

```
2497 vi-append-at-eol
2498 .RE

2500 .sp
2501 .ne 2
2502 .na
2503 \fB\fBM-b\fR\fR
2504 .ad
2505 .RS 21n
2506 backward-word
2507 .RE

2509 .sp
2510 .ne 2
2511 .na
2512 \fB\fBM-B\fR\fR
2513 .ad
2514 .RS 21n
2515 backward-word
2516 .RE

2518 .sp
2519 .ne 2
2520 .na
2521 \fB\fBM-C\fR\fR
2522 .ad
2523 .RS 21n
2524 vi-change-rest-of-line
2525 .RE

2527 .sp
2528 .ne 2
2529 .na
2530 \fB\fBM-cb\fR\fR
2531 .ad
2532 .RS 21n
2533 vi-backward-change-word
2534 .RE

2536 .sp
2537 .ne 2
2538 .na
2539 \fB\fBM-cB\fR\fR
2540 .ad
2541 .RS 21n
2542 vi-backward-change-word
2543 .RE

2545 .sp
2546 .ne 2
2547 .na
2548 \fB\fBM-cc\fR\fR
2549 .ad
2550 .RS 21n
2551 vi-change-line
2552 .RE

2554 .sp
2555 .ne 2
2556 .na
2557 \fB\fBM-ce\fR\fR
2558 .ad
2559 .RS 21n
2560 vi-forward-change-word
2561 .RE
```



```
2563 .sp
2564 .ne 2
2565 .na
2566 \fB\fBM-cE\fR\fR
2567 .ad
2568 .RS 21n
2569 vi-forward-change-word
2570 .RE

2572 .sp
2573 .ne 2
2574 .na
2575 \fB\fBM-cw\fR\fR
2576 .ad
2577 .RS 21n
2578 vi-forward-change-word
2579 .RE

2581 .sp
2582 .ne 2
2583 .na
2584 \fB\fBM-cW\fR\fR
2585 .ad
2586 .RS 21n
2587 vi-forward-change-word
2588 .RE

2590 .sp
2591 .ne 2
2592 .na
2593 \fB\fBM-cF\fR\fR
2594 .ad
2595 .RS 21n
2596 vi-backward-change-find
2597 .RE

2599 .sp
2600 .ne 2
2601 .na
2602 \fB\fBM-cf\fR\fR
2603 .ad
2604 .RS 21n
2605 vi-forward-change-find
2606 .RE

2608 .sp
2609 .ne 2
2610 .na
2611 \fB\fBM-cT\fR\fR
2612 .ad
2613 .RS 21n
2614 vi-backward-change-to
2615 .RE

2617 .sp
2618 .ne 2
2619 .na
2620 \fB\fBM-ct\fR\fR
2621 .ad
2622 .RS 21n
2623 vi-forward-change-to
2624 .RE

2626 .sp
2627 .ne 2
2628 .na
```

```
2629 \fB\fBM-c;\fR\fR
2630 .ad
2631 .RS 21n
2632 vi-change-refind
2633 .RE

2635 .sp
2636 .ne 2
2637 .na
2638 \fB\fBM-c,\fR\fR
2639 .ad
2640 .RS 21n
2641 vi-change-invert-refind
2642 .RE

2644 .sp
2645 .ne 2
2646 .na
2647 \fB\fBM-ch\fR\fR
2648 .ad
2649 .RS 21n
2650 vi-backward-change-char
2651 .RE

2653 .sp
2654 .ne 2
2655 .na
2656 \fB\fBM-c^H\fR\fR
2657 .ad
2658 .RS 21n
2659 vi-backward-change-char
2660 .RE

2662 .sp
2663 .ne 2
2664 .na
2665 \fB\fBM-c^?\fR\fR
2666 .ad
2667 .RS 21n
2668 vi-backward-change-char
2669 .RE

2671 .sp
2672 .ne 2
2673 .na
2674 \fB\fBM-cl\fR\fR
2675 .ad
2676 .RS 21n
2677 vi-forward-change-char
2678 .RE

2680 .sp
2681 .ne 2
2682 .na
2683 \fB\fBM-c\\fR\fR
2684 .ad
2685 .RS 21n
2686 vi-forward-change-char (META-c-space)
2687 .RE

2689 .sp
2690 .ne 2
2691 .na
2692 \fB\fBM-c^\fR\fR
2693 .ad
2694 .RS 21n
```



```
2695 vi-change-to-bol
2696 .RE

2698 .sp
2699 .ne 2
2700 .na
2701 \fB\fBM-c0\fR\fR
2702 .ad
2703 .RS 21n
2704 vi-change-to-bol
2705 .RE

2707 .sp
2708 .ne 2
2709 .na
2710 \fB\fBM-c$\fR\fR
2711 .ad
2712 .RS 21n
2713 vi-change-rest-of-line
2714 .RE

2716 .sp
2717 .ne 2
2718 .na
2719 \fB\fBM-c|\fR\fR
2720 .ad
2721 .RS 21n
2722 vi-change-to-column
2723 .RE

2725 .sp
2726 .ne 2
2727 .na
2728 \fB\fBM-c%\fR\fR
2729 .ad
2730 .RS 21n
2731 vi-change-to-parenthesis
2732 .RE

2734 .sp
2735 .ne 2
2736 .na
2737 \fB\fBM-dh\fR\fR
2738 .ad
2739 .RS 21n
2740 backward-delete-char
2741 .RE

2743 .sp
2744 .ne 2
2745 .na
2746 \fB\fBM-d^H\fR\fR
2747 .ad
2748 .RS 21n
2749 backward-delete-char
2750 .RE

2752 .sp
2753 .ne 2
2754 .na
2755 \fB\fBM-d^?\fR\fR
2756 .ad
2757 .RS 21n
2758 backward-delete-char
2759 .RE
```

```
.sp
.ne 2
.na
\fB\fBM-dl\fR\fR
.ad
.RS 21n
forward-delete-char
.RE

.sp
.ne 2
.na
\fB\fBM-d\fR\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
```

```
2893 delete-invert-refind
2894 .RE

2896 .sp
2897 .ne 2
2898 .na
2899 \fB\fBM-d^\fR\fR
2900 .ad
2901 .RS 21n
2902 backward-kill-line
2903 .RE

2905 .sp
2906 .ne 2
2907 .na
2908 \fB\fBM-d0\fR\fR
2909 .ad
2910 .RS 21n
2911 backward-kill-line
2912 .RE

2914 .sp
2915 .ne 2
2916 .na
2917 \fB\fBM-d$\fR\fR
2918 .ad
2919 .RS 21n
2920 kill-line
2921 .RE

2923 .sp
2924 .ne 2
2925 .na
2926 \fB\fBM-D\fR\fR
2927 .ad
2928 .RS 21n
2929 kill-line
2930 .RE

2932 .sp
2933 .ne 2
2934 .na
2935 \fB\fBM-d|\fR\fR
2936 .ad
2937 .RS 21n
2938 delete-to-column
2939 .RE

2941 .sp
2942 .ne 2
2943 .na
2944 \fB\fBM-d%\fR\fR
2945 .ad
2946 .RS 21n
2947 delete-to-parenthesis
2948 .RE

2950 .sp
2951 .ne 2
2952 .na
2953 \fB\fBM-e\fR\fR
2954 .ad
2955 .RS 21n
2956 forward-word
2957 .RE
```



```
2959 .sp
2960 .ne 2
2961 .na
2962 \fB\fBM-E\fR\fR
2963 .ad
2964 .RS 21n
2965 forward-word
2966 .RE

2968 .sp
2969 .ne 2
2970 .na
2971 \fB\fBM-f\fR\fR
2972 .ad
2973 .RS 21n
2974 forward-find-char
2975 .RE

2977 .sp
2978 .ne 2
2979 .na
2980 \fB\fBM-F\fR\fR
2981 .ad
2982 .RS 21n
2983 backward-find-char
2984 .RE

2986 .sp
2987 .ne 2
2988 .na
2989 \fB\fBM--\fR\fR
2990 .ad
2991 .RS 21n
2992 up-history
2993 .RE

2995 .sp
2996 .ne 2
2997 .na
2998 \fB\fBM-h\fR\fR
2999 .ad
3000 .RS 21n
3001 cursor-left
3002 .RE

3004 .sp
3005 .ne 2
3006 .na
3007 \fB\fBM-H\fR\fR
3008 .ad
3009 .RS 21n
3010 beginning-of-history
3011 .RE

3013 .sp
3014 .ne 2
3015 .na
3016 \fB\fBM-i\fR\fR
3017 .ad
3018 .RS 21n
3019 vi-insert
3020 .RE

3022 .sp
3023 .ne 2
3024 .na
```

```
3025 \fB\fBM-I\fR\fR
3026 .ad
3027 .RS 21n
3028 vi-insert-at-bol
3029 .RE

3031 .sp
3032 .ne 2
3033 .na
3034 \fB\fBM-j\fR\fR
3035 .ad
3036 .RS 21n
3037 down-history
3038 .RE

3040 .sp
3041 .ne 2
3042 .na
3043 \fB\fBM-J\fR\fR
3044 .ad
3045 .RS 21n
3046 history-search-forward
3047 .RE

3049 .sp
3050 .ne 2
3051 .na
3052 \fB\fBM-k\fR\fR
3053 .ad
3054 .RS 21n
3055 up-history
3056 .RE

3058 .sp
3059 .ne 2
3060 .na
3061 \fB\fBM-K\fR\fR
3062 .ad
3063 .RS 21n
3064 history-search-backward
3065 .RE

3067 .sp
3068 .ne 2
3069 .na
3070 \fB\fBM-l\fR\fR
3071 .ad
3072 .RS 21n
3073 cursor-right
3074 .RE

3076 .sp
3077 .ne 2
3078 .na
3079 \fB\fBM-L\fR\fR
3080 .ad
3081 .RS 21n
3082 end-of-history
3083 .RE

3085 .sp
3086 .ne 2
3087 .na
3088 \fB\fBM-n\fR\fR
3089 .ad
3090 .RS 21n
```



```
3091 history-re-search-forward
3092 .RE

3094 .sp
3095 .ne 2
3096 .na
3097 \fB\fBM-N\fR\fR
3098 .ad
3099 .RS 21n
3100 history-re-search-backward
3101 .RE

3103 .sp
3104 .ne 2
3105 .na
3106 \fB\fBM-p\fR\fR
3107 .ad
3108 .RS 21n
3109 append-yank
3110 .RE

3112 .sp
3113 .ne 2
3114 .na
3115 \fB\fBM-P\fR\fR
3116 .ad
3117 .RS 21n
3118 yank
3119 .RE

3121 .sp
3122 .ne 2
3123 .na
3124 \fB\fBM-r\fR\fR
3125 .ad
3126 .RS 21n
3127 vi-replace-char
3128 .RE

3130 .sp
3131 .ne 2
3132 .na
3133 \fB\fBM-R\fR\fR
3134 .ad
3135 .RS 21n
3136 vi-overwrite
3137 .RE

3139 .sp
3140 .ne 2
3141 .na
3142 \fB\fBM-s\fR\fR
3143 .ad
3144 .RS 21n
3145 vi-forward-change-char
3146 .RE

3148 .sp
3149 .ne 2
3150 .na
3151 \fB\fBM-S\fR\fR
3152 .ad
3153 .RS 21n
3154 vi-change-line
3155 .RE
```

```
3157 .sp
3158 .ne 2
3159 .na
3160 \fB\fBM-t\fR\fR
3161 .ad
3162 .RS 21n
3163 forward-to-char
3164 .RE

3166 .sp
3167 .ne 2
3168 .na
3169 \fB\fBM-T\fR\fR
3170 .ad
3171 .RS 21n
3172 backward-to-char
3173 .RE

3175 .sp
3176 .ne 2
3177 .na
3178 \fB\fBM-u\fR\fR
3179 .ad
3180 .RS 21n
3181 vi-undo
3182 .RE

3184 .sp
3185 .ne 2
3186 .na
3187 \fB\fBM-w\fR\fR
3188 .ad
3189 .RS 21n
3190 forward-to-word
3191 .RE

3193 .sp
3194 .ne 2
3195 .na
3196 \fB\fBM-W\fR\fR
3197 .ad
3198 .RS 21n
3199 forward-to-word
3200 .RE

3202 .sp
3203 .ne 2
3204 .na
3205 \fB\fBM-x\fR\fR
3206 .ad
3207 .RS 21n
3208 forward-delete-char
3209 .RE

3211 .sp
3212 .ne 2
3213 .na
3214 \fB\fBM-X\fR\fR
3215 .ad
3216 .RS 21n
3217 backward-delete-char
3218 .RE

3220 .sp
3221 .ne 2
3222 .na
```



```
3223 \fB\fBM-yh\fR\fR
3224 .ad
3225 .RS 21n
3226 backward-copy-char
3227 .RE

3229 .sp
3230 .ne 2
3231 .na
3232 \fB\fBM-y^H\fR\fR
3233 .ad
3234 .RS 21n
3235 backward-copy-char
3236 .RE

3238 .sp
3239 .ne 2
3240 .na
3241 \fB\fBM-y^?\fR\fR
3242 .ad
3243 .RS 21n
3244 backward-copy-char
3245 .RE

3247 .sp
3248 .ne 2
3249 .na
3250 \fB\fBM-yl\fR\fR
3251 .ad
3252 .RS 21n
3253 forward-copy-char
3254 .RE

3256 .sp
3257 .ne 2
3258 .na
3259 \fB\fBM-y\\fR\fR
3260 .ad
3261 .RS 21n
3262 forward-copy-char (META-y-space)
3263 .RE

3265 .sp
3266 .ne 2
3267 .na
3268 \fB\fBM-ye\fR\fR
3269 .ad
3270 .RS 21n
3271 forward-copy-word
3272 .RE

3274 .sp
3275 .ne 2
3276 .na
3277 \fB\fBM-yE\fR\fR
3278 .ad
3279 .RS 21n
3280 forward-copy-word
3281 .RE

3283 .sp
3284 .ne 2
3285 .na
3286 \fB\fBM-yw\fR\fR
3287 .ad
3288 .RS 21n
```

```
3289 forward-copy-word
3290 .RE

3292 .sp
3293 .ne 2
3294 .na
3295 \fB\fBM-yW\fR\fR
3296 .ad
3297 .RS 21n
3298 forward-copy-word
3299 .RE

3301 .sp
3302 .ne 2
3303 .na
3304 \fB\fBM-yb\fR\fR
3305 .ad
3306 .RS 21n
3307 backward-copy-word
3308 .RE

3310 .sp
3311 .ne 2
3312 .na
3313 \fB\fBM-yB\fR\fR
3314 .ad
3315 .RS 21n
3316 backward-copy-word
3317 .RE

3319 .sp
3320 .ne 2
3321 .na
3322 \fB\fBM-yf\fR\fR
3323 .ad
3324 .RS 21n
3325 forward-copy-find
3326 .RE

3328 .sp
3329 .ne 2
3330 .na
3331 \fB\fBM-yF\fR\fR
3332 .ad
3333 .RS 21n
3334 backward-copy-find
3335 .RE

3337 .sp
3338 .ne 2
3339 .na
3340 \fB\fBM-yt\fR\fR
3341 .ad
3342 .RS 21n
3343 forward-copy-to
3344 .RE

3346 .sp
3347 .ne 2
3348 .na
3349 \fB\fBM-yT\fR\fR
3350 .ad
3351 .RS 21n
3352 backward-copy-to
3353 .RE
```



```
3355 .sp
3356 .ne 2
3357 .na
3358 \fB\fBM-y;\fR\fR
3359 .ad
3360 .RS 21n
3361 copy-refind
3362 .RE

3364 .sp
3365 .ne 2
3366 .na
3367 \fB\fBM-y,\fR\fR
3368 .ad
3369 .RS 21n
3370 copy-invert-refind
3371 .RE

3373 .sp
3374 .ne 2
3375 .na
3376 \fB\fBM-y^\fR\fR
3377 .ad
3378 .RS 21n
3379 copy-to-bol
3380 .RE

3382 .sp
3383 .ne 2
3384 .na
3385 \fB\fBM-y0\fR\fR
3386 .ad
3387 .RS 21n
3388 copy-to-bol
3389 .RE

3391 .sp
3392 .ne 2
3393 .na
3394 \fB\fBM-y$\fR\fR
3395 .ad
3396 .RS 21n
3397 copy-rest-of-line
3398 .RE

3400 .sp
3401 .ne 2
3402 .na
3403 \fB\fBM-yy\fR\fR
3404 .ad
3405 .RS 21n
3406 copy-line
3407 .RE

3409 .sp
3410 .ne 2
3411 .na
3412 \fB\fBM-Y\fR\fR
3413 .ad
3414 .RS 21n
3415 copy-line
3416 .RE

3418 .sp
3419 .ne 2
3420 .na
```

```
3421 \fB\fBM-y|\fR\fR
3422 .ad
3423 .RS 21n
3424 copy-to-column
3425 .RE

3427 .sp
3428 .ne 2
3429 .na
3430 \fB\fBM-y%\fR\fR
3431 .ad
3432 .RS 21n
3433 copy-to-parenthesis
3434 .RE

3436 .sp
3437 .ne 2
3438 .na
3439 \fB\fBM-^E\fR\fR
3440 .ad
3441 .RS 21n
3442 emacs-mode
3443 .RE

3445 .sp
3446 .ne 2
3447 .na
3448 \fB\fBM-^H\fR\fR
3449 .ad
3450 .RS 21n
3451 cursor-left
3452 .RE

3454 .sp
3455 .ne 2
3456 .na
3457 \fB\fBM-^?\fR\fR
3458 .ad
3459 .RS 21n
3460 cursor-left
3461 .RE

3463 .sp
3464 .ne 2
3465 .na
3466 \fB\fBM-^L\fR\fR
3467 .ad
3468 .RS 21n
3469 clear-screen
3470 .RE

3472 .sp
3473 .ne 2
3474 .na
3475 \fB\fBM-^N\fR\fR
3476 .ad
3477 .RS 21n
3478 down-history
3479 .RE

3481 .sp
3482 .ne 2
3483 .na
3484 \fB\fBM-^P\fR\fR
3485 .ad
3486 .RS 21n
```



```
3487 up-history
3488 .RE

3490 .sp
3491 .ne 2
3492 .na
3493 \fB\fBM-^R\fR\fR
3494 .ad
3495 .RS 21n
3496 redisplay
3497 .RE

3499 .sp
3500 .ne 2
3501 .na
3502 \fB\fBM-^D\fR\fR
3503 .ad
3504 .RS 21n
3505 list-or-eof
3506 .RE

3508 .sp
3509 .ne 2
3510 .na
3511 \fB\fBM-^I\fR\fR
3512 .ad
3513 .RS 21n
3514 complete-word
3515 .RE

3517 .sp
3518 .ne 2
3519 .na
3520 \fB\fBM-\r\fR\fR
3521 .ad
3522 .RS 21n
3523 newline
3524 .RE

3526 .sp
3527 .ne 2
3528 .na
3529 \fB\fBM-\en\fR\fR
3529 \fB\fBM-\n\fR\fR
3530 .ad
3531 .RS 21n
3532 newline
3533 .RE

3535 .sp
3536 .ne 2
3537 .na
3538 \fB\fBM-^X^R\fR\fR
3539 .ad
3540 .RS 21n
3541 read-init-files
3542 .RE

3544 .sp
3545 .ne 2
3546 .na
3547 \fB\fBM-^Xh\fR\fR
3548 .ad
3549 .RS 21n
3550 list-history
3551 .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"
.sp
.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
.sp
.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
.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
.TE

.SH SEE ALSO
.sp
.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)
```

```
**********************************************************
   12638 Tue Sep 10 18:35:21 2013
new/usr/src/man/man7d/poll.7d
4023 - Typo in file(1) manpage and various others
**********************************************************
'\" te
.\" Copyright (c) 2007 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 POLL 7D "Sep 10, 2013"
.TH POLL 7D "Mar 28, 2007"
.SH NAME
poll \- driver for fast poll on many file descriptors
.SH SYNOPSIS
.LP
.nf
\fB#include <sys/devpoll.h>
int fd = open("/dev/poll", O_RDWR);
ssize_t n = write(int fd, struct pollfd buf[], int bufsize);
int n = ioctl(int fd, DP_POLL, struct dvpoll* arg);
int n = ioctl(int fd, DP_ISPOLLED, struct pollfd* pfd);\fR
.fi

.SH PARAMETERS
.sp
.ne 2
.na
\fB\fIfd\fR \fR
.ad
.RS 12n
Open file descriptor that refers to the  \fB/dev/poll\fR driver.
.RE

.sp
.ne 2
.na
\fB\fIpath\fR \fR
.ad
.RS 12n
\fB/dev/poll\fR
.RE

.sp
.ne 2
.na
\fB\fIbuf\fR \fR
.ad
.RS 12n
Array of  \fBpollfd\fR structures.
.RE

.sp
.ne 2
.na
\fB\fIbufsize\fR \fR
.ad
.RS 12n
Size of  \fIbuf\fR in bytes.
.RE

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



