

new/usr/src/cmd/grep/grep.c

1

```
*****
10801 Sat Sep 14 23:25:09 2013
new/usr/src/cmd/grep/grep.c
3546 add support for grep -o option
*****
1 /*
2  * CDDL HEADER START
3  *
4  * The contents of this file are subject to the terms of the
5  * Common Development and Distribution License, Version 1.0 only
6  * (the "License"). You may not use this file except in compliance
7  * with the License.
8  *
9  * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
10 * or http://www.opensolaris.org/os/licensing.
11 * See the License for the specific language governing permissions
12 * and limitations under the License.
13 *
14 * When distributing Covered Code, include this CDDL HEADER in each
15 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
16 * If applicable, add the following below this CDDL HEADER, with the
17 * fields enclosed by brackets "[" replaced with your own identifying
18 * information: Portions Copyright [yyyy] [name of copyright owner]
19 *
20 * CDDL HEADER END
21 */
22 /*
23 * Copyright 2005 Sun Microsystems, Inc. All rights reserved.
24 * Use is subject to license terms.
25 */
27 /*      Copyright (c) 1984, 1986, 1987, 1988, 1989 AT&T */
28 /*      All Rights Reserved */
30 /*      Copyright (c) 1987, 1988 Microsoft Corporation */
31 /*      All Rights Reserved */
33 /* Copyright 2012 Nexenta Systems, Inc. All rights reserved. */
35 /*
36 * Copyright 2013 Damian Bogel. All rights reserved.
37 * Copyright (c) 2013 Andrew Stormont. All rights reserved.
38 #endif /* !codereview */
39 */
41 /*
42 * grep -- print lines matching (or not matching) a pattern
43 *
44 *      status returns:
45 *          0 - ok, and some matches
46 *          1 - ok, but no matches
47 *          2 - some error
48 */
50 #include <sys/types.h>
52 #include <ctype.h>
53 #include <fcntl.h>
54 #include <locale.h>
55 #include <memory.h>
56 #include <regex.h>
57 #include <stdio.h>
58 #include <stdlib.h>
59 #include <string.h>
60 #include <unistd.h>
61 #include <ftw.h>
```

new/usr/src/cmd/grep/grep.c

2

```
62 #include <limits.h>
63 #include <sys/param.h>
65 static const char *errstr[] = {
66     "Range endpoint too large.",
67     "Bad number.",
68     "'\\digit' out of range.",
69     "No remembered search string.",
70     "\\( \\) imbalance.",
71     "Too many \\(.",
72     "More than 2 numbers given in \\{ \\}.",
73     "} expected after \\(.",
74     "First number exceeds second in \\{ \\}.",
75     "[ ] imbalance.",
76     "Regular expression overflow.",
77     "Illegal byte sequence.",
78     "Unknown regexp error code!!",
79     NULL
80 };
82 #define STDIN_FILENAME  gettext("(standard input)")
84 #define errmsg(msg, arg)      (void) fprintf(stderr, gettext(msg), arg)
85 #define BLKSIZE 512
86 #define GBUFSIZ 8192
87 #define MAX_DEPTH 1000
89 static int      temp;
90 static long long lnum;
91 static char     *linebuf;
92 static char     *prntbuf = NULL;
93 static long     fw_lPrntBufLen = 0;
94 static int      nflag;
95 static int      bflag;
96 static int      lflag;
97 static int      cflag;
98 static int      rflag;
99 static int      Rflag;
100 static int      vflag;
101 static int      sflag;
102 static int      iflag;
103 static int      wflag;
104 static int      hflag;
105 static int      Hflag;
106 static int      qflag;
107 static int      oflag;
108 #endif /* !codereview */
109 static int      errflg;
110 static int      nfile;
111 static long long tln;
112 static int      nsucc;
113 static int      outfn = 0;
114 static int      nlflag;
115 static char     *ptr, *ptrend;
116 static char     *expbuf;
118 static void     execute(const char *, int);
119 static void     regerr(int);
120 static void     prepare(const char *);
121 static int      recursive(const char *, const struct stat *, int, struct FTW *);
122 static int      succeed(const char *);
124 int
125 main(int argc, char **argv)
126 {
127     int      c;
```

```

128     char *arg;
129     extern int optind;

131     (void) setlocale(LC_ALL, "");
132 #if !defined(TEXT_DOMAIN) /* Should be defined by cc -D */
133 #define TEXT_DOMAIN "SYS_TEST" /* Use this only if it weren't */
134 #endif
135     (void) textdomain(TEXT_DOMAIN);

137     while ((c = getopt(argc, argv, "hHqblcnoRrsviyw")) != -1)
138     while ((c = getopt(argc, argv, "hHqblcnoRrsviyw")) != -1)
139     switch (c) {
140     /* based on options order h or H is set as in GNU grep */
141     case 'h':
142         hflag++;
143         Hflag = 0; /* h excludes H */
144         break;
145     case 'H':
146         if (!lflag) /* H is excluded by l */
147             Hflag++;
148         hflag = 0; /* H excludes h */
149         break;
150     case 'q': /* POSIX: quiet: status only */
151         qflag++;
152         break;
153     case 'v':
154         vflag++;
155         break;
156     case 'c':
157         cflag++;
158         break;
159     case 'n':
160         nflag++;
161         break;
162     case 'o':
163         oflag++;
164         break;
165 #endif /* ! codereview */
166     case 'R':
167         Rflag++;
168         /* FALLTHROUGH */
169     case 'r':
170         rflag++;
171         break;
172     case 'b':
173         bflag++;
174         break;
175     case 's':
176         sflag++;
177         break;
178     case 'l':
179         lflag++;
180         Hflag = 0; /* l excludes H */
181         break;
182     case 'y':
183     case 'i':
184         iflag++;
185         break;
186     case 'w':
187         wflag++;
188         break;
189     case '?':
190         errflg++;
191     }

192     if (errflg || (optind >= argc)) {

