

new/usr/src/cmd/egrep/egrep.y

26107 Thu May 30 19:29:52 2013

new/usr/src/cmd/egrep/egrep.y

3737 grep does not support -H option

```
1 %{
2 /*
3 * CDDL HEADER START
4 *
5 * The contents of this file are subject to the terms of the
6 * Common Development and Distribution License, Version 1.0 only
7 * (the "License"). You may not use this file except in compliance
8 * with the License.
9 *
10 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
11 * or http://www.opensolaris.org/os/licensing.
12 * See the License for the specific language governing permissions
13 * and limitations under the License.
14 *
15 * When distributing Covered Code, include this CDDL HEADER in each
16 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
17 * If applicable, add the following below this CDDL HEADER, with the
18 * fields enclosed by brackets "[]" replaced with your own identifying
19 * information: Portions Copyright [yyyy] [name of copyright owner]
20 *
21 * CDDL HEADER END
22 */
23 %}
24 /*
25 * Copyright 2005 Sun Microsystems, Inc. All rights reserved.
26 * Use is subject to license terms.
27 */

28 /* Copyright (c) 1984, 1986, 1987, 1988, 1989 AT&T */
29 /* All Rights Reserved */

30 /* Copyright (c) 1987, 1988 Microsoft Corporation */
31 /* All Rights Reserved */

32 /*
33 * Copyright (c) 1987, 1988 Microsoft Corporation
34 * All Rights Reserved */

35 /*
36 * Copyright 2013 Damian Bogel. All rights reserved.
37 */
38 #pragma ident "%Z%%M% %I%      %E% SMI"

39 /*
40 * egrep -- print lines containing (or not containing) a regular expression
41 *
42 *     status returns:
43 *             0 - ok, and some matches
44 *             1 - ok, but no matches
45 *             2 - some error; matches irrelevant
46 */
47 %token CHAR MCHAR DOT MDOT CCL NCCL MCCL NMCL OR CAT STAR PLUS QUEST
48 %left OR
49 %left CHAR MCHAR DOT CCL NCCL MCCL NMCL `(
50 %left CAT
51 %left STAR PLUS QUEST

52 %
53 #include <stdio.h>
54 #include <ctype.h>
55 #include <memory.h>
56 #include <wchar.h>
57 #include <wctype.h>
```

1

new/usr/src/cmd/egrep/egrep.y

59 #include <widec.h>

60 #include <stdlib.h>

61 #include <limits.h>

62 #include <locale.h>

64 #define STDIN_FILENO gettext("(standard input)")

66 #endif /* ! codereview */

67 #define BLKSIZE 512 /* size of reported disk blocks */

68 #define EBUFSIZ 8192

69 #define MAXLIN 350

70 #define NCHARS 256

71 #define MAXPOS 4000

72 #define NSTATES 64

73 #define FINAL -1

74 #define RIGHT '\n' /* serves as record separator and as \$ */

75 #define LEFT '\n' /* beginning of line */

76 int gotofn[NSTATES][NCHARS];

77 int state[NSTATES];

78 int out[NSTATES];

79 int line = 1;

80 int *name;

81 int *left;

82 int *right;

83 int *parent;

84 int *foll;

85 int *positions;

86 char *chars;

87 wchar_t *lower;

88 wchar_t *upper;

89 int maxlin, maxclin, maxwclin, maxpos;

90 int nxtpos = 0;

91 int inxtpos;

92 int nxtchar = 0;

93 int *tmpstat;

94 int *initstat;

95 int istat;

96 int nstate = 1;

97 int xstate;

98 int count;

99 int icount;

100 char *input;

103 wchar_t lyylval;

104 wchar_t nextch();

105 wchar_t maxmin();

106 int compare();

107 void overflo();

109 char reinit = 0;

111 long long lnum;

112 int bflag;

113 int cflag;

114 int eflag;

115 int fflag;

116 int Hflag;

117 #endif /* ! codereview */

118 int hflag;

119 int iflag;

120 int lflag;

121 int nflag;

122 int qflag;

64 int sflag;

123 int vflag;