```
.RS 12n
Pointer to  \fBpollcall\fR structure.
.RE

.sp
.ne 2
.na
\fB\fIpfd\fR \fR
.ad
.RS 12n
Pointer to \fBpollfd\fR structure.
.RE

.SH DESCRIPTION
.LP
Note -
.sp
.RS 2
The \fB/dev/poll\fR device, associated driver and corresponding manpages may be
removed in a future Solaris release. For similar functionality in the event
ports framework, see \fBport_create\fR(3C).
.RE
.sp
.LP
The  \fB/dev/poll\fR driver is a special driver that enables you to monitor
multiple sets  of polled file descriptors. By using the  \fB/dev/poll\fR
driver, you can efficiently poll large numbers of file descriptors. Access to
the \fB/dev/poll\fR driver is provided through \fBopen\fR(2), \fBwrite\fR(2),
and  \fBioctl(2)\fR system calls.
.sp
.LP
Writing an array of  \fBpollfd\fR struct to the  \fB/dev/poll\fR driver has the
effect of  adding these file descriptors to the monitored \fBpoll\fR file
descriptor set  represented by the \fIfd\fR. To monitor multiple file
descriptor sets, open the \fB/dev/poll\fR driver multiple times. Each \fBfd\fR
corresponds to one set. For each \fBpollfd\fR struct entry (defined in
\fBsys/poll.h\fR):
.sp
.in +2
.nf
 struct pollfd {
    int  fd;
    short events;
    short revents;
 }
.fi
.in -2

.sp
.LP
The  \fBfd\fR field specifies the file descriptor being polled. The
\fBevents\fR field indicates the interested \fBpoll\fR \fBevents\fR on the file
descriptor. If a \fBpollfd\fR array contains multiple \fBpollfd\fR entries with
the same \fBfd\fR field, the "events" field in each \fBpollfd\fR entry is
OR'ed. A special \fBPOLLREMOVE\fR event in the \fBevents\fR field of the
\fBpollfd\fR structure removes the \fBfd\fR from the monitored set. The
\fBrevents\fR field is not used. Write returns the  number of bytes written
successfully or \fB-1\fR when write fails.
.sp
.LP
The \fBDP_POLL\fR ioctl is used to retrieve returned \fBpoll\fR \fBevents\fR
occured on the  polled file descriptors in the monitored set represented by
\fIfd\fR. \fIarg\fR \fIis\fR \fIa\fR pointer to the devpoll structures which
are defined as follows:
.sp
.in +2
```

```
.nf
 struct dvpoll {
     struct pollfd* dp_fds;
     int dp_nfds;
     int dp_timeout;
 }
.fi
.in -2

.sp
.LP
The  \fBdp_fds\fR points to a buffer that holds an array of returned
\fBpollfd\fR structures. The \fBdp_nfds\fR field specifies the size of the
buffer in terms of the  number of \fBpollfd\fR entries it contains. The
\fBdp_nfds\fR field also indicates the maximum number of file descriptors from
which poll information can be obtained. If there is no interested  \fBevents\fR
on any of the polled file descriptors, the \fBDP_POLL\fR ioctl call will wait
\fBdp_timeout\fR milliseconds before returning. If \fBdp_timeout\fR is
\fB0\fR, the ioctl call returns immediately. If \fBdp_timeout\fR is \fB-1\fR,
the call blocks until an interested \fBpoll\fR \fBevents\fR is available or the
call is interrupted. Upon return, if the ioctl call has failed, \fB-1\fR is
returned. The memory content pointed by \fBdp_fds\fR is not modified. A return
value \fB0\fR means the ioctl is timed out. In this case, the memory content
pointed by  \fBdp_fds\fR is not modified. If the call is successful, it returns
the number of valid \fBpollfd\fR entries in  the array pointed by \fBdp_fds\fR;
the contents of the rest of the buffer is undefined. For each valid
\fBpollfd\fR entry, the \fBfd\fR field indicates the file desciptor on which
the polled \fBevents\fR happened. The  \fBevents\fR field is the user specified
\fBpoll\fR \fBevents\fR. The \fBrevents\fR field contains the  \fBevents\fR
occurred. \fB-1\fR is returned if the  call fails.
.sp
.LP
\fBDP_ISPOLLED\fR ioctl allows you to query if a file descriptor is already in
the  monitored set represented by  \fBfd\fR. The \fBfd\fR field of the
\fBpollfd\fR structure indicates the file descriptor of interest. The
\fBDP_ISPOLLED\fR ioctl returns  \fB1\fR if the file descriptor is in the set.
The \fBevents\fR field contains  \fB0\fR. The \fBrevents\fR field contains the
currently polled \fBevents\fR. The ioctl returns  \fB0\fR if the file
descriptor is not in the set. The  \fBpollfd\fR structure pointed by  \fIpfd\fR
is not modified. The ioctl returns a  \fB-1\fR if the call fails.
.SH EXAMPLES
.sp
.LP
The following example shows how  \fB/dev/poll\fR may be used.
.sp
.in +2
.nf
{
        ...
        /*
         * open the driver
         */
        if ((wfd = open("/dev/poll", O_RDWR)) < 0) {
                exit(-1);
        }
        pollfd = (struct pollfd* )malloc(sizeof(struct pollfd) * MAXBUF);
        if (pollfd == NULL) {
                close(wfd);
                exit(-1);
        }
        /*
         * initialize buffer
         */
        for (i = 0; i < MAXBUF; i++) {
                pollfd[i].fd = fds[i];
                pollfd[i].events = POLLIN;
```



```
                pollfd[i].revents = 0;
        }
        if (write(wfd, &pollfd[0], sizeof(struct pollfd) * MAXBUF) !=
                        sizeof(struct pollfd) * MAXBUF) {
                perror("failed to write all pollfds");
                close (wfd);
                free(pollfd);
                exit(-1);
        }
        /*
         * read from the devpoll driver
         */
        dopoll.dp_timeout = -1;
        dopoll.dp_nfds = MAXBUF;
        dopoll.dp_fds = pollfd;
        result = ioctl(wfd, DP_POLL, &dopoll);
        if (result < 0) {
                perror("/dev/poll ioctl DP_POLL failed");
                close (wfd);
                free(pollfd);
                exit(-1);
        }
        for (i = 0; i < result; i++) {
                read(dopoll.dp_fds[i].fd, rbuf, STRLEN);
        }
 ...
}
.fi
.in -2

.sp
.LP
The following example is part of a test program which shows how
\fBDP_ISPOLLED()\fR ioctl may be used.
.sp
.in +2
.nf
{
        ...

        loopcnt = 0;
        while (loopcnt < ITERATION) {
                rn = random();
                rn %= RANGE;
                if (write(fds[rn], TESTSTRING, strlen(TESTSTRING)) !=
                                strlen(TESTSTRING)) {
                        perror("write to fifo failed.");
                        close (wfd);
                        free(pollfd);
                        error = 1;
                        goto out1;
                }
                dpfd.fd = fds[rn];
                dpfd.events = 0;
                dpfd.revents = 0;
                result = ioctl(wfd, DP_ISPOLLED, &dpfd);
                if (result < 0) {
                        perror("/dev/poll ioctl DP_ISPOLLED failed");
                        printf("errno = %d\en", errno);
                        printf("errno = %d\n", errno);
                        close (wfd);
                        free(pollfd);
                        error = 1;
                        goto out1;
                }
                if (result != 1) {
```

```
                        printf("DP_ISPOLLED returned incorrect result: %d.\en",
                        printf("DP_ISPOLLED returned incorrect result: %d.\n",
                                result);
                        close (wfd);
                        free(pollfd);
                        error = 1;
                        goto out1;
                }
                if (dpfd.fd != fds[rn]) {
                        printf("DP_ISPOLLED returned wrong fd %d, expect %d\en",
                        printf("DP_ISPOLLED returned wrong fd %d, expect %d\n",
                                dpfd.fd, fds[rn]);
                        close (wfd);
                        free(pollfd);
                        error = 1;
                        goto out1;
 }
                if (dpfd.revents != POLLIN) {
                        printf("DP_ISPOLLED returned unexpected revents %d\en",
                        printf("DP_ISPOLLED returned unexpected revents %d\n",
                                dpfd.revents);
                        close (wfd);
                        free(pollfd);
                        error = 1;
                        goto out1;
                }
                if (read(dpfd.fd, rbuf, strlen(TESTSTRING)) !=
                                strlen(TESTSTRING)) {
                        perror("read from fifo failed");
                        close (wfd);
                        free(pollfd);
                        error = 1;
                        goto out1;
                }
                loopcnt++;
        }