```

```

193     errmsg("Usage: grep [-c|-l|-q|-o] [-r|-R] -hHbnsviw "
194     errmsg("Usage: grep [-c|-l|-q] [-r|-R] -hHbnsviw "
195     "pattern file . . .\n",
196     (char *)NULL);
197     exit(2);
198 }

199     argv = &argv[optind];
200     argc -= optind;
201     nfile = argc - 1;

203     if (strrchr(*argv, '\n') != NULL)
204         regerr(41);

206     if (iflag) {
207         for (arg = *argv; *arg != NULL; ++arg)
208             *arg = (char)tolower((int)((unsigned char)*arg));
209     }

211     if (wflag) {
212         unsigned int wordlen;
213         char *wordbuf;

215         wordlen = strlen(*argv) + 5; /* '\\\ '<' *argv '\\\ '>' '\0' */
216         if ((wordbuf = malloc(wordlen)) == NULL) {
217             errmsg("grep: Out of memory for word\n", (char *)NULL);
218             exit(2);
219         }

221         (void) strcpy(wordbuf, "\\<");
222         (void) strcat(wordbuf, *argv);
223         (void) strcat(wordbuf, "\\>");
224         *arg = wordbuf;
225     }

227     expbuf = compile(*argv, (char *)0, (char *)0);
228     if (regerrno)
229         regerr(regerrno);

231     if (--argc == 0)
232         execute(NULL, 0);
233     else
234         while (argc-- > 0)
235             prepare(++argv);

237     return (nsucc == 2 ? 2 : (nsucc == 0 ? 1 : 0));
238 }

    unchanged_portion_omitted

304 static void
305 execute(const char *file, int base)
306 {
307     char *lbuf, *p;
308     long count;
309     long offset = 0;
310     char *next_ptr = NULL;
311     long next_count = 0;

313     tln = 0;

315     if (prntbuf == NULL) {
316         fw_lPrntBufLen = GBUFSIZ + 1;
317         if ((prntbuf = malloc(fw_lPrntBufLen)) == NULL) {
318             exit(2); /* out of memory - BAIL */
319         }
320     }
    if ((linebuf = malloc(fw_lPrntBufLen)) == NULL) {

```

```

321         exit(2); /* out of memory - BAIL */
322     }
323 }
324
325 if (file == NULL) {
326     temp = 0;
327     file = STDIN_FILENAME;
328 } else if ((temp = open(file + base, O_RDONLY)) == -1) {
329     if (!sflag)
330         errmsg("grep: can't open %s\n", file);
331     nsucc = 2;
332     return;
333 }
334
335 /* read in first block of bytes */
336 if ((count = read(temp, prntbuf, GBUFSIZ)) <= 0) {
337     (void) close(temp);
338
339     if (cflag && !qflag) {
340         if (Hflag || (nfile > 1 && !hflag))
341             (void) fprintf(stdout, "%s:", file);
342         if (!rflag)
343             (void) fprintf(stdout, "%lld\n", tln);
344     }
345     return;
346 }
347
348 lnum = 0;
349 ptr = prntbuf;
350 for (;;) {
351     /* look for next newline */
352     if ((ptrend = memchr(ptr + offset, '\n', count)) == NULL) {
353         offset += count;
354
355         /*
356          * shift unused data to the beginning of the buffer
357          */
358         if (ptr > prntbuf) {
359             (void) memmove(prntbuf, ptr, offset);
360             ptr = prntbuf;
361         }
362
363         /*
364          * re-allocate a larger buffer if this one is full
365          */
366         if (offset + GBUFSIZ > fw_lPrntBufLen) {
367             /*
368              * allocate a new buffer and preserve the
369              * contents...
370              */
371             fw_lPrntBufLen += GBUFSIZ;
372             if ((prntbuf = realloc(prntbuf,
373                 fw_lPrntBufLen)) == NULL)
374                 exit(2);
375
376             /*
377              * set up a bigger linebuffer (this is only used
378              * for case insensitive operations). Contents do
379              * not have to be preserved.
380              */
381             free(linebuf);
382             if ((linebuf = malloc(fw_lPrntBufLen)) == NULL)
383                 exit(2);
384
385             ptr = prntbuf;
386         }

```

```

388         p = prntbuf + offset;
389         if ((count = read(temp, p, GBUFSIZ)) > 0)
390             continue;
391
392         if (offset == 0)
393             /* end of file already reached */
394             break;
395
396         /* last line of file has no newline */
397         ptrend = ptr + offset;
398         nlflag = 0;
399     } else {
400         next_ptr = ptrend + 1;
401         next_count = offset + count - (next_ptr - ptr);
402         nlflag = 1;
403     }
404     lnum++;
405     *ptrend = '\0';
406
407     if (iflag) {
408         /*
409          * Make a lower case copy of the record
410          */
411         p = ptr;
412         for (lbuf = linebuf; p < ptrend; )
413             *lbuf++ = (char)tolower((int)
414                 (unsigned char)*p++);
415         *lbuf = '\0';
416         lbuf = linebuf;
417     } else
418         /*
419          * Use record as is
420          */
421         lbuf = ptr;
422
423     /* lflag only once */
424     if (step(lbuf, expbuf) ^ vflag) {
425         if (oflag) {
426             /*
427              * Only store the matching bits
428              */
429             ptr = loc1;
430             ptrend = loc2;
431         }
432         if (succeed(file) == 1)
433             if ((step(lbuf, expbuf) ^ vflag) && succeed(file) == 1)
434                 break;
435     }
436 #endif /* ! codereview */
437
438     if (!nlflag)
439         break;
440
441     ptr = next_ptr;
442     count = next_count;
443     offset = 0;
444 }
445 (void) close(temp);
446
447 if (cflag && !qflag) {
448     if (Hflag || (!hflag && ((nfile > 1) ||
449         (rflag && outfn))))
450         (void) fprintf(stdout, "%s:", file);
451     (void) fprintf(stdout, "%lld\n", tln);