.fi
.in -2

.SH ERRORS
.sp
.ne 2
.na
\fB\fBEACCES\fR \fR
.ad
.RS 11n
A process does not have permission to access the content cached in
\fB/dev/poll\fR.
.RE

.sp
.ne 2
.na
\fB\fBEINTR\fR \fR
.ad
.RS 11n
A signal was caught during the execution of the \fBioctl\fR(2) function.
.RE

.sp
.ne 2
.na
\fB\fBEFAULT\fR \fR
.ad
.RS 11n
```



```
The request argument requires a data transfer to or from a buffer pointed to by
\fIarg\fR, but \fIarg\fR points to an illegal address.
.RE

.sp
.ne 2
.na
\fB\fBEINVAL\fR \fR
.ad
.RS 11n
The request or \fIarg\fR parameter is not valid for  this device, or field of
the dvpoll struct pointed by \fIarg\fR is not valid  (for example, when using
write/pwrite dp_nfds is greater than {OPEN_MAX}, or when using the DPPOLL ioctl
dp_nfds is greater than or equal to {OPEN_MAX}}.
.RE

.sp
.ne 2
.na
\fB\fBENXIO\fR \fR
.ad
.RS 11n
The \fBO_NONBLOCK\fR flag is set, the named file is a FIFO, the \fBO_WRONLY\fR
flag is set, and no process has the file open for reading; or the named file is
a character special or block special file and the device associated with this
special file does not exist.
.RE

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

.sp
.TS
box;
l l
l l .
ATTRIBUTE TYPE  ATTRIBUTE VALUE
Architecture    SPARC, x86
Interface Stability     Obsolete
MT-Level        Safe
.TE

.SH SEE ALSO
.sp
.LP
\fBopen\fR(2), \fBpoll\fR(2), \fBwrite\fR(2), \fBattributes\fR(5)
.SH NOTES
.sp
.LP
The \fB/dev/poll\fR API is particularly beneficial to applications that poll a
large number of file descriptors repeatedly.  Applications will exhibit the
best performance gain if the polled file descriptor list rarely change.
.sp
.LP
When using the \fB/dev/poll\fR driver, you should remove a closed file
descriptor from a monitored poll set. Failure to do so may result in a
\fBPOLLNVAL\fR \fBrevents\fR being returned for the closed file descriptor.
When a file descriptor is closed but not removed from the monitored set, and is
reused in subsequent open of a different device, you will be polling the device
associated with the reused file descriptor. In a multithreaded application,
careful coordination among threads doing close and \fBDP_POLL\fR ioctl is
recommended for consistent results.
.sp
```

```
387 .LP
388 The \fB/dev/poll\fR driver caches a list of polled file descriptors, which are
389 specific to a process. Therefore, the  \fB/dev/poll\fR file descriptor of a
390 process will be inherited by its child process, just like any other file
391 descriptors. But the child process will have very limited access through this
392 inherited \fB/dev/poll\fR file descriptor. Any attempt to write or do ioctl by
393 the child process will result in an \fBEACCES\fR error. The child process
394 should close the inherited  \fB/dev/poll\fR file descriptor and open its own if
395 desired.
396 .sp
397 .LP
398 The  \fB/dev/poll\fR driver does not yet support polling. Polling on a
399 \fB/dev/poll\fR file descriptor will result in \fBPOLLERR\fR being returned in
400 the \fBrevents\fR field of \fBpollfd\fR structure.
```

```
**********************************************************
    6502 Tue Sep 10 18:35:21 2013
new/usr/src/man/man7d/tsalarm.7d
4023 - Typo in file(1) manpage and various others
**********************************************************
   1 '\" te
   2 .\" Copyright (c) 2004, Sun Microsystems, Inc.
   3 .\" The contents of this file are subject to the terms of the Common Development
   4 .\" You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE or http:
   5 .\" When distributing Covered Code, include this CDDL HEADER in each file and in
   6 .TH TSALARM 7D "Sep 10, 2013"
   6 .TH TSALARM 7D "Mar 16, 2004"
   7 .SH NAME
   8 tsalarm \- Alarm device driver
   9 .SH SYNOPSIS
  10 .LP
  11 .nf
  12 tsalarm@0:ctl
  13 .fi

  15 .SH DESCRIPTION
  16 .sp
  17 .LP
  18 The \fBtsalarm\fR driver is a Multi-threaded, loadable non-STREAMS pseudo
  19 driver that manages ALOM alarms. The \fBtsalarm\fR driver provides an interface
  20 through which alarm relays can be controlled on SUNW,Netra-240 and
  21 SUNW,Netra-440 platforms.
  22 .SH HARDWARE INTERFACE
  23 .sp
  24 .LP
  25 The alarm hardware differs depending on platform. The Netra 240 and 440
  26 platforms features four dry contact alarm relays       which are controlled by
  27 ALOM. You can set each alarm to "on" or "off" by  using ioctl interfaces
  28 provided from the host. The four alarms are labeled as "critical," "major,"
  29 "minor," and "user." The user alarm is set by a user application depending on
  30 system condition. LED's in front of the box provide a visual indication of the
  31 four alarms. The number of alarms and their meanings/labels may vary across
  32 platforms.
  33 .SH IOCTLS
  34 .sp
  35 .LP
  36 The interface provided by the \fBtsalarm\fR driver comprises ioctls that enable
  37 applications to manipulate the alarm module. The alarm  module is accessed via
  38 two device nodes:  i) \fB/dev/lom\fR and \fB/dev/tsalarm:ctl\fR.
  39 .sp
  40 .LP
  41 The following ioctls are supported by  the \fB/dev/lom\fR and
  42 \fB/dev/tsalarm:ctl\fR devices:
  43 .sp
  44 .ne 2
  45 .na
  46 \fB\fBTSIOCALCTL - Turn an alarm on or off.\fR\fR
  47 .ad
  48 .sp .6
  49 .RS 4n
  50 The argument is a pointer to the \fBts_aldata_t/lom_aldata_t\fR structure. This
  51 structure is described below. \fBalarm_no member\fR is an integer which
  52 specifies the alarm to which the command is to be applied. The
  53 \fBalarm_state/state\fR structure member indicates the state to which the alarm
  54 should be set (where 0 == off). An error  (\fBEINVAL\fR) is returned if either
  55 an invalid alarm_no or invalid alarm_state is provided.
  56 .RE

  58 .sp
  59 .ne 2
  60 .na
```



```
  61 \fB\fBTSIOCALSTATE - Get the state of the alarms.\fR\fR
  62 .ad
  63 .sp .6
  64 .RS 4n
  65 The argument is a pointer to the \fBts_aldata_t/lom_aldata_t\fR structure. This
  66 structure is described below. \fBalarm_no member\fR is an integer which
  67 indicates the alarm to which the command will be applied. The
  68 \fBalarm_state\fR member holds the alarm's current state and is filled in by
  69 the driver. A zero indicates that the alarm is off. An error (\fBEINVAL\fR) is
  70 returned if an invalid alarm_no is provided. The structures and definitions for
  71 the  values  are defined below.
  72 .RE

  74 .sp
  75 .LP
  76 Alarm values:
  77 .sp
  78 .in +2
  79 .nf
  80 The following old style values are defined in <lom.io.h>

  82      #define ALARM_NUM_0         0 /* number of zero'th alarm */

  84      #define ALARM_NUM_1         1 /* number of first alarm */

  86      #define ALARM_NUM_2         2 /* number of second alarm */

  88      #define ALARM_NUM_3         3 /* number of third alarm */

  90   Alarm values defined in <lom.io.h>

  92      #define ALARM_OFF          0 /* Turn off alarm */

  94      #define ALARM_ON           1 /* Turn on alarm */
  95 .fi
  96 .in -2

  98 .sp
  99 .LP
 100 Alarm Data Structure:
 101 .sp
 102 .in +2
 103 .nf
 104 This structure is defined in <lom.io.h>

 106      typedef struct {

 108           int alarm_no;       /* alarm to apply command to */

 110           int alarm_state;    /* state of alarm (0 == off) */

 112      } ts_aldata_t;
 113 .fi
 114 .in -2

 116 .sp
 117 .LP
 118 Use the following LOM interfaces to get and set the alarms. These definitions
 119 are included in <lom_io.h>
 120 .sp
 121 .in +2
 122 .nf
 123     #define ALARM_CRITICAL         0 /* number of critical alarm */

 125     #define ALARM_MAJOR            1 /* number of major alarm */
```

```
127     #define ALARM_MINOR            2 /* number of minor alarm */

129     #define ALARM_USER             3 /* number of user alarm */
130 .fi
131 .in -2

133 .sp
134 .LP
135 The following alarm data structure is provided in <lom_io.h>:
136 .sp
137 .in +2
138 .nf
139 typedef struct {

141         int alarm_no;

143         int state;

145     } lom_aldata_t;
146 .fi
147 .in -2

149 .SH ERRORS
150 .sp
151 .LP
152 An \fBopen()\fR will fail if:
153 .sp
154 .ne 2
155 .na
156 \fBENXIO\fR
157 .ad
158 .RS 9n
159 The driver is not installed in the system.
160 .RE

162 .sp
163 .LP
164 An \fBioctl()\fR will fail if:
165 .sp
166 .ne 2
167 .na
168 \fBEFAULT\fR
169 .ad
170 .RS 10n
171 There was a hardware failure during the specified operation.
172 .RE

174 .sp
175 .ne 2
176 .na
177 \fBEINVAL\fR
178 .ad
179 .RS 10n
180 The alarm number specified is not valid or an invalid value was supplied.
181 .RE

183 .sp
184 .ne 2
185 .na
186 \fBENXIO\fR
187 .ad
188 .RS 10n
189 The driver is not installed in the system or the monitor callback routine could
190 not be scheduled.
191 .RE
```



```
193 .SH EXAMPLES
194 .sp
195 .in +2
196 .nf
197 How to set an alarm:

199     #include <sys/unistd.h>
200     #include <fcntl.h>
201     #include <stdio.h>
202     #include <lom_io.h>

204     #define LOM_DEVICE "/dev/lom"

206     int
207     main()
208     {
209        lom_aldata_t lld;
210        int fd = open(LOM_DEVICE, O_RDWR);

212        if (fd == -1) {
213                printf("Error opening device: %s\en", LOM_DEVICE);
213                printf("Error opening device: %s\n", LOM_DEVICE);
214                exit (1);
215        }

217        lld.alarm_no = ALARM_CRITICAL;   /* Set the critical alarm */
218        lld.state = ALARM_ON;   /* Set the alarm */

220        if (ioctl(fd, LOMIOCALCTL, (char *)&lld) != 0)
221                printf("Setting alarm failed");
222        else
223                printf("Alarm set successfully");

225        close(fd);

227     }
228 .fi
229 .in -2

231 .SH FILES
232 .sp
233 .ne 2
234 .na
235 \fB\fB/dev/lom\fR\fR
236 .ad
237 .sp .6
238 .RS 4n
239 LOM device.
240 .RE

242 .sp
243 .ne 2
244 .na
245 \fB\fB/dev/tsalarm:ctl\fR\fR
246 .ad
247 .sp .6
248 .RS 4n
249 Alarm control device.
250 .RE

252 .sp
253 .ne 2
254 .na
255 \fB\fB/platform/platform/kernel/drv/sparcv9/tsalarm\fR\fR
256 .ad
257 .sp .6
```

```
 258 .RS 4n
 259 Device driver module.
 260 .RE

 262 .sp
 263 .ne 2
 264 .na
 265 \fB\fB/platform/SUNW,Netra-240/kernel/drv/tsalarm.conf\fR\fR
 266 .ad
 267 .sp .6
 268 .RS 4n
 269 Driver configuration file.
 270 .RE

 272 .SH ATTRIBUTES
 273 .sp
 274 .LP
 275 See \fBattributes\fR(5) for descriptions of the following attributes:
 276 .sp

 278 .sp
 279 .TS
 280 box;
 281 c | c
 282 l | l .
 283 ATTRIBUTE TYPE  ATTRIBUTE VALUE
 284 _
 285 Architecture    SPARC
 286 .TE

 288 .SH SEE ALSO
 289 .sp
 290 .LP
 291 \fBattributes\fR(5)
 292 .sp
 293 .LP
 294 \fIWriting Device Drivers\fR
```

```
**********************************************************
   43986 Tue Sep 10 18:35:22 2013
new/usr/src/man/man7d/ugen.7d
4023 - Typo in file(1) manpage and various others
**********************************************************
   1 '\" te
   2 .\" Copyright (c) 2007, Sun Microsystems, Inc.  All Rights Reserved
   3 .\" The contents of this file are subject to the terms of the Common Development
   4 .\" You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE or http:
   5 .\" When distributing Covered Code, include this CDDL HEADER in each file and in
   6 .TH UGEN 7D "Sep 10, 2013"
   6 .TH UGEN 7D "Dec 3, 2008"
   7 .SH NAME
   8 ugen \- USB generic driver
   9 .SH SYNOPSIS
  10 .LP
  11 .nf
  12 \fBNode Name@unit-address\fR
  13 .fi

  15 .LP
  16 .nf
  17 \fB#include <sys/usb/clients/ugen/usb_ugen.h>\fR
  18 .fi

  20 .SH DESCRIPTION
  21 .sp
  22 .LP
  23 \fBugen\fR is a generic USBA (Solaris USB Architecture) compliant client
  24 character driver that presents USB devices to applications through a standard
  25 \fBopen\fR(2), \fBclose\fR(2), \fBread\fR(2), \fBwrite\fR(2),
  26 \fBaioread\fR(3C), \fBaiowrite\fR(3C) Unix interface. Uninterpreted raw data
  27 are transfered to and from the device via file descriptors created for each USB
  28 endpoint. Status is obtained by reading file descriptors created for endpoint
  29 and full device status.
  30 .sp
  31 .LP
  32 \fBugen\fR supports control, bulk, isochronous and interrupt (in and  out)
  33 transfers. \fBlibusb\fR(3LIB) uses \fBugen\fR to access devices that do not
  34 contain drivers (such as digital cameras and PDAs). Refer to
  35 \fB/usr/sfw/share/doc/libusb/libusb.txt\fR for details.
  36 .SH BINDING
  37 .sp
  38 .LP
  39 In general, no explicit binding of the \fBugen\fR driver is necessary because
  40 \fBusb_mid\fR(7D) is the default driver for devices without a class or vendor
  41 unique driver. \fBusb_mid\fR(7D) creates the same logical device names as
  42 \fBugen\fR, but only if no child interfaces are explicitly bound to \fBugen\fR.
  43 If it is necessary to bind \fBugen\fR explicitly to a device or interface, the
  44 following section explains the necessary steps.
  45 .sp
  46 .LP
  47 \fBugen\fR can bind to a device with one or more interfaces in its entirety, or
  48 to a single interface of that device. The binding type depends on information
  49 that is passed to \fBadd_drv\fR(1M) or \fBupdate_drv\fR(1M).
  50 .sp
  51 .LP
  52 An \fBadd_drv\fR(1M) command binds \fBugen\fR to a list of device types it is
  53 to control. \fBupdate_drv\fR(1M) adds an additional device type to the list of
  54 device types being managed by the driver.
  55 .sp
  56 .LP
  57 Names used to bind drivers can be found in \fB/var/adm/messages\fR. When a
  58 device is on-lined after hot insertion, and no driver is found, there will be
  59 an entry containing:
  60 .sp
```