```

```

452 }
453
454 static int
455 succeed(const char *f)
456 {
457     int nchars;
458     nsucc = (nsucc == 2) ? 2 : 1;
459
460     if (qflag) {
461         /* no need to continue */
462         return (1);
463     }
464
465     if (cflag) {
466         tln++;
467         return (0);
468     }
469
470     if (lflag) {
471         (void) fprintf(stdout, "%s\n", f);
472         return (1);
473     }
474
475     if (Hflag || (!hflag && (nfile > 1 || (rflag && outfn)))) {
476         /* print filename */
477         (void) fprintf(stdout, "%s:", f);
478     }
479
480     if (bflag)
481         /* print block number */
482         (void) fprintf(stdout, "%lld:", (offset_t)
483             ((lseek(temp, (off_t)0, SEEK_CUR) - 1) / BLKSIZE));
484
485     if (nflag)
486         /* print line number */
487         (void) fprintf(stdout, "%lld:", lnum);
488
489     if (nlflag) {
490         /* newline at end of line */
491         *ptrend = '\n';
492         nchars = ptrend - ptr + 1;
493     } else {
494         /* don't write sentinel \0 */
495         nchars = ptrend - ptr;
496     }
497
498     (void) fwrite(ptr, 1, nchars, stdout);
499     return (0);
500 }
501
502 static void
503 regerr(int err)
504 {
505     errmsg("grep: RE error %d: ", err);
506     switch (err) {
507         case 11:
508             err = 0;
509             break;
510         case 16:
511             err = 1;
512             break;
513         case 25:
514             err = 2;
515             break;
516         case 41:
517             err = 3;

```

```

518         break;
519     case 42:
520         err = 4;
521         break;
522     case 43:
523         err = 5;
524         break;
525     case 44:
526         err = 6;
527         break;
528     case 45:
529         err = 7;
530         break;
531     case 46:
532         err = 8;
533         break;
534     case 49:
535         err = 9;
536         break;
537     case 50:
538         err = 10;
539         break;
540     case 67:
541         err = 11;
542         break;
543     default:
544         err = 12;
545         break;
546     }
547
548     errmsg("%s\n", gettext(errstr[err]));
549     exit(2);
550 }

```

```

*****
14225 Sat Sep 14 23:25:09 2013
new/usr/src/man/man1/grep.1
3546 add support for grep -o option
*****
1 \" te
2.\" Copyright (c) 2013 Andrew Stormont. All rights reserved.
3 #endif /* ! codereview */
4.\" Copyright 2012 Nexenta Systems, Inc. All rights reserved.
5.\" Copyright 1989 AT&T
6.\" Copyright (c) 2008, Sun Microsystems, Inc. All Rights Reserved
7.\" Portions Copyright (c) 1992, X/Open Company Limited All Rights Reserved
8.\" Sun Microsystems, Inc. gratefully acknowledges The Open Group for permission
9.\" http://www.opengroup.org/bookstore/.
10.\" The Institute of Electrical and Electronics Engineers and The Open Group, ha
11.\" This notice shall appear on any product containing this material.
12.\" The contents of this file are subject to the terms of the Common Development
13.\" You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE or http:
14.\" When distributing Covered Code, include this CDDL HEADER in each file and in
15.TH GREP 1 "May 3, 2013"
16.SH NAME
17 grep \- search a file for a pattern
18.SH SYNOPSIS
19.LP
20.nf
21 \fB/usr/bin/grep\fR [\fB-c\fR | \fB-l\fR | \fB-q\fR | \fB-o\fR] [\fB-r\fR | \fB-
22 \fB/usr/bin/grep\fR [\fB-c\fR | \fB-l\fR | \fB-q\fR] [\fB-r\fR | \fB-R\fR] [\fB-B-
22 \fB-limited-regular-expression\fR [\fBfilename\fR]...
23 .fi

25 .LP
26 .nf
27 \fB/usr/xpg4/bin/grep\fR [\fB-E\fR | \fB-F\fR] [\fB-c\fR | \fB-l\fR | \fB-q\fR |
28 \fB/usr/xpg4/bin/grep\fR [\fB-E\fR | \fB-F\fR] [\fB-c\fR | \fB-l\fR | \fB-q\fR]
28 [\fB-bHhinsvwx\fR] \fB-e\fR \fIpattern_list\fR... [\fB-f\fR \fIpattern_file\
29 [\fIfile\fR]...
30 .fi

32 .LP
33 .nf
34 \fB/usr/xpg4/bin/grep\fR [\fB-E\fR | \fB-F\fR] [\fB-c\fR | \fB-l\fR | \fB-q\fR |
35 \fB/usr/xpg4/bin/grep\fR [\fB-E\fR | \fB-F\fR] [\fB-c\fR | \fB-l\fR | \fB-q\fR]
35 [\fB-bHhinsvwx\fR] [\fB-e\fR \fIpattern_list\fR]... \fB-f\fR \fIpattern_file
36 [\fIfile\fR]...
37 .fi

39 .LP
40 .nf
41 \fB/usr/xpg4/bin/grep\fR [\fB-E\fR | \fB-F\fR] [\fB-c\fR | \fB-l\fR | \fB-q\fR |
42 \fB/usr/xpg4/bin/grep\fR [\fB-E\fR | \fB-F\fR] [\fB-c\fR | \fB-l\fR | \fB-q\fR]
42 [\fB-bHhinsvwx\fR] \fIpattern\fR [\fIfile\fR]...
43 .fi

45 .SH DESCRIPTION
46 .sp
47 .LP
48 The \fBgrep\fR utility searches text files for a pattern and prints all lines
49 that contain that pattern. It uses a compact non-deterministic algorithm.
50 .sp
51 .LP
52 Be careful using the characters \fB$\fR, \fB*\fR, \fB[\fR, \fB^\fR, \fB|\fR,
53 \fB(\fR, \fB)\fR, and \fB)\fR in the \fIpattern_list\fR because they are also
54 meaningful to the shell. It is safest to enclose the entire \fIpattern_list\fR
55 in single quotes \fBa'\fR&...\fBa'\fR&.
56 .sp
57 .LP