```
  61 .in +2
  62 .nf
  63 USB 2.0 device (usb<vid>,<pid>)...
  64 .fi
  65 .in -2

  67 .sp
  68 .LP
  69 where vid is the USB vendor identifier in hex and pid is the  product
  70 identifier in hex supplied by the device  descriptor \fBusb_dev_descr\fR(9S).
  71 .sp
  72 .LP
  73 When using ugen for the first time, you must add the  driver utilizing
  74 \fBadd_drv\fR(1M), using a command of the following form:
  75 .sp
  76 .in +2
  77 .nf
  78 Assuming that the vid is 472 and pid is b0b0:

  80 add_drv -n -m '* <device perms> <owner> <group>'
  81    -i '"usb472,b0b0"' ugen
  82 .fi
  83 .in -2

  85 .sp
  86 .LP
  87 If the command fails with:
  88 .sp
  89 .in +2
  90 .nf
  91 (ugen) already in use as a driver or alias.
  92 .fi
  93 .in -2

  95 .sp
  96 .LP
  97 \&...add the device using \fBupdate_drv\fR(1M):
  98 .sp
  99 .in +2
 100 .nf
 101 update_drv -a -m '* <device perms> <owner> <group>'
 102    -i '"usb472,b0b0"' ugen
 103 .fi
 104 .in -2

 106 .sp
 107 .LP
 108 This binds \fBugen\fR to the entire device.
 109 .sp
 110 .LP
 111 If ugen only binds to one interface of the device, use the following
 112 driver_alias instead of usb<vid>,<pid>:
 113 .sp
 114 .in +2
 115 .nf
 116        usbif<vid>,<pid>.config<cfg value>.<interface number>
 117 .fi
 118 .in -2

 120 .sp
 121 .LP
 122 where cfg value is the value of bConfigurationValue in the configuration
 123 descriptor (\fBusb_cfg_descr\fR(9S)). For example "usbif1234,4567.config1.0."
 124 .sp
 125 .LP
 126 Note that you can use update_drv to  also  remove  bindings. Please see
```

```
\fBupdate_drv\fR(1M) for more information.
.sp
.LP
After a successful add_drv or update_drv, remove the device and reinsert. Check
with the \fBprtconf\fR(1M) -D option to determine if \fBugen\fR is successfully
bound to the device and the nodes created in /dev/usb/<vid>.<pid> (see below).
.sp
.LP
An example showing how to bind a child device representing interface 0 of
configuration 1 of a composite device follows:
.sp
.in +2
.nf
update_drv -a -m '* 0666 root sys'
    -i '"usbif472,b0b0.config1.0"' ugen
.fi
.in -2

.sp
.LP
Note that you can completely uninstall the \fBugen\fR driver and delete it from
the system by doing:
.sp
.in +2
.nf
pkgrm SUNWugen
.fi
.in -2

.sp
.LP
Any \fBpkgadd\fR of SUNWugen after the \fBpkgrm\fR reactivates any pre-existing
ugen driver device-bindings.
.sp
.LP
Any pre-existing ugen driver device-bindings are preserved across operating
system upgrades.
.SH LOGICAL DEVICE NAME FORMAT
.sp
.LP
For each device or child device it manages, \fBugen\fR creates one logical
device name for device-wide status and one logical device name for endpoint 0.
\fBugen\fR also creates logical device names for all other endpoints within the
device node's binding scope (interface or device), plus logical device names
for their status.
.sp
.LP
If separate \fBugen\fR instances control different interfaces of the same
device, the device-wide status and endpoint logical device names created for
each instance will share access to the same source or endpoint pipes. For
example, a device with two interfaces, each operated by their own \fBugen\fR
instance, will show \fBendpoint0\fR as \fBif0cntrl0\fR to the first interface,
and will show it as \fBif1cntrl0\fR to the second interface. Both of these
logical device names share \fBendpoint0\fR. Likewise for the same device,
\fBugen\fR makes the device-wide status available as \fBif0devstat\fR to the
first interface and as \fBif1devstat\fR to the second interface.
\fBif0devstat\fR and \fBif1devstat\fR both return the same data.
.sp
.LP
Any \fBugen\fR logical device name can be held open by only one user at a time,
regardless of whether the\fB O_EXCL\fR flag passed to \fBopen\fR(2). When a
single pipe or data source is shared by multiple logical device names, such as
if[0,1]cntrl0 or if[0,1]devstat above, more than one logical device name
sharing the pipe or data source can be open at a time.  However, only one user
may access the shared pipe or data source at a time, regardless of the logical
device name used for access.
```



```
.sp
.LP
When \fBugen\fR is bound to an entire device, the following logical device
names are created (each on a single line). \fIN\fR represents the instance
number of the device type.
.sp
.in +2
.nf
Endpoint 0 (default endpoint):

        /dev/usb/<vid>.<pid>/<N>/cntrl0
        /dev/usb/<vid>.<pid>/<N>/cntrl0stat

    For example:

        /dev/usb/472.b0b0/0/cntrl0
        /dev/usb/472.b0b0/0/cntrl0stat

Configuration index 1, Endpoints > 0, alternate 0:

        /dev/usb/<vid>.<pid>/<N>/if<interface#>
                                <in|out|cntrl><endpoint#>
        /dev/usb/<vid>.<pid>/<N>/if<interface#>
                                <in|out|cntrl><endpoint#>stat

    For example:

        /dev/usb/472.b0b0/0/if0in1
        /dev/usb/472.b0b0/0/if0in1stat

Configuration index 1, Endpoints > 0, alternate > 0:

        /dev/usb/<vid>.<pid>/<N>/if<interface#>.
                                <alternate><in|out|cntrl><endpoint#>
        /dev/usb/<vid>.<pid>/<N>/if<interface#>.
                                <alternate<in|out|cntrl><endpoint#>stat

    For example:

        /dev/usb/472.b0b0/0/if0.1in3
        /dev/usb/472.b0b0/0/if0.1in3stat

Configuration index> 1, Endpoints > 0, alternate 0:
        /dev/usb/<vid>.<pid>/<N>/cfg<value>if<interface#>
                                <in|out|cntrl><endpoint#>
        /dev/usb/<vid>.<pid>/<N>/cfg<value>if<interface#>
                                <in|out|cntrl><endpoint#>stat

    For example:

       /dev/usb/472.b0b0/0/cfg2if0in1
       /dev/usb/472.b0b0/0/cfg2if0in1stat

    Note that the configuration value from the configuration
    descriptor indexed by the configuration index is used in
    the node name and not the configuration index itself.

Configuration index> 1, Endpoints > 0, alternate > 0:
        /dev/usb/<vid>.<pid>/<N>/cfg<value>if<interface#>.
                                <alternate<in|out|cntrl><endpoint#>
        /dev/usb/<vid>.<pid>/<N>/cfg<value>if<interface#>.
                                <alternate<in|out|cntrl><endpoint#>stat

    For example:

        /dev/usb/472.b0b0/0/cfg2if0.1in1
```

```
259         /dev/usb/472.b0b0/0/cfg2if0.1in1stat

261   Device status:

263         /dev/usb/<vid>.<pid>/<N>/devstat

265     For example:

267         /dev/usb/472.b0b0/0/devstat
268 .fi
269 .in -2

271 .sp
272 .LP
273 When \fBugen\fR is bound to a single device interface, the following logical
274 device nodes are created:
275 .sp
276 .in +2
277 .nf
278 Endpoint 0 (default endpoint):

280          /dev/usb/<vid>.<pid>/<N>/if<interface#>cntrl0
281          /dev/usb/<vid>.<pid>/<N>/if<interface#>cntrl0stat

283     For example:

285          /dev/usb/472.b0b0/0/if0cntrl0
286          /dev/usb/472.b0b0/0/if0cntrl0stat

288 Device status:
289         /dev/usb/<vid>.<pid>/<N>/if<interface#>devstat

291     For example:
292         /dev/usb/472.b0b0/0/if0devstat
293 .fi
294 .in -2

296 .sp
297 .LP
298 The format for all other logical device names is identical to the format used
299 when \fBugen\fR is bound to the entire device.
300 .sp
301 .LP
302 Opening the endpoint of a different configuration or different alternate
303 interface will cause an implicit change of configuration or a switch to an
304 alternate interface. A configuration change is prohibited when any non-zero
305 endpoint device nodes are open. An alternate interface switch is prohibited if
306 any endpoint in the same interface is open.
307 .SH HOT-PLUGGING
308 .sp
309 .LP
310 A device may be hot-removed at any time. Following hot-removal, the device
311 status changes to USB_DEV_STAT_DISCONNECTED, the status of open endpoints
312 change to USB_LC_STAT_DISCONNECTED upon their access, and all subsequent
313 transfer requests fail. Endpoints are reactivated by first reinserting the
314 device and then closing and reopening all endpoints that were open when the
315 device was disconnected.
316 .SH CPR (CHECKPOINT/RESUME)
317 .sp
318 .LP
319 CPR (Checkpoint/Resume) may be initiated at any time and is treated similarly
320 to a hot-removal. Upon successful suspend and resume, all subsequent transfer
321 requests fail as an indication to the application to reinitialize. Applications
322 should close and reopen all endpoints to reinstate them. All endpoint and
323 device status on Resume (before close and reopen) is USB_LC_STAT_SUSPENDED. A
324 system suspend will fail while \fBugen\fR is performing a transfer.
```



```
325 .SH DEVICE POWER MANAGEMENT
326 .sp
327 .LP
328 Devices which support remote wakeup can be power managed when they have no open
329 logical device nodes. When an application opens the first logical device node
330 of a device, that application should assume that a reinitialization of device
331 state is required.
332 .SH DEVICE STATUS MANAGEMENT
333 .sp
334 .LP
335 Applications can monitor device status changes by reading the device status
336 from the device status logical name. When opened without O_NONBLOCK and
337 O_NDELAY, all reads from that file descriptor (with the exception of the the
338 intial read that follows the open) block until a device status change occurs.
339 Calls to read will always return immediately if opened with \fBO_NONBLOCK\fR or
340 \fBO_NDELAY\fR. Nonblocking calls to read which have no data to return, return
341 no error and zero bytes read.
342 .sp
343 .LP
344 Device statuses are:
345 .sp
346 .ne 2
347 .na
348 \fBUSB_DEV_STAT_ONLINE\fR
349 .ad
350 .RS 29n
351 Device is available.
352 .RE

354 .sp
355 .ne 2
356 .na
357 \fBUSB_DEV_STAT_DISCONNECTED\fR
358 .ad
359 .RS 29n
360 Device has been disconnected.
361 .RE

363 .sp
364 .ne 2
365 .na
366 \fBUSB_DEV_STAT_RESUMED\fR
367 .ad
368 .RS 29n
369 Device has been resumed, however, endpoints which were open on suspend have not
370 yet been closed and reopened.
371 .RE

373 .sp
374 .ne 2
375 .na
376 \fBUSB_DEV_STAT_UNAVAILABLE\fR
377 .ad
378 .RS 29n
379 Device has been reconnected, however, endpoints which were open on disconnect
380 have not yet been closed and reopened.
381 .RE

383 .sp
384 .LP
385 The following code reads the device status device logical name:
386 .sp
387 .in +2
388 .nf
389 int fd;
390 int status;
```

```
392 if ((fd = open("/dev/usb/472.b0b0/0/devstat",
393     O_RDONLY)) < 0)     {
394         /* handle error */
395 }
_____unchanged_portion_omitted_

460 data_xfered = write(ep1_data_fd, request, sizeof (request));

462 /* An error occured during the data transfer. */
463 if (data_xfered != sizeof (request)) {

465         /* Read status file descriptor for details on failure. */
466         if (read(ep1_stat_fd, (int *)&status, sizeof (status)) !=
467             sizeof (status)) {
468                 status = USB_LC_STAT_UNSPECIFIED_ERR;
469         }

471         /* Take appropriate action. */
472         switch (status) {
473         case USB_LC_STAT_STALL:
474                 printf ("Endpoint stalled.\en");
475                 break;
476         case ...
477                 ...
478         }

480    }
481 .fi
482 .in -2

484 .SH CONTROL TRANSFERS
485 .sp
486 .LP
487 The control endpoint is typically used to set up the device and to query device
488 status or configuration.
489 .sp
490 .LP
491 Applications requiring I/O on a control endpoint should open the corresponding
492 logical device name and use regular UNIX I/O system calls. For example:
493 \fBread\fR(2), \fBwrite\fR(2), \fBaioread\fR(3C) and \fBaiowrite\fR(3C).
494 \fBpoll\fR(2) is not supported on control endpoints.
495 .sp
496 .LP
497 A control endpoint must be opened with \fBO_RDWR\fR since it is bidirectional.
498 It cannot be opened with \fBO_NONBLOCK\fR or \fBO_NDELAY\fR.
499 .sp
500 .LP
501 For example:
502 .sp
503 .in +2
504 .nf
505 fd = open("/dev/usb/472.b0b0/0/cntrl0", O_RDWR);

507 .fi
508 .in -2

510 .sp
511 .in +2
512 .nf
513 fdstat = open("/dev/usb/472.b0b0/0/cntrl0stat", O_RDONLY);
514 .fi
515 .in -2

517 .sp
518 .LP
```



```
519 Control endpoints can be read and written. A \fBread\fR operation receives data
520 \fBfrom\fR the device and a \fBwrite\fR operation sends data \fBto\fR the
521 device.
522 .sp
523 .LP
524 To perform a control-IN transfer, perform a \fBwrite\fR(2) of USB setup data
525 (see section 9.3 of the \fIUSB 1.1\fR or \fI2.0\fR specifications) followed by
526 a \fBread\fR(2) on the same control endpoint to fetch the desired data. For
527 example:
528 .sp
529 .in +2
530 .nf
531 void init_cntrl_req(
532     uchar_t *req, uchar_t bmRequestType, uchar_t bRequest,
533     ushort_t wValue, ushort_t wIndex, ushort_t wLength) {
534         req[0] = bmRequestType;
535         req[1] = bRequest;
536         req[2] = 0xFF & wValue;
537         req[3] = 0xFF & (wValue >> 8);
538         req[4] = 0xFF & wIndex;
539         req[5] = 0xFF & (wIndex >> 8);
540         req[6] = 0xFF & wLength;
541         req[7] = 0xFF & (wLength >> 8);
542  }

544  ....


547         uchar_t dev_descr_req[8];
548         usb_dev_descr_t descr;

550         init_cntrl_req(dev_descr_req,
551             USB_DEV_REQ_DEV_TO_HOST, USB_REQ_GET_DESCR,
552             USB_DESCR_TYPE_SETUP_DEV, 0, sizeof (descr));

554         count = write(fd, dev_descr_req, sizeof (dev_descr_req));
555         if (count != sizeof (dev_descr_req)) {
556                 /* do some error recovery */
557                 ...
558         }

560         count = read(fd, &descr, sizeof (descr));
561         if (count != sizeof (descr)) {
562                 /* do some error recovery */
563         }
564 .fi
565 .in -2

567 .sp
568 .LP
569 The application can issue any number of reads to read data received on a
570 control endpoint. \fBugen\fR successfully completes all reads, returning the
571 number of bytes transferred. Zero is returned when there is no data to
572 transfer.
573 .sp
574 .LP
575 If the \fBread\fR/\fBwrite\fR fails and returns \fB-1\fR, you can access the
576 endpoint’s status device logical name for precise error information:
577 .sp
578 .in +2
579 .nf
580         int status;

582         count = read(fdstat, &status, sizeof (status));
583         if (count == sizeof (status)) {
584                 switch (status) {
```

```
585                 case USB_LC_STAT_SUSPENDED:
586                 case USB_LC_STAT_DISCONNECTED:
587                         /* close all endpoints */
588                         ...
589                         break;
590                 default:
591                         ...
592                         break;
593                 }
594         }
595 .fi
596 .in -2

598 .sp
599 .LP
600 Refer to the ERRORS section for all possible error values.
601 .sp
602 .LP
603 To perform a control-OUT transfer, send in a single transfer, the USB setup
604 data followed by any accompanying data bytes.
605 .sp
606 .in +2
607 .nf
608     /* 1st 8 bytes of wbuf are setup. */
609     init_cntrl_req(wbuf, .......);

611     /* Data bytes begin at byte 8 of wbuf. */
612     bcopy(data, &wuf[8], sizeof (data));

614     /* Send it all in a single transfer. */
615     count = write(fd, wbuf, sizeof (wbuf));
616 .fi
617 .in -2

619 .sp
620 .LP
621 A \fBwrite\fR(2) returns the number of bytes (both setup and data) actually
622 transferred, (whether or not the \fBwrite\fR is completely successful),
623 provided that some data is actually transferred. When no data is transferred,
624 \fBwrite\fR(2) returns \fB-1\fR. Applications can read the corresponding
625 endpoint status to retrieve detailed error information. Note that it is an
626 error to specify a size different than:
627 .sp
628 .LP
629 (number of data bytes + number of setup bytes).
630 .sp
631 .LP
632 Here is a more extensive example which gets all descriptors of a device
633 configuration.  For sake of brevity, uninteresting parts are omitted.
634 .sp
635 .in +2
636 .nf
637    #include <sys/usb/usba.h>
638    #include <sys/usb/clients/ugen/usb_ugen.h>

640    uchar_t *config_cloud;
641    uchar_t *curr_descr;

643    uchar_t *bytes;

645    int curr_descr_len;
646    int curr_descr_type;

648    usb_cfg_descr_t cfg_descr;
649    usb_if_descr_t if_descr;
650    usb_ep_descr_t ep_descr;
```



```
652    /* See 9.13 of USB 2.0 spec for ordering. */
653    static char *pipetypes[] = {
654         "Control", "Isochronous", "Bulk", "Interrupt"
655    };

657    /*
658     * Setup to send a request to read just the config descriptor.  The
659     * size of the whole cloud, containing all cfg, interface, endpoint,
660     * class and vendor-specific descriptors, will be returned as part of
661     * the config descriptor.
662     */
663    init_cntrl_req(&setup_data, USB_DEV_REQ_DEV_TO_HOST, USB_REQ_GET_DESCR,
664                    USB_DESCR_TYPE_SETUP_CFG, 0, USB_CFG_DESCR_SIZE);

666    /*
667     * Write setup data. USB device will prepare to return the whole
668     * config cloud as a response to this. We will read this separately.
669     */
670    count = write(ctrl_fd, &setup_data, sizeof (setup_data));
671    if (count != sizeof (setup_data)) {
672            /* Error recovery. */
673    } else {
674            count = read(ctrl_fd, &cfg_descr, USB_CFG_DESCR_SIZE);
675            if (count != USB_CFG_DESCR_SIZE) {
676                    /* Error recovery. */
677            }
678    }

680    /* USB data is little endian. */
681    bytes = (uchar_t *)(&cfg_descr.wTotalLength);
682    totalLength = bytes[0] + (bytes[1] << 8);

684    /*
685     * The size of the whole cloud is in the bLength field.  Set up
686     * to read this amount of data, to get the whole cloud.
687     */
688    config_cloud = malloc(totalLength);

690    init_cntrl_req(&setup_data, USB_DEV_REQ_DEV_TO_HOST, USB_REQ_GET_DESCR,
691                    USB_DESCR_TYPE_SETUP_CFG, 0, totalLength);

693    count = write(ctrl_fd, &setup_data, sizeof (setup_data));
694    if (count != sizeof (setup_data)) {
695            /* Error recovery. */
696    } else {
697            count = read(ctrl_fd, config_cloud, totalLength);
698            if (count != totalLength) {
699                    /* Error recovery. */
700             }
701    }

703    /* Got the data.  Now loop, dumping out the descriptors found. */

705    curr_descr = config_cloud;
706    offset = 0;
707    while (offset < totalLength) {

709            /* All descr have length and type at offset 0 and 1 */
710            curr_descr_len = curr_descr[0];
711            curr_descr_type = curr_descr[1];

713            switch (curr_descr_type) {
714            case USB_DESCR_TYPE_CFG:

716                    /*
```

```
                     * Copy data into separate structure, needed for
                     * proper alignment of all non char fields.  Note:
                     * non-char fields of all descriptors begin on aligned
                     * boundaries.  The issue is that some structures may
                     * be adjacent to others which have an odd-numbered
                     * byte size, and may thus start on an odd-numbered
                     * boundary.  */
                    bcopy(curr_descr, &cfg_descr, curr_descr_len);

                    /* Remember to read any words in endian-neutral way. */

                    (void) printf("\enConfig %d found.\en",
                        cfg_descr.bConfigurationValue);
                    break;

            case USB_DESCR_TYPE_IF:
                    bcopy(curr_descr, &if_descr, curr_descr_len);
                    (void) printf("\en\etInterface %d, Alt %d found.\en",
                        if_descr.bInterfaceNumber,
                        if_descr.bAlternateSetting);
                    break;

            case USB_DESCR_TYPE_EP:
                    bcopy(curr_descr, &ep_descr, curr_descr_len);
                    (void) printf("\en\et\etEndpoint %d (%s-%s) found.\en",
                        (ep_descr.bEndpointAddress & USB_EP_NUM_MASK),
                        (pipetypes[
                            ep_descr.bmAttributes & USB_EP_ATTR_MASK]),
                        ((ep_descr.bEndpointAddress &
                        USB_EP_DIR_IN) ? "IN" : "OUT"));
                     break;

             default:
                     (void) printf(
                         "\en\et\et\etOther descriptor found.  Type:%d\en",
                         curr_descr_type);
                     break;
             }

             offset += curr_descr_len;
             curr_descr = &config_cloud[offset];
    }
.fi
.in -2

.SH INTERRUPT-IN TRANSFERS
.sp
.LP
Applications requiring data from an interrupt-IN endpoint should open the
corresponding logical device name and use \fBread\fR(2), \fBaioread\fR(3C) and
\fBpoll\fR(2) system calls.
.sp
.LP
An interrupt-IN endpoint must be opened with \fBO_RDONLY\fR. It can also be
opened using \fBO_NONBLOCK\fR or \fBO_NDELAY\fR if desired.
.sp
.in +2
.nf
fd = open("/dev/usb/472.b0b0/0/if0in1", O_RDONLY);
.fi
.in -2

.sp
.in +2
.nf
fdstat = open("/dev/usb/472.b0b0/0/if0in1stat", O_RDONLY);
```



```
.fi
.in -2

.sp
.LP
\fBugen\fR starts polling interrupt\(emIN endpoints immediately upon opening
them and stops polling them upon closure. (Polling refers to interrogation of
the device by the driver and should not be confused with \fBpoll\fR(2), which
is an interrogation of the driver by the application.)
.sp
.LP
A \fBread\fR(2) of an endpoint opened with the \fBO_NONBLOCK\fR or
\fBO_NDELAY\fR flags set will not block when there is insufficient data
available to satisfy the request. The \fBread\fR simply returns what it can
without signifying any error.
.sp
.LP
Applications should continuously check for and consume interrupt data.
\fBugen\fR enables buffering of up to one second of incoming data. In case of
buffer overflow, \fBugen\fR stops polling the interrupt-IN endpoint until the
application consumes all the data. In this case, a \fBread\fR(2) of an empty
buffer returns \fB-1\fR, sets the endpoint status to
\fBUSB_LC_STAT_INTR_BUF_FULL\fR (to indicate that the buffer had been full and
polling had been stopped) and causes \fBugen\fR to start polling the endpoint
again. To retrieve the status, the application can open and read the
corresponding endpoint’s status device logical name.
.sp
.in +2
.nf
for (;;) {
        count = read(fd, buf, sizeof(buf));
        if (count == -1) {
                int cnt, status;

                cnt = read(fdstat, &status, sizeof (status));
                if (cnt == -1) {
                        /* more error recovery here */
                } else {
                        switch (status) {
                        case USB_LC_STAT_INTR_BUF_FULL:
                                ...
                                break;
                        default:
                                ...
                                break;
                        }
                }
           }
           /* process the data */
           ....
        }
.fi
.in -2

.sp
.LP
\fBugen\fR will never drop data. However, the device may drop data if the
application cannot read it at the rate that it is produced.
.sp
.LP
Applications requiring unbuffered data from an interrupt-IN endpoint should
open the associated status endpoint with O_RDWR before opening the associated
interrupt-IN endpoint and write a control byte with USB_EP_INTR_ONE_XFER set.
All other bits are reserved and should be 0.
.sp
```

```
.LP
"One transfer" mode will persist until disabled explicitly after the associated
interrupt-IN endpoint has been closed by writing a control byte with
USB_EP_INTR_ONE_XFER cleared.
.sp
.LP
"One transfer" mode is implicitly disabled when the status/control endpoint is
closed.
.sp
.LP
Attempts to change the "one transfer" mode while the endpoint is open will
result in \fBEINVAL\fR.
.sp
.LP
An application can open multiple interrupt-IN endpoints and can call
\fBpoll\fR(2) to monitor the availability of new data. (Note: poll works with
interrupt-IN data endpoints, not their status endpoints.)
.sp
.in +2
.nf
        struct pollfd pfd[2];

        bzero(pfd, sizeof (pfd));
        pfd[0].fd = fd1;    /* fd1 is one interrupt-IN endpoint. */
        pfd[0].events = POLLIN;
        pfd[1].fd = fd2;    /* fd2 is another interrupt-IN endpoint. */
        pfd[1].events = POLLIN;

        for (;;) {
                poll(pfd, 2, -1);

                if (pfd[0].revents & POLLIN) {
                        count = read(fd1, buf, sizeof (buf));
                        ....
                }
                if (pfd[1].revents & POLLIN) {
                        count = read(fd2, buf, sizeof (buf));
                        ....
                }
        }
.fi
.in -2

.sp
.LP
You can monitor the device status endpoint via \fBpoll\fR(2) concurrently with
the multiple interrupt-IN endpoints.  Simply add another pollfd element to the
pfd array in the previous code example, and initialize the new element's
\fBfd\fR field with the file  descriptor  of the device status endpoint (opened
without O_NONBLOCK  or  O_NDELAY).  Set the new element's event field to POLLIN
like the other elements. Note that only interrupt-IN endpoints and the device
status endpoint can be monitored using \fBpoll\fR(2).
.SH INTERRUPT-OUT TRANSFERS
.sp
.LP
Applications requiring output on an interrupt-OUT endpoint can open the
corresponding  logical  device name and perform regular UNIX I/O  system calls
such as \fBwrite\fR(2) and \fBaiowrite\fR(3C).
.sp
.LP
An interrupt-OUT endpoint must be opened with O_WRONLY.
.sp
.in +2
.nf
fd = open("/dev/usb/472.b0b0/0/if0out3", O_WRONLY);
```



```
fdstat = open("/dev/usb/472.b0b0/0/if0out3stat", O_RDONLY);


.fi
.in -2

.sp
.LP
Data can be written to an interrupt-OUT endpoint as follows:
.sp
.in +2
.nf
      count = write(fd, buf, sizeof (buf)):
      if (count == -1) {
            /* error recovery */
      }
.fi
.in -2

.SH BULK TRANSFERS
.sp
.LP
Applications requiring I/O on a bulk endpoint can open the corresponding
logical device name and perform regular UNIX I/O system calls. For example:
\fBread\fR(2), \fBwrite\fR(2), \fBaioread\fR(3C) and \fBaiowrite\fR(3C).
\fBpoll\fR(2) is not supported on bulk endpoints.
.sp
.LP
A bulk endpoint must be opened with \fBO_RDONLY\fR or \fBO_WRONLY\fR and cannot
be opened with \fBO_NONBLOCK\fR or \fBO_NDELAY:\fR
.sp
.in +2
.nf
fd = open("/dev/usb/472.b0b0/0/if0in2", O_RDONLY);
.fi
.in -2

.sp
.in +2
.nf
fdstat = open("/dev/usb/472.b0b0/0/if0in2stat", O_RDONLY);
.fi
.in -2

.sp
.LP
Data can be read from a bulk-IN endpoint as follows:
.sp
.in +2
.nf
        count = read(fd, buf, sizeof (buf)):
        if (count == -1) {
                /* error recovery */
        }

 Data can be written to a bulk-OUT endpoint as follows:

        count = write(fd, buf, sizeof (buf)):
        if (count == -1) {
                /* error recovery */
        }
.fi
.in -2

.SH ISOCHRONOUS TRANSFERS
.sp
```

```
 981 .LP
 982 Applications requiring I/O on an isochronous endpoint can open the
 983 corresponding logical device name and perform regular UNIX I/O system calls
 984 such as \fBread\fR(2), \fBwrite\fR(2), \fBpoll\fR(2), \fBaioread\fR(3C) and
 985 \fBaiowrite\fR(3C). An isochronous endpoint must be opened with \fBO_RDWR\fR.
 986 .sp
 987 .in +2
 988 .nf
 989 fd = open("/dev/usb/472.b0b0/0/if0.3in2", O_RDWR);