```

```

58 If no files are specified, \fBgrep\fR assumes standard input. Normally, each
59 line found is copied to standard output. The file name is printed before each
60 line found if there is more than one input file.
61 .SS "/usr/bin/grep"
62 .sp
63 .LP
64 The \fB/usr/bin/grep\fR utility uses limited regular expressions like those
65 described on the \fBregexp\fR(5) manual page to match the patterns.
66 .SS "/usr/xpg4/bin/grep"
67 .sp
68 .LP
69 The options \fB-E\fR and \fB-F\fR affect the way \fB/usr/xpg4/bin/grep\fR
70 interprets \fIpattern_list\fR. If \fB-E\fR is specified,
71 \fB/usr/xpg4/bin/grep\fR interprets \fIpattern_list\fR as a full regular
72 expression (see \fB-E\fR for description). If \fB-F\fR is specified,
73 \fBgrep\fR interprets \fIpattern_list\fR as a fixed string. If neither are
74 specified, \fBgrep\fR interprets \fIpattern_list\fR as a basic regular
75 expression as described on \fBregexp\fR(5) manual page.
76 .SH OPTIONS
77 .sp
78 .LP
79 The following options are supported for both \fB/usr/bin/grep\fR and
80 \fB/usr/xpg4/bin/grep\fR:
81 .sp
82 .ne 2
83 .na
84 \fB\b\b\fR
85 .ad
86 .RS 6n
87 Precedes each line by the block number on which it was found. This can be
88 useful in locating block numbers by context (first block is 0).
89 .RE

91 .sp
92 .ne 2
93 .na
94 \fB\b-c\fR
95 .ad
96 .RS 6n
97 Prints only a count of the lines that contain the pattern.
98 .RE

100 .sp
101 .ne 2
102 .na
103 \fB\b-H\fR
104 .ad
105 .RS 6n
106 Precedes each line by the name of the file containing the matching line.
107 .RE

109 .sp
110 .ne 2
111 .na
112 \fB\b-h\fR
113 .ad
114 .RS 6n
115 Prevents the name of the file containing the matching line from being prepended
116 to that line. Used when searching multiple files.
117 .RE

119 .sp
120 .ne 2
121 .na
122 \fB\b-i\fR
123 .ad