 991 fdstat = open("/dev/usb/472.b0b0/0/if0.3in2stat", O_RDONLY);
 992 .fi
 993 .in -2

 995 .sp
 996 .LP
 997 Applications can use the status logical name to retrieve the state of the
 998 isochronous data endpoint, including details on why the most recent transfer
 999 failed.
1000 .sp
1001 .LP
1002 Applications have the flexibility to specify the number of isochronous packets
1003 and the size of individual packets they want to transfer. Applications should
1004 use the following data structures to exchange isochronous packet information
1005 with the \fBugen\fR driver:
1006 .sp
1007 .in +2
1008 .nf
1009 typedef struct ugen_isoc_pkt_descr {
1010                 /*
1011                 * Set by the application, for all isochro.
1012                 * requests, to the num. of bytes to xfer
1013                 * in a packet.
1014                 */
1015               ushort_t        dsc_isoc_pkt_len;

1017                 /*
1018                 * Set by ugen to actual num. of bytes sent/received
1019                 * in a packet.
1020                         */
1021       ushort_t        dsc_isoc_pkt_actual_len;

1023                 /*
1024                 * Per pkt. status set by ugen driver both for the
1025                 * isochronous IN and OUT requests. Application can
1026                 * use USB_LC_STAT_* to parse the status.
1027                 */
1028      int     dsc_isoc_pkt_status;
1029  } ugen_isoc_pkt_descr_t;

1031  typedef struct ugen_isoc_req_head {
1032    /* pkt count of the isoc request */
1033     int req_isoc_pkts_count;

1035   /* pkt descriptors */
1036   ugen_isoc_pkt_descr_t req_isoc_pkt_descrs[1];
1037  } ugen_isoc_req_head_t;
1038 .fi
1039 .in -2

1041 .sp
1042 .LP
1043 \fBreq_isoc_pkts_count\fR is limited by the capability of the USB host
1044 controller driver. The current upper bound for the \fBuhci\fR and \fBohci\fR
1045 drivers is 512. The upper bound for the \fBehci\fR driver is 1024.
1046 .sp
```



```
1047 .LP
1048 For an isochronous-IN endpoint, applications must first use the
1049 \fBugen_isoc_req_head_t\fR structure followed by \fBugen_isoc_pkt_descr_t\fR to
1050 write packet request information to the \fBugen\fR node. The \fBugen\fR driver
1051 then checks the validity of the request. If it is valid, the driver immediately
1052 begins isochronous polling on the IN endpoint and applications can proceed with
1053 \fBread\fR(2) of the data on the isochronous-IN endpoint. Upon successful
1054 return of \fBread\fR(2), isochronous packet descriptors (whose
1055 \fBdsc_isoc_pkt_actual_len\fR and \fBdsc_isoc_pkt_status\fR fields were filled
1056 by the driver) are returned, followed by the request's device payload data.
1057 .sp
1058 .LP
1059 Applications should continuously check for and consume isochronous data. The
1060 \fBugen\fR driver enables buffering of up to eight seconds of incoming data for
1061 full-speed isochronous endpoint, one second of data for high-speed isochronous
1062 endpoints who request one transaction per microframe and 1/3 of a second of
1063 incoming data for high-speed high-bandwidth isochronous endpoints who request
1064 three transactions per microframe. In case of buffer overflow, \fBugen\fR
1065 discards the oldest data.
1066 .sp
1067 .LP
1068 The isochronous-IN polling can only be stopped by a \fBclose\fR(2) associated
1069 file descriptor. If applications want to change packet information, they must
1070 first \fBclose\fR(2) the endpoint to stop the isochronous-IN polling, then
1071 \fBopen\fR(2) the endpoint and \fBwrite\fR(2) new packets request.
1072 .sp
1073 .LP
1074 The following example shows how to read an isochronous-IN endpoint:
1075 .sp
1076 .in +2
1077 .nf
1078                 #include  <sys/usb/clients/ugen/usb_ugen.h>

1080                 char *buf, *p;
1081                 ushort_t  pktlen;
1082                 int pktcnt, i;
1083                 int len;
1084                 ugen_isoc_req_head_t *req;
1085                 ugen_isoc_pkt_descr_t *pktdesc;
1086                 char rdbuf[5000];

1088                 pktcnt = 4; /* 4 packets in this request */
1089
1090                 len = sizeof(int) +
1091                         sizeof(ugen_isoc_pkt_descr_t) * pktcount;

1093                 buf = malloc(len);
1094                 if (!buf) {
1095                         /* Error recovery. */
1096             }

1098                 req = (ugen_isoc_req_head_t *)buf;
1099                 req->req_isoc_pkts_count = pktcnt;

1101                 pktdesc = (ugen_isoc_pkt_descr_t *)
1102                         (req->req_isoc_pkt_descrs);

1104      for (i = 0; i < pktcnt; i++) {
1105                                 /*
1106                                 * pktlen should not exceed xfer
1107                                 * capability of an endpoint
1108                                 */
1109                                 pktdesc[i].dsc_isoc_pkt_len = pktlen;

1111                                 pktdesc[i].dsc_isoc_pkt_actual_len = 0;
1112                                 pktdesc[i].dsc_isoc_pkt_status = 0;
```

```
1113            }

1115                        /*
1116                        * write request info to driver and len must
1117                        * be exactly the sum of
1118                        * sizeof(int) + sizeof(ugen_isoc_pkt_descr_t) * pktcnt.
1119                        * Otherwise, an error is returned.
1120                        */
1121                        if (write(fd, buf, len) < 0) {
1122                                /* Error recovery. */
1123            }

1125                        /*
1126                        * Read length should be sum of all pkt descriptors
1127                        * length + payload data length of all pkts
1128                        * (sizeof(ugen_isoc_pkt_descr_t) + pktlen) * pktcnt
1129                        */
1130                        if (read(fd, rdbuf, (sizeof(ugen_isoc_pkt_descr_t) +
1131                pktlen) * pktcnt) < 0) {
1132                                /* Error recovery. */
1133            }

1135                        pktdesc = (ugen_isoc_pkt_descr_t *) rdbuf;

1137                        /* points to payload beginning */
1138                        p = rdbuf + pktcnt * sizeof(ugen_isoc_pkt_descr_t);

1140                        for (i = 0; i < pktcnt; i++) {
1141                                printf("packet %d len = %d,"
1142                                        " actual_len = %d, status = 0x%x\en",
1142                                        " actual_len = %d, status = 0x%x\n",
1143                                        i, pktdesc->dsc_isoc_pkt_len,
1144                                        pktdesc->dsc_isoc_pkt_actual_len,
1145                                        pktdesc->dsc_isoc_pkt_status);

1147                                /* Processing data */

1149                                /*
1150                                * next packet data payload, do NOT use
1151                                * dsc_isoc_pkt_actual_len
1152                                */
1153                                p += pktdesc->dsc_isoc_pkt_len;

1155                                pktdesc++;
1156         }
1157 .fi
1158 .in -2

1160 .sp
1161 .LP
1162 For an isochronous-OUT endpoint, applications use the same packet descriptor
1163 and request structures to write request   information to the \fBugen\fR node.
1164 Following the packet request head information is the packet payload data. Upon
1165 successful return of \fBwrite\fR(2), applications can \fBread\fR(2) the same
1166 \fBugen\fR file immediately to retrieve the individual packet transfer status
1167 of the last request. If the application isn't concerned about the status, it
1168 can omit it.
1169 .sp
1170 .LP
1171 In the following example, an application transfers data on an isochronous-OUT
1172 endpoint:
1173 .sp
1174 .in +2
1175 .nf
1176                 #include <sys/usb/clients/ugen/usb_ugen.h>
1177                 char *buf, *p;
```



```
1178                 ushort_t i, pktlen;
1179                 int len, pktcnt;
1180                 ugen_isoc_req_head_t *req;
1181                 ugen_isoc_pkt_descr_t *pktdesc;
1182                 char rdbuf[4096];

1184                 pktcnt = 4;

1186                 /*
1187                 * set packet length to a proper value, don't
1188                 * exceed endpoint's capability
1189                 */
1190                 pktlen = 1024;

1192                 len = sizeof(int) +
1193                         sizeof(ugen_isoc_pkt_descr_t) * pktcount;
1194
1195                 len += pktlen * pktcnt;

1197                 buf = malloc(len);
1198                 if (!buf) {
1199                         /* Error recovery. */
1200             }

1202                 req = (ugen_isoc_req_head_t *)buf;
1203                 req->req_isoc_pkts_count = pktcnt;

1205                 pktdesc =
1206                         (ugen_isoc_pkt_descr_t *)(req->req_isoc_pkt_descrs);

1208                 for (i = 0; i < pktcnt; i++) {
1209                         pktdesc[i].dsc_isoc_pkt_len = pktlen;
1210                         pktdesc[i].dsc_isoc_pkt_actual_len = 0;
1211                         pktdesc[i].dsc_isoc_pkt_status = 0;
1212             }
1213
1214                 /* moving to beginning of payload data */
1215                 p = buf + sizeof(int) + sizeof(*pktdesc) * pktcnt;
1216                 for (i = 0; i < pktcnt; i++) {

1218                         /* fill in the data buffer */

1220                         p += pktlen;
1221      }

1223                 /*
1224                 * write packet request information and data to ugen driver
1225                 *
1226                 * len should be the exact value of sizeof(int) +
1227                 *    sizeof(ugen_isoc_pkt_descr_t) * pktcnt + payload length
1228                 */
1229                 if (write(fd, buf, len) < 0) {
1230                         /* Error recovery. */
1231             }

1233                 /* read packet status */
1234                 if (read(fd, rdbuf, sizeof(*pktdesc) * pktcnt) < 0) {

1236                         /* Error recovery. */

1238                 } else {

1240                         /* Parse every packet's transfer status */

1242             }
1243 .fi
```

```
.in -2

.SH ERRORS
.sp
.LP
The following statuses are returned by endpoint status device logical names:
.sp
.ne 2
.na
\fBUSB_LC_STAT_NOERROR\fR
.ad
.sp .6
.RS 4n
No error.
.RE

.sp
.ne 2
.na
\fBUSB_LC_STAT_CRC\fR
.ad
.sp .6
.RS 4n
CRC error detected.
.RE

.sp
.ne 2
.na
\fBUSB_LC_STAT_BITSTUFFING\fR
.ad
.sp .6
.RS 4n
Bit stuffing error.
.RE

.sp
.ne 2
.na
\fBUSB_LC_STAT_DATA_TOGGLE_MM\fR
.ad
.sp .6
.RS 4n
Data toggle did not match.
.RE

.sp
.ne 2
.na
\fBUSB_LC_STAT_STALL\fR
.ad
.sp .6
.RS 4n
Endpoint returned stall.
.RE

.sp
.ne 2
.na
\fBUSB_LC_STAT_DEV_NOT_RESP\fR
.ad
.sp .6
.RS 4n
Device not responding.
.RE
```



```
.sp
.ne 2
.na
\fBUSB_LC_STAT_UNEXP_PID\fR
.ad
.sp .6
.RS 4n
Unexpected Packet Identifier (PID).
.RE

.sp
.ne 2
.na
\fBUSB_LC_STAT_PID_CHECKFAILURE\fR
.ad
.sp .6
.RS 4n
Check bits on PID failed.
.RE

.sp
.ne 2
.na
\fBUSB_LC_STAT_DATA_OVERRUN\fR
.ad
.sp .6
.RS 4n
Data overrun.
.RE

.sp
.ne 2
.na
\fBUSB_LC_STAT_DATA_UNDERRUN\fR
.ad
.sp .6
.RS 4n
Data underrun.
.RE

.sp
.ne 2
.na
\fBUSB_LC_STAT_BUFFER_OVERRUN\fR
.ad
.sp .6
.RS 4n
Buffer overrun.
.RE

.sp
.ne 2
.na
\fBUSB_LC_STAT_BUFFER_UNDERRUN\fR
.ad
.sp .6
.RS 4n
Buffer underrun.
.RE

.sp
.ne 2
.na
\fBUSB_LC_STAT_TIMEOUT\fR
.ad
.sp .6
```

```
.RS 4n
Command timed out.
.RE

.sp
.ne 2
.na
\fBUSB_LC_STAT_NOT_ACCESSED\fR
.ad
.sp .6
.RS 4n
Not accessed by the hardware.
.RE

.sp
.ne 2
.na
\fBUSB_LC_STAT_UNSPECIFIED_ERR\fR
.ad
.sp .6
.RS 4n
Unspecified USBA or HCD error.
.RE

.sp
.ne 2
.na
\fBUSB_LC_STAT_NO_BANDWIDTH\fR
.ad
.sp .6
.RS 4n
No bandwidth available.
.RE

.sp
.ne 2
.na
\fBUSB_LC_STAT_HW_ERR\fR
.ad
.sp .6
.RS 4n
Host Controller h/w error.
.RE

.sp
.ne 2
.na
\fBUSB_LC_STAT_SUSPENDED\fR
.ad
.sp .6
.RS 4n
Device was suspended.
.RE

.sp
.ne 2
.na
\fBUSB_LC_STAT_DISCONNECTED\fR
.ad
.sp .6
.RS 4n
Device was disconnected.
.RE

.sp
.ne 2
```



```
.na
\fBUSB_LC_STAT_INTR_BUF_FULL\fR
.ad
.sp .6
.RS 4n
Polling was stopped as the interrupt-IN data buffer was full. Buffer is  now
empty and polling has been resumed.
.RE

.sp
.ne 2
.na
\fBUSB_LC_STAT_INTERRUPTED\fR
.ad
.sp .6
.RS 4n
Request was interrupted.
.RE

.sp
.ne 2
.na
\fBUSB_LC_STAT_NO_RESOURCES\fR
.ad
.sp .6
.RS 4n
No resources available for request.
.RE

.sp
.ne 2
.na
\fBUSB_LC_STAT_INTR_POLLING_FAILED\fR
.ad
.sp .6
.RS 4n
Failed to restart polling.
.RE

.sp
.ne 2
.na
\fBUSB_LC_STAT_ISOC_POLLING_FAILED\fR
.ad
.sp .6
.RS 4n
Failed to start isochronous polling.
.RE

.sp
.ne 2
.na
\fBUSB_LC_STAT_ISOC_UNINITIALIZED\fR
.ad
.sp .6
.RS 4n
Isochronous packet information not initialized.
.RE

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

All packets in this isochronous request have errors. The polling on this isochronous-IN endpoint is suspended and can be resumed on next \fBread\fR(2).
.RE