```

```

124 .RS 6n
125 Ignores upper/lower case distinction during comparisons.
126 .RE

128 .sp
129 .ne 2
130 .na
131 \fB\fB-l\fR\fR
132 .ad
133 .RS 6n
134 Prints only the names of files with matching lines, separated by NEWLINE
135 characters. Does not repeat the names of files when the pattern is found more
136 than once.
137 .RE

139 .sp
140 .ne 2
141 .na
142 \fB\fB-n\fR\fR
143 .ad
144 .RS 6n
145 Precedes each line by its line number in the file (first line is 1).
146 .RE

148 .sp
149 .ne 2
150 .na
151 \fB\fB-o\fR\fR
152 .ad
153 .RS 6n
154 Print only the matching part of the line.
155 .RE

157 .sp
158 .ne 2
159 .na
160 #endif /* ! codereview */
161 \fB\fB-r\fR\fR
162 .ad
163 .RS 6n
164 Read all files under each directory, recursively. Follow symbolic links on
165 the command line, but skip symlinks that are encountered recursively. If file
166 is a device, FIFO, or socket, skip it.
167 .RE

169 .sp
170 .ne 2
171 .na
172 \fB\fB-R\fR\fR
173 .ad
174 .RS 6n
175 Read all files under each directory, recursively, following all symbolic links.
176 .RE

178 .sp
179 .ne 2
180 .na
181 \fB\fB-q\fR\fR
182 .ad
183 .RS 6n
184 Quiet. Does not write anything to the standard output, regardless of matching
185 lines. Exits with zero status if an input line is selected.
186 .RE

188 .sp
189 .ne 2

```

```

190 .na
191 \fB\fB-s\fR\fR
192 .ad
193 .RS 6n
194 Suppresses error messages about nonexistent or unreadable files.
195 .RE

197 .sp
198 .ne 2
199 .na
200 \fB\fB-v\fR\fR
201 .ad
202 .RS 6n
203 Prints all lines except those that contain the pattern.
204 .RE

206 .sp
207 .ne 2
208 .na
209 \fB\fB-w\fR\fR
210 .ad
211 .RS 6n
212 Searches for the expression as a word as if surrounded by \fB\e<\fR and
213 \fB\e>\fR&.
214 .RE

216 .SS "/usr/xpg4/bin/grep"
217 .sp
218 .LP
219 The following options are supported for \fB/usr/xpg4/bin/grep\fR only:
220 .sp
221 .ne 2
222 .na
223 \fB\fB-e\fR \fIpattern_list\fR\fR
224 .ad
225 .RS 19n
226 Specifies one or more patterns to be used during the search for input. Patterns
227 in \fIpattern_list\fR must be separated by a NEWLINE character. A null pattern
228 can be specified by two adjacent newline characters in \fIpattern_list\fR.
229 Unless the \fB-E\fR or \fB-F\fR option is also specified, each pattern is
230 treated as a basic regular expression. Multiple \fB-e\fR and \fB-f\fR options
231 are accepted by \fBgrep\fR. All of the specified patterns are used when
232 matching lines, but the order of evaluation is unspecified.
233 .RE

235 .sp
236 .ne 2
237 .na
238 \fB\fB-E\fR\fR
239 .ad
240 .RS 19n
241 Matches using full regular expressions. Treats each pattern specified as a full
242 regular expression. If any entire full regular expression pattern matches an
243 input line, the line is matched. A null full regular expression matches every
244 line. Each pattern is interpreted as a full regular expression as described on
245 the \fBregex\fR(5) manual page, except for \fB\e(\fR and \fB\e)\fR, and
246 including:
247 .RS +4
248 .TP
249 1.
250 A full regular expression followed by \fB+\fR that matches one or more
251 occurrences of the full regular expression.
252 .RE
253 .RS +4
254 .TP
255 2.

```

```

256 A full regular expression followed by \fB?\fR that matches 0 or 1
257 occurrences of the full regular expression.
258 .RE
259 .RS +4
260 .TP
261 3.
262 Full regular expressions separated by | or by a new-line that match strings
263 that are matched by any of the expressions.
264 .RE
265 .RS +4
266 .TP
267 4.
268 A full regular expression that is enclosed in parentheses \fB()\fR for
269 grouping.
270 .RE
271 The order of precedence of operators is \fB[\]\fR, then \fB*\|?\|+\fR, then
272 concatenation, then | and new-line.
273 .RE

275 .sp
276 .ne 2
277 .na
278 \fB\fB-f\fR \fIpattern_file\fR\fR
279 .ad
280 .RS 19n
281 Reads one or more patterns from the file named by the path name
282 \fIpattern_file\fR. Patterns in \fIpattern_file\fR are terminated by a NEWLINE
283 character. A null pattern can be specified by an empty line in
284 \fIpattern_file\fR. Unless the \fB-E\fR or \fB-F\fR option is also specified,
285 each pattern is treated as a basic regular expression.
286 .RE

288 .sp
289 .ne 2
290 .na
291 \fB\fB-F\fR\fR
292 .ad
293 .RS 19n
294 Matches using fixed strings. Treats each pattern specified as a string instead
295 of a regular expression. If an input line contains any of the patterns as a
296 contiguous sequence of bytes, the line is matched. A null string matches every
297 line. See \fBfgrep\fR(1) for more information.
298 .RE

300 .sp
301 .ne 2
302 .na
303 \fB\fB-x\fR\fR
304 .ad
305 .RS 19n
306 Considers only input lines that use all characters in the line to match an
307 entire fixed string or regular expression to be matching lines.
308 .RE

310 .SH OPERANDS
311 .sp
312 .LP
313 The following operands are supported:
314 .sp
315 .ne 2
316 .na
317 \fB\fIfile\fR\fR
318 .ad
319 .RS 8n
320 A path name of a file to be searched for the patterns. If no \fIfile\fR
321 operands are specified, the standard input is used.