.sp
.LP
The following system call \fBerrno\fR values are returned:
.sp
.ne 2
.na
\fB\fBEINVAL\fR\fR
.ad
.RS 11n
An attempt was made to enable or disable "one transfer" mode while the associated endpoint was open.
.RE

.sp
.ne 2
.na
\fB\fBEBUSY\fR\fR
.ad
.RS 11n
The endpoint has been opened and another open is attempted.
.RE

.sp
.ne 2
.na
\fB\fBEACCES\fR\fR
.ad
.RS 11n
An endpoint open was attempted with incorrect flags.
.RE

.sp
.ne 2
.na
\fB\fBENOTSUP\fR\fR
.ad
.RS 11n
Operation not supported.
.RE

.sp
.ne 2
.na
\fB\fBENXIO\fR\fR
.ad
.RS 11n
Device associated with the file descriptor does not exist.
.RE

.sp
.ne 2
.na
\fBENODEV\fR
.ad
.RS 11n
Device has been hot-removed or a suspend/resume happened before this command.
.RE

.sp
.ne 2
.na
\fBEIO\fR

.ad
.RS 11n
An I/O error occurred. Send a read on the endpoint status minor node to get the exact error information.
.RE

.sp
.ne 2
.na
\fBEINTR\fR
.ad
.RS 11n
Interrupted system call.
.RE

.sp
.ne 2
.na
\fBENOMEM\fR
.ad
.RS 11n
No memory for the allocation of internal structures.
.RE

.SH FILES
.sp
.in +2
.nf
        /kernel/drv/ugen   32 bit ELF kernel module (x86 platform only)
        /kernel/drv/sparcv9/ugen   64 bit ELF kernel module

        /dev/usb/<vid>.<pid>/<N>/cntrl0
        /dev/usb/<vid>.<pid>/<N>/cntrl0stat

        /dev/usb/<vid>.<pid>/<N>/if<interface#>
                                <in|out|cntrl><endpoint#>
        /dev/usb/<vid>.<pid>/<N>/if<interface#>
                                <in|out|cntrl><endpoint#>stat

        /dev/usb/<vid>.<pid>/<N>/if<interface#>.
                                <alternate><in|out|cntrl<endpoint#>
        /dev/usb/<vid>.<pid>/<N>/if<interface#>.
                                <alternate><in|out|cntrl><endpoint#>stat

        /dev/usb/<vid>.<pid>/<N>/cfg<value>if<interface#>
                                <in|out|cntrl><endpoint#>
        /dev/usb/<vid>.<pid>/<N>/cfg<value>if<interface#>
                                <in|out|cntrl<endpoint#stat>

        /dev/usb/<vid>.<pid>/<N>/cfg<value>if<interface#>.
                                <alternate><in|out|cntrl><endpoint#>
        /dev/usb/<vid>.<pid>/<N>/cfg<value>if<interface#>.
                                <alternate><in|out|cntrl><endpoint#>stat

        /dev/usb/<vid>.<pid>/<N>/devstat

        /dev/usb/<vid>.<pid>/<N>/if<interface#>cntrl0
        /dev/usb/<vid>.<pid>/<N>/if<interface#>cntrl0stat
.fi
.in -2

.sp
.LP
where \fIN\fR is an integer representing the instance number of this type of device. (All logical device names for a single device share the same \fIN\fR.)

```
.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
_
Architecture    PCI-based  SPARC
.TE

.SH SEE ALSO
.sp
.LP
\fBlibusb\fR(3LIB), \fBclose\fR(2), \fBpoll\fR(2), \fBread\fR(2),
\fBwrite\fR(2), \fBaioread\fR(3C), \fBaiowrite\fR(3C), \fBusba\fR(7D),
\fBusb_dev_descr\fR(9S).
.SH DIAGNOSTICS
.sp
.LP
In addition to being logged, the following messages may appear on the system
console. All messages are formatted in the following manner:
.sp
.in +2
.nf
Warning: <device path> (ugen<instance num>): Error Message...
.fi
.in -2
.sp

.sp
.ne 2
.na
\fBToo many minor nodes. \fR
.ad
.sp .6
.RS 4n
Device has too many minor nodes. Not all are available.
.RE

.sp
.ne 2
.na
\fBInstance number too high (<\fInumber\fR>).\fR
.ad
.sp .6
.RS 4n
Too many devices are using this driver.
.RE

.sp
.ne 2
.na
\fBCannot access <device>.  Please reconnect.\fR
.ad
.sp .6
.RS 4n
This device has been disconnected because a device other than the original one
has been inserted. The driver informs you of this fact by displaying the name
of the original device.
.RE
```



```
.sp
.ne 2
.na
\fBDevice is not identical to the previous one on this port. Please disconnect
and reconnect.\fR
.ad
.sp .6
.RS 4n
Same condition as described above; however in this case, the driver is unable
to identify the original device with a name string.
.RE

.SH NOTES
.sp
.LP
\fBugen\fR returns \fB-1\fR for all commands and sets \fBerrno\fR to
\fBENODEV\fR when device has been hot-removed or resumed from a suspend. The
application must close and reopen all open minor nodes to reinstate successful
communication.
```

```
**********************************************************
   21024 Tue Sep 10 18:35:22 2013
new/usr/src/man/man7i/agpgart_io.7i
4023 - Typo in file(1) manpage and various others
**********************************************************
   1 '\" te
   2 .\" Copyright (c) 2008, Sun Microsystems, Inc.  All Rights Reserved
   3 .\" The contents of this file are subject to the terms of the Common Development
   4 .\" You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE or http:
   5 .\" When distributing Covered Code, include this CDDL HEADER in each file and in
   6 .TH AGPGART_IO 7I "Sep 10, 2013"
   6 .TH AGPGART_IO 7I "Sep 25, 2008"
   7 .SH NAME
   8 agpgart_io \- Solaris agpgart driver I/O control operations
   9 .SH SYNOPSIS
  10 .LP
  11 .nf
  12 #include <sys/agpgart.h>
  13 .fi

  15 .SH DESCRIPTION
  16 .sp
  17 .LP
  18 The Accelerated Graphics Port (AGP) is a PCI bus technology enhancement that
  19 improves 3D graphics performance by using low-cost system memory. AGP chipsets
  20 use the Graphics Address Remapping Table (GART) to map discontiguous system
  21 memory into a contiguous PCI memory range (known as the AGP Aperture), enabling
  22 the graphics card to utilize the mapped aperture range as video memory.
  23 .sp
  24 .LP
  25 The \fBagpgart\fR driver creates a pseudo device node at \fB/dev/agpgart\fR and
  26 provides a set of ioctls for managing  allocation/deallocation of system
  27 memory, setting mappings between system memory and aperture range, and setting
  28 up AGP devices. The \fBagpgart\fR driver manages both pseudo and real device
  29 nodes, but to initiate AGP-related operations you operate only on the
  30 \fB/dev/agpgart\fR pseudo device node. To do this, open \fB/dev/agpgart\fR. The
  31 macro defined for the pseudo device node name is:
  32 .sp
  33 .in +2
  34 .nf
  35 #define  AGP_DEVICE      "/dev/agpgart"
  36 .fi
  37 .in -2

  39 .sp
  40 .LP
  41 The \fBagpgart_io\fR driver implementation is AGP architecture-dependent and
  42 cannot be made generic. Currently, the \fBagpgart_io\fR driver only supports
  43 specific AGP  systems. To determine if a system is supported, run an
  44 \fBopen\fR(2) system  call on the AGP_DEVICE node. (Note that  \fBopen\fR(2)
  45 fails if a system is not supported). After the AGP_DEVICE is opened, you can
  46 use \fBkstat\fR(1M) to  read the system architecture  type.
  47 .sp
  48 .LP
  49 In addition to AGP system support, the \fBagpgart\fR ioctls can also be used on
  50 Intel integrated graphics devices (IGD). IGD devices usually have no dedicated
  51 video memory and must  use system memory as video memory. IGD devices contain
  52 translation tables (referred to as \fBGTT\fR tables) that are similar to the
  53 GART translation table for address mapping purposes.
  54 .sp
  55 .LP
  56 Processes must open the \fBagpgart_io\fR driver utilizing a GRAPHICS_ACCESS
  57 privilege. Then all the ioctls  can be called  by this processes with the saved
  58 file descriptor. With the exception of AGPIOC_INFO, the AGPIOC_ACQUIRE ioctl
  59 must be called before any other ioctl. Once a process has acquired GART, it
  60 cannot be acquired by another process until the former process calls
```



```
  61 AGPIOC_RELEASE.
  62 .sp
  63 .LP
  64 If the AGP_DEVICE fails to open, it may be due to one of the following reasons:
  65 .sp
  66 .ne 2
  67 .na
  68 \fBEAGAIN\fR
  69 .ad
  70 .sp .6
  71 .RS 4n
  72 GART table allocation failed.
  73 .RE

  75 .sp
  76 .ne 2
  77 .na
  78 \fBEIO\fR
  79 .ad
  80 .sp .6
  81 .RS 4n
  82 Internal hardware initialization failed.
  83 .RE

  85 .sp
  86 .ne 2
  87 .na
  88 \fBENXIO\fR
  89 .ad
  90 .sp .6
  91 .RS 4n
  92 Getting device soft state error. (This is unlikely to happen.)
  93 .RE

  95 .sp
  96 .ne 2
  97 .na
  98 \fB EPERM\fR
  99 .ad
 100 .sp .6
 101 .RS 4n
 102 Without enough privilege.
 103 .RE

 105 .SH IOCTLS
 106 .sp
 107 .LP
 108 With the exception of GPIOC_INFO, all ioctls shown in this section are
 109 protected by GRAPHICS_ACCESS privilege. (Only processes with GRAPHICS_ACCESS
 110 privilege in its effective set can access the privileged ioctls).
 111 .sp
 112 .LP
 113 Common ioctl error codes are shown below. (Additional error codes may be
 114 displayed by individual ioctls.)
 115 .sp
 116 .ne 2
 117 .na
 118 \fBENXIO\fR
 119 .ad
 120 .sp .6
 121 .RS 4n
 122 Ioctl command not supported or getting device soft state error.
 123 .RE

 125 .sp
 126 .ne 2
```

```
.na
\fBEPERM\fR
.ad
.sp .6
.RS 4n
Process not privileged.
.RE

.sp
.ne 2
.na
\fB\fBAGPIOC_INFO\fR\fR
.ad
.sp .6
.RS 4n
Get system wide AGP or IGD hardware information. This command can be called by
any process from user or kernel context.
.sp
.in +2
.nf
The argument is a pointer to agp_info_t structure.

 typedef struct  _agp_info {
       agp_version_t agpi_version; /* OUT: AGP version supported */
       uint32_t agpi_devid;    /* OUT: bridge vendor + device */
       uint32_t agpi_mode;     /* OUT: mode of bridge */
       ulong_t  agpi_aperbase; /* OUT: base of aperture */
       size_t   agpi_apersize; /* OUT: aperture size in MB */
       uint32_t agpi_pgtotal;  /* OUT: max aperture pages avail. */
       uint32_t agpi_pgsystem; /* OUT: same as pg_total */
       uint32_t agpi_pgused; /* OUT: no. of currently used pages */
 } agp_info_t;

agpi_version  The version of AGP protocol the bridge device is
              compatible with, for example, major 3 and minor 0
              means AGP version 3.0.

              typedef struct _agp_version {
                     uint16_t   agpv_major;
                     uint16_t   agpv_minor;
              } agp_version_t;

agpi_devid    AGP bridge vendor and device ID.
agpi_mode     Current AGP mode, read from AGP status register of
              target device. The main bits are defined as below.
              /* AGP status register bits definition */

                #define AGPSTAT_RQ_MASK         0xff000000
                #define AGPSTAT_SBA             (0x1 << 9)
                #define AGPSTAT_OVER4G          (0x1 << 5)
                #define AGPSTAT_FW              (0x1 << 4)
                #define AGPSTAT_RATE_MASK       0x7
                /* AGP 3.0 only bits */
                #define AGPSTAT_ARQSZ_MASK      (0x7 << 13)
                #define AGPSTAT_CAL_MASK        (0x7 << 10)
                #define AGPSTAT_GART64B         (0x1 << 7)
                #define AGPSTAT_MODE3           (0x1 << 3)
                /* rate for 2.0 mode */
                #define AGP2_RATE_1X                    0x1
                #define AGP2_RATE_2X                    0x2
                #define AGP2_RATE_4X                    0x4
                /* rate for 3.0 mode */
                #define AGP3_RATE_4X                    0x1
                #define AGP3_RATE_8X                    0x2

agpi_aperbase   The base address of aperture in PCI memory space.
```



```
agpi_apersize   The size of the aperture in megabytes.
agpi_pgtotal    Represents the maximum memory
                pages the system can allocate
                according to aperture size and
                system memory size (which may differ
                from the maximum locked memory a process
                can have. The latter  is subject
                to the memory resource limit imposed
                by the  resource_controls(5) for each
                project(4)):

                  project.max-device-locked-memory

                This value can be modified through system
                utilities like prctl(1).

agpi_pgsystem  Same as pg_total.
agpi_pgused    System pages already allocated by the driver.

Return Values:

            EFAULT  Argument copy out error
            EINVAL  Command invalid
            0       Success
.fi
.in -2

.RE

.sp
.ne 2
.na
\fB\fBAGPIOC_ACQUIRE\fR\fR
.ad
.sp .6
.RS 4n
Acquire control of GART. With the exception of AGPIOC_INFO, a process must
acquire GART before can it call other agpgart ioctl commands. Additionally,
only processes with GRAPHICS_ACCESS privilege may access this ioctl. In the
current agpgart implementation, GART access is exclusive, meaning that only one
process can perform GART operations at a time. To release control over GART,
call AGPIOC_RELEASE. This command can be called from user or kernel context.
.sp
The argument should be NULL.
.sp
Return values:
.sp
.ne 2
.na
\fBEBUSY\fR
.ad
.RS 9n
GART has been acquired
.RE

.sp
.ne 2
.na
\fB0\fR
.ad
.RS 9n
Success.
.RE

.RE
```

```
.sp
.ne 2
.na
\fB\fBAGPIOC_RELEASE\fR\fR
.ad
.sp .6
.RS 4n
Release GART control. If a process releases GART control, it cannot perform
additional GART operations until GART is reacquired. Note that this command
does not free allocated memory or clear GART entries. (All clear jobs are done
by direct calls or by closing the device). When a process exits without making
this ioctl, the final \fBclose\fR(2) performs this automatically. This command
can be called from user or kernel context.
.sp
The argument should be NULL.
.sp
Return values:
.sp
.ne 2
.na
\fBEPERM\fR
.ad
.RS 9n
Not owner of GART.
.RE

.sp
.ne 2
.na
\fB0\fR
.ad
.RS 9n
Success.
.RE

.RE

.sp
.ne 2
.na
\fB\fBAGPIOC_SETUP\fR\fR
.ad
.sp .6
.RS 4n
Setup AGPCMD register. An AGPCMD register resides in both  the  AGP master and
target devices. The AGPCMD register controls the working mode of the AGP master
and target devices. Each device must be  configured  using the same mode. This
command can be called from user or kernel context.
.sp
.in +2
.nf
The argument is a pointer to agp_setup_t structure:

     typedef struct _agp_setup {
          uint32_t agps_mode; /* IN: value to be set for AGPCMD */
     } agp_setup_t;

agps_mode  Specifying the mode to be set. Each bit of the value may have
           a specific meaning, please refer to AGP 2.0/3.0 specification
           or hardware datasheets for details.

               /* AGP command register bits definition */
               #define     AGPCMD_RQ_MASK          0xff000000
               #define     AGPCMD_SBAEN            (0x1 << 9)
               #define     AGPCMD_AGPEN            (0x1 << 8)
               #define     AGPCMD_OVER4GEN         (0x1 << 5)
```



```
               #define     AGPCMD_FWEN             (0x1 << 4)
               #define     AGPCMD_RATE_MASK        0x7
               /* AGP 3.0 only bits */
               #define     AGP3_CMD_ARQSZ_MASK     (0x7 << 13)
               #define     AGP3_CMD_CAL_MASK       (0x7 << 10)
               #define     AGP3_CMD_GART64BEN      (0x1 << 7)
.fi
.in -2

The final values set to the AGPCMD register of the master/target devices are
decided by the agps_mode value and AGPSTAT of the master and target devices.
.sp
Return Values:
.sp
.ne 2
.na
\fBEPERM\fR
.ad
.RS 10n
Not owner of GART.
.RE

.sp
.ne 2
.na
\fBEFAULT\fR
.ad
.RS 10n
Argument copy in error.
.RE

.sp
.ne 2
.na
\fBEINVAL\fR
.ad
.RS 10n
Command invalid for non-AGP system.
.RE

.sp
.ne 2
.na
\fBEIO\fR
.ad
.RS 10n
Hardware setup error.
.RE

.sp
.ne 2
.na
\fB0\fR
.ad
.RS 10n
Success.
.RE

.RE

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

```
391 .RS 4n
392 Allocate system memory for graphics device. This command returns a unique ID
393 which can be used in subsequent operations to represent the allocated memory.
394 The memory is made up of discontiguous physical pages. In rare cases, special
395 memory types may be required. The allocated memory must be bound to the GART
396 table before it can be used by graphics device. Graphics applications can also
397 \fBmmap\fR(2) the memory to userland for data storing. Memory should be freed
398 when it is no longer used by calling AGPIOC_DEALLOCATE or simply by closing the
399 device. This command can be called from user or kernel context.
400 .sp
401 .in +2
402 .nf
403 The argument is a pointer to agp_allocate_t structure.

405      typedef struct _agp_allocate {
406            int32_t  agpa_key;     /* OUT:ID of allocated memory */
407            uint32_t agpa_pgcount;/* IN: no. of pages to be allocated */
408            uint32_t agpa_type;/* IN: type of memory to be allocated */
409            uint32_t agpa_physical; /* OUT: reserved */
410      } agp_allocate_t;
411 .fi
412 .in -2

414 .sp
415 .ne 2
416 .na
417 \fBagpa_key\fR
418 .ad
419 .RS 21n
420 Unique ID of the allocated memory.
421 .RE

423 .sp
424 .ne 2
425 .na
426 \fBagpa_pgcount\fR
427 .ad
428 .RS 21n
429 Number of pages to be allocated. The driver currently supports only 4K pages.
430 The value cannot exceed the agpi_pgtotal value returned by AGPIOC_INFO ioct and
431 is subject to the limit of project.max-device-locked-memory. If the memory
432 needed is larger than the resource limit but not larger than agpi_pgtotal, use
433 \fBprctl\fR(1) or other system utilities to change the default value of memory
434 resource limit beforehand.
435 .RE

437 .sp
438 .ne 2
439 .na
440 \fBagpa_type\fR
441 .ad
442 .RS 21n
443 Type of memory to be allocated. The valid value of agpa_type should be
444 AGP_NORMAL. It is defined as:
445 .sp
446 .in +2
447 .nf
448          #define AGP_NORMAL      0
449 .fi
450 .in -2

452 Above, AGP_NORMAL represents the discontiguous non-cachable physical memory
453 which doesn't consume kernel virtual space but can be mapped to user space by
454 \fBmmap\fR(2). This command may support more type values in the future.
455 .RE
```



```
457 .sp
458 .ne 2
459 .na
460 \fBagpa_physical\fR
461 .ad
462 .RS 21n
463 Reserved for special uses. In normal operations, the value is undefined.
464 .sp
465 Return Values:
466 .sp
467 .ne 2
468 .na
469 \fBEPERM\fR
470 .ad
471 .RS 10n
472 Not owner of GART.
473 .RE

475 .sp
476 .ne 2
477 .na
478 \fBEINVAL\fR
479 .ad
480 .RS 10n
481 Argument not valid.
482 .RE

484 .sp
485 .ne 2
486 .na
487 \fBEFAULT\fR
488 .ad
489 .RS 10n
490 Argument copy in/out error.
491 .RE

493 .sp
494 .ne 2
495 .na
496 \fBENOMEM\fR
497 .ad
498 .RS 10n
499 Memory allocation error.
500 .RE

502 .sp
503 .ne 2
504 .na
505 \fB0\fR
506 .ad
507 .RS 10n
508 Success.
509 .RE

511 .RE

513 .sp
514 .ne 2
515 .na
516 \fBAGPIOC_DEALLOCATE\fR
517 .ad
518 .RS 21n
519 Deallocate the memory identified by a key assigned in a previous allocation. If
520 the memory isn't unbound from GART, this command unbinds it automatically. The
521 memory should no longer be used and those still in mapping to userland cannot
522 be deallocated. Always call AGPIOC_DEALLOCATE explicitly (instead of
```

deallocating implicitly by closing the device), as the system won’t carry out the job until the last reference to the device file is dropped. This command from user or kernel context.

.sp

The input argument is a key of type int32_t, no output argument.

.sp

Return Values:

.sp

.ne 2

.na

\fBEPERM\fR

.ad

.RS 10n

Not owner of GART.

.RE

.sp

.ne 2

.na

\fBEINVAL\fR

.ad

.RS 10n

Key not valid or memory in use.

.RE

.sp

.ne 2

.na

\fB0\fR

.ad

.RS 10n

Success.

.RE

.RE

.sp

.ne 2

.na

\fBAGPIOC_BIND\fR

.ad

.RS 21n

Bind allocated memory. This command binds the allocated   memory identified by a key to a specific offset of the GART table, which enables GART   to translate the aperture range at the offset to system memory. Each GART entry represents one physical page. If the GART range is previously bound to other system memory, it returns an error. Once the memory is bound, it cannot be bound to other offsets unless it is unbound. To unbind the memory, call AGPIOC_UNBIND or deallocate the memory. This command can be called from user or kernel context.

.sp

.in +2

.nf

The argument is a pointer to agp_bind_t structure:

```
     typedef struct _agp_bind {
          int32_t  agpb_key;      /* IN: ID of memory to be bound */
          uint32_t agpb_pgstart;  /* IN: offset in aperture */
     } agp_bind_t;
```

.fi

.in -2

.sp

.ne 2

.na

\fBagpb_key\fR

.ad

.RS 20n

The unique ID of the memory to be bound, which is previously allocated by calling AGPIOC_ALLOCATE.

.RE

.sp

.ne 2

.na

\fBagpb_pgstart\fR

.ad

.RS 20n

The starting page offset to be bound in aperture space.

.RE

Return Values:

.sp

.ne 2

.na

\fBEPERM\fR

.ad

.RS 20n

Not owner of GART.

.RE

.sp

.ne 2

.na

\fBEFAULT\fR

.ad

.RS 20n

Argument copy in error.

.RE

.sp

.ne 2

.na

\fBEINVAL\fR

.ad

.RS 20n

Argument not valid.

.RE

.sp

.ne 2

.na

\fBEIO\fR

.ad

.RS 20n

Binding to the GTT table of IGD devices failed.

.RE

.sp

.ne 2

.na

\fB0\fR

.ad

.RS 20n

Success.

.RE

.RE

.sp

.ne 2

.na

```
655 \fBAGPIOC_UNBIND\fR
656 .ad
657 .RS 21n
658 Unbind memory identified by a key from the GART. This command clears the
659 corresponding entries in the GART table. Only the memory not in mapping to
660 userland is allowed to be unbound.
661 .sp
662 This ioctl command can be called from user or kernel context.
663 .sp
664 .in +2
665 .nf
666 The argument is a pointer to agp_unbind_t structure.

668      typedef struct _agp_unbind {
669           int32_t  agpu_key; /* IN: key of memory to be unbound*/
670           uint32_t agpu_pri; /* Not used: for compat. with Xorg */
671      } agp_unbind_t;
672 .fi
673 .in -2

675 .sp
676 .ne 2
677 .na
678 \fBagpu_key\fR
679 .ad
680 .RS 20n
681 Unique ID of the memory to be unbound which was previously bound by calling
682 AGPIOC_BIND.
683 .RE

685 .sp
686 .ne 2
687 .na
688 \fBagpu_pri\fR
689 .ad
690 .RS 20n
691 Reserved for compatibility with X.org/XFree86, not used.
692 .RE

694 Return Values:
695 .sp
696 .ne 2
697 .na
698 \fBEPERM\fR
699 .ad
700 .RS 20n
701 Not owner of GART.
702 .RE

704 .sp
705 .ne 2
706 .na
707 \fBEFAULT\fR
708 .ad
709 .RS 20n
710 Argument copy in error.
711 .RE

713 .sp
714 .ne 2
715 .na
716 \fBEINVAL\fR
717 .ad
718 .RS 20n
719 Argument not valid or memory in use.
720 .RE
```



```
722 .sp
723 .ne 2
724 .na
725 \fBEIO\fR
726 .ad
727 .RS 20n
728 Unbinding from the GTT table of IGD devices failed.
729 .RE

731 .sp
732 .ne 2
733 .na
734 \fB0\fR
735 .ad
736 .RS 20n
737 Success
738 .RE

740 .RE

742 .RE

744 .SH EXAMPLE
745 .sp
746 .LP
747 Below is an sample program showing how agpgart ioctls can be used:
748 .sp
749 .in +2
750 .nf
751 #include <stdio.h>
752 #include <stdlib.h>
753 #include <unistd.h
754 #include <sys/ioccom.h>
755 #include <sys/types.h>
756 #include <fcntl.h>
757 #include <errno.h>
758 #include <sys/mman.h>
759 #include <sys/agpgart.h>

761 #define AGP_PAGE_SIZE   4096

763 int main(int argc, char *argv[])
764 {
765         int fd, ret;
766         agp_allocate_t alloc;
767         agp_bind_t bindinfo;
768         agp_info_t agpinfo;
769         agp_setup_t modesetup;
770         int *p = NULL;
771         off_t mapoff;
772         size_t maplen;

774         if((fd = open(AGP_DEVICE, O_RDWR))== -1) {
775                 printf("open AGP_DEVICE error with %d\en", errno);\e
776                 exit(-1);
777         }
778         printf("device opened\en");

780         ret = ioctl(fd, AGPIOC_INFO, &agpinfo);
781         if(ret == -1) {
782                 printf("Get info error %d\en", errno);
782                 printf("Get info error %d\n", errno);
783                 exit(-1);
784         }
785         printf("AGPSTAT is %x\en", agpinfo.agpi_mode);
```

```
786         printf("APBASE is %x\en", agpinfo.agpi_aperbase);
787         printf("APSIZE is %dMB\en", agpinfo.agpi_apersize);
788         printf("pg_total is %d\en", agpinfo.agpi_pgtotal);

790         ret = ioctl(fd, AGPIOC_ACQUIRE);
791         if(ret == -1) {
792                 printf(" Acquire GART error %d\en", errno);
793                 exit(-1);
794         }

796         modesetup.agps_mode = agpinfo.agpi_mode;
797         ret = ioctl(fd, AGPIOC_SETUP, &modesetup);
798         if(ret == -1) {
799                 printf("set up AGP mode error\en", errno);
800                 exit(-1);
801         }

803         printf("Please input the number of pages you want to allocate\en");
804         scanf("%d", &alloc.agpa_pgcount);
805         alloc.agpa_type = AGP_NORMAL;
806         ret = ioctl(fd, AGPIOC_ALLOCATE, &alloc);
807         if(ret == -1) {
808                 printf("Allocate memory error %d\en", errno);
809                 exit(-1);
810         }

812         printf("Please input the aperture page offset to bind\en");
813         scanf("%d", &bindinfo.agpb_pgstart);
814         bindinfo.agpb_key = alloc.agpa_key;
815         ret = ioctl(fd, AGPIOC_BIND, &bindinfo);
816         if(ret == -1) {
817                 printf("Bind error %d\en", errno);
818                 exit(-1);
819         }
820         printf("Bind successful\en");

822         /*
823          * Now gart aperture space from (bindinfo.agpb_pgstart) to
824          * (bindinfo.agpb_pgstart + alloc.agpa_pgcount) can be used for
825          * AGP graphics transactions
826          */
827         ...

829         /*
830          * mmap can allow user processes to store graphics data
831          * to the aperture space
832          */
833         maplen = alloc.agpa_pgcount * AGP_PAGE_SIZE;
834         mapoff = bindinfo.agpb_pgstart * AGP_PAGE_SIZE;
835         p = (int *)mmap((caddr_t)0, maplen, (PROT_READ | PROT_WRITE),
836                         MAP_SHARED, fd, mapoff);
837         if (p == MAP_FAILED) {
838                 printf("Mmap error %d\en", errno);
839                 exit(-1);
840         }
841         printf("Mmap successful\en");
842         ...

844         /*
845          * When user processes finish access to the aperture space,
846          * unmap the memory range
847          */
848         munmap((void *)p, maplen);
849         ...

851         /*
```



```
852          * After finishing AGP transactions, the resources can be freed
853          * step by step or simply by close device.
854          */
855         ret = ioctl(fd, AGPIOC_DEALLOCATE, alloc.agpa_key);
856         if(ret == -1) {
857                 printf(" Deallocate memory error %d\en", errno);
858                 exit(-1);
859         }

861         ret = ioctl(fd, AGPIOC_RELEASE);
862         if(ret == -1) {
863                 printf(" Release GART error %d\en", errno);
864                 exit(-1);
865         }

867         close(fd);
868 }
```

_____***unchanged_portion_omitted_***