```

```

322 .RE

324 .SS "/usr/bin/grep"
325 .sp
326 .ne 2
327 .na
328 \fB\fIpattern\fR\fR
329 .ad
330 .RS 11n
331 Specifies a pattern to be used during the search for input.
332 .RE

334 .SS "/usr/xpg4/bin/grep"
335 .sp
336 .ne 2
337 .na
338 \fB\fIpattern\fR\fR
339 .ad
340 .RS 11n
341 Specifies one or more patterns to be used during the search for input. This
342 operand is treated as if it were specified as \fB-e\fR \fIpattern_list\fR.
343 .RE

345 .SH USAGE
346 .sp
347 .LP
348 The \fB-e\fR \fIpattern_list\fR option has the same effect as the
349 \fIpattern_list\fR operand, but is useful when \fIpattern_list\fR begins with
350 the hyphen delimiter. It is also useful when it is more convenient to provide
351 multiple patterns as separate arguments.
352 .sp
353 .LP
354 Multiple \fB-e\fR and \fB-f\fR options are accepted and \fBgrep\fR uses all of
355 the patterns it is given while matching input text lines. Notice that the order
356 of evaluation is not specified. If an implementation finds a null string as a
357 pattern, it is allowed to use that pattern first, matching every line, and
358 effectively ignore any other patterns.
359 .sp
360 .LP
361 The \fB-q\fR option provides a means of easily determining whether or not a
362 pattern (or string) exists in a group of files. When searching several files,
363 it provides a performance improvement (because it can quit as soon as it finds
364 the first match) and requires less care by the user in choosing the set of
365 files to supply as arguments (because it exits zero if it finds a match even if
366 \fBgrep\fR detected an access or read error on earlier file operands).
367 .SS "Large File Behavior"
368 .sp
369 .LP
370 See \fBlargefile\fR(5) for the description of the behavior of \fBgrep\fR when
371 encountering files greater than or equal to 2 Gbyte ( 2^31 bytes).
372 .SH EXAMPLES
373 .LP
374 \fBExample 1 \fRFinding All Uses of a Word
375 .sp
376 .LP
377 To find all uses of the word "\fBposix\fR" (in any case) in the file
378 \fBtext.mm\fR, and write with line numbers:

380 .sp
381 .in +2
382 .nf
383 example% \fB/usr/bin/grep -i -n posix text.mm\fR
384 .fi
385 .in -2
386 .sp

```

```

388 .LP
389 \fBExample 2 \fRFinding All Empty Lines
390 .sp
391 .LP
392 To find all empty lines in the standard input:

394 .sp
395 .in +2
396 .nf
397 example% \fB/usr/bin/grep ^$\fR
398 .fi
399 .in -2
400 .sp

402 .sp
403 .LP
404 or

406 .sp
407 .in +2
408 .nf
409 example% \fB/usr/bin/grep -v .\fR
410 .fi
411 .in -2
412 .sp

414 .LP
415 \fBExample 3 \fRFinding Lines Containing Strings
416 .sp
417 .LP
418 All of the following commands print all lines containing strings \fBabc\fR or
419 \fBdef\fR or both:

421 .sp
422 .in +2
423 .nf
424 example% \fB/usr/xpg4/bin/grep 'abc
425 def'\fR
426 example% \fB/usr/xpg4/bin/grep -e 'abc
427 def'\fR
428 example% \fB/usr/xpg4/bin/grep -e 'abc' -e 'def'\fR
429 example% \fB/usr/xpg4/bin/grep -E 'abc|def'\fR
430 example% \fB/usr/xpg4/bin/grep -E -e 'abc|def'\fR
431 example% \fB/usr/xpg4/bin/grep -E -e 'abc' -e 'def'\fR
432 example% \fB/usr/xpg4/bin/grep -E 'abc
433 def'\fR
434 example% \fB/usr/xpg4/bin/grep -E -e 'abc
435 def'\fR
436 example% \fB/usr/xpg4/bin/grep -F -e 'abc' -e 'def'\fR
437 example% \fB/usr/xpg4/bin/grep -F 'abc
438 def'\fR
439 example% \fB/usr/xpg4/bin/grep -F -e 'abc
440 def'\fR
441 .fi
442 .in -2
443 .sp

445 .LP
446 \fBExample 4 \fRFinding Lines with Matching Strings
447 .sp
448 .LP
449 Both of the following commands print all lines matching exactly \fBabc\fR or
450 \fBdef\fR:

452 .sp
453 .in +2

```

```

454 .nf
455 example% \fB/usr/xpg4/bin/grep -E '^abc$ ^def$'\fR
456 example% \fB/usr/xpg4/bin/grep -F -x 'abc def'\fR
457 .fi
458 .in -2
459 .sp

461 .SH ENVIRONMENT VARIABLES
462 .sp
463 .LP
464 See \fBenviron\fR(5) for descriptions of the following environment variables
465 that affect the execution of \fBgrep\fR: \fBBLANG\fR, \fBLC_ALL\fR,
466 \fBLC_COLLATE\fR, \fBLC_CTYPE\fR, \fBLC_MESSAGES\fR, and \fBNLSPATH\fR.
467 .SH EXIT STATUS
468 .sp
469 .LP
470 The following exit values are returned:
471 .sp
472 .ne 2
473 .na
474 \fB0\fR
475 .ad
476 .RS 5n
477 One or more matches were found.
478 .RE

480 .sp
481 .ne 2
482 .na
483 \fB1\fR
484 .ad
485 .RS 5n
486 No matches were found.
487 .RE

489 .sp
490 .ne 2
491 .na
492 \fB2\fR
493 .ad
494 .RS 5n
495 Syntax errors or inaccessible files (even if matches were found).
496 .RE

498 .SH ATTRIBUTES
499 .sp
500 .LP
501 See \fBattributes\fR(5) for descriptions of the following attributes:
502 .SS "/usr/bin/grep"
503 .sp

505 .sp
506 .TS
507 box;
508 c | c
509 l | l .
510 ATTRIBUTE TYPE ATTRIBUTE VALUE
511 _
512 CSI Not Enabled
513 .TE

515 .SS "/usr/xpg4/bin/grep"
516 .sp

518 .sp
519 .TS

```



```
520 box;
521 c | c
522 l | l .
523 ATTRIBUTE TYPE ATTRIBUTE VALUE
524 _
525 CSI Enabled
526 _
527 Interface Stability Committed
528 _
529 Standard See \fBstandards\fR(5).
530 .TE

532 .SH SEE ALSO
533 .sp
534 .LP
535 \fBbegrep\fR(1), \fBfgrep\fR(1), \fBsed\fR(1), \fBsh\fR(1), \fBattributes\fR(5),
536 \fBenviron\fR(5), \fBlargefile\fR(5), \fBregex\fR(5), \fBregexp\fR(5),
537 \fBstandards\fR(5)
538 .SH NOTES
539 .SS "/usr/bin/grep"
540 .sp
541 .LP
542 Lines are limited only by the size of the available virtual memory. If there is
543 a line with embedded nulls, \fBgrep\fR only matches up to the first null. If
544 the line matches, the entire line is printed.
545 .SS "/usr/xpg4/bin/grep"
546 .sp
547 .LP
548 The results are unspecified if input files contain lines longer than
549 \fBLINE_MAX\fR bytes or contain binary data. \fBLINE_MAX\fR is defined in
550 \fB/usr/include/limits.h\fR.
```