2

```

124 int      nfile;
125 long long blkno;
126 long long tln;
127 int      nsucc;
128 int      badbotch;
129 extern char *optarg;
130 extern int optind;

132 int      f;
133 FILE    *expfile;
134 }

136 %%
137 s:   t
138     {
139         unary(FINAL, $1);
140         line--;
141     }
142 ;
143 t:   b r
144     { $$ = node(CAT, $1, $2); }
145     | OR b r OR
146     { $$ = node(CAT, $2, $3); }
147     | OR b r
148     { $$ = node(CAT, $2, $3); }
149     | b r OR
150     { $$ = node(CAT, $1, $2); }
151 ;
152 b:
153     { /* if(multibyte)
154        $$ = mdotenter();
155     else */
156        $$ = enter(DOT);
157     $$ = unary(STAR, $$);
158 }
159 ;
160 r:   CHAR
161     { $$ = iflag && isalpha($1) ?
162       node(OR, enter(tolower($1)), enter(toupper($1))) : enter($1); }
163     | MCHAR
164     { $$ = (iflag && iswalPHA(llylval)) ?
165       node(OR, mchar(towlower(llylval)), mchar(towupper(llylval))) :
166       mchar(llylval); }
167     | DOT
168     { if(multibyte)
169        $$ = mdotenter();
170     else
171        $$ = enter(DOT);
172 }
173     | CCL
174     { $$ = cccenter(CCL); }
175     | NCCL
176     { $$ = cccenter(NCCL); }
177     | MCCL
178     { $$ = ccl(CCL); }
179     | NMCCCL
180     { $$ = ccl(NCCL); }
181 ;
183 r:   r OR r
184     { $$ = node(OR, $1, $3); }
185     | r r %prec CAT
186     { $$ = node(CAT, $1, $2); }
187     | r STAR
188     { $$ = unary(STAR, $1); }
189     | r PLUS

```

```

190             { $$ = unary(PLUS, $1); }
191             | r QUEST
192             { $$ = unary(QUEST, $1); }
193             | '(' r ')'
194             { $$ = $2; }
195             | error
196             ;

198 /**
199 void    add(int *, int);
200 void    clearg(void);
201 void    execute(char *);
202 void    follow(int);
203 int     mgetc(void);
204 void    synerror(void);

207 void
208 yyerror(char *s)
209 {
210     fprintf(stderr, "egrep: %s\n", s);
211     exit(2);
212 }

_____unchanged_portion_omitted_____

652 #define USAGE "[ -bchilnsv ] [ -e exp ] [ -f file ] [ strings ] [ file ] ..."

654 int
655 main(int argc, char **argv)
656 {
657     char c;
658     char nl = '\n';
659     int errflag = 0;
660
661     (void)setlocale(LC_ALL, "");

663 #if !defined(TEXT_DOMAIN)          /* Should be defined by cc -D */
664     #define TEXT_DOMAIN "SYS_TEST" /* Use this only if it weren't. */
665 #endif
666     (void) textdomain(TEXT_DOMAIN);

668     while((c = getopt(argc, argv, "ybcie:f:Hlnvs")) != -1)
669     while((c = getopt(argc, argv, "ybcie:f:hlnvs")) != -1)
670         switch(c) {
671             case 'b':
672                 bflag++;
673                 continue;
675             case 'c':
676                 cflag++;
677                 continue;
679             case 'e':
680                 eflag++;
681                 input = optarg;
682                 continue;
684             case 'f':
685                 fflag++;
686                 expfile = fopen(optarg, "r");
687                 if(expfile == NULL) {
688                     fprintf(stderr,
689                             gettext("egrep: can't open %s\n"), optarg);
690                     exit(2);
691                 }

```

```

692             continue;
693
694         case 'H':
695             if (!lflag) /* H is excluded by l as in GNU grep */
696                 Hflag++;
697             hflag = 0; /* H excludes h */
698             continue;
699
700 #endif /* ! codereview */
701     case 'h':
702         hflag++;
703         Hflag = 0; /* h excludes H */
704 #endif /* ! codereview */
705     continue;
706
707     case 'y':
708     case 'i':
709         iflag++;
710         continue;
711
712     case 'l':
713         lflag++;
714         Hflag = 0; /* l excludes H */
715 #endif /* ! codereview */
716     continue;
717
718     case 'n':
719         nflag++;
720         continue;
721
722     case 'q':
723     case 's': /* Solaris: legacy option */
724         qflag++;
725     case 'S':
726         sflag++;
727         continue;
728
729     case 'v':
730         vflag++;
731         continue;
732
733     case '?':
734         errflag++;
735     }
736     if (errflag || ((argc <= 0) && !fflag && !eflag)) {
737         fprintf(stderr, gettext("usage: egrep %s\n"), gettext(USAGE));
738         exit(2);
739     }
740     if(!eflag && !fflag) {
741         input = argv[optind];
742         optind++;
743     }
744
745     argc -= optind;
746     argv = &argv[optind];
747
748     /* allocate initial space for arrays */
749     if((name = (int *)malloc(MAXLIN*sizeof(int))) == (int *)0)
750         overflow();
751     if((left = (int *)malloc(MAXLIN*sizeof(int))) == (int *)0)
752         overflow();
753     if((right = (int *)malloc(MAXLIN*sizeof(int))) == (int *)0)
754         overflow();
755     if((parent = (int *)malloc(MAXLIN*sizeof(int))) == (int *)0)
756         overflow();
757     if((foll = (int *)malloc(MAXLIN*sizeof(int))) == (int *)0)
758         overflow();

```

```

759         overflow();
760     if((tmpstat = (int *)malloc(MAXLIN*sizeof(int))) == (int *)0)
761         overflow();
762     if((initstat = (int *)malloc(MAXLIN*sizeof(int))) == (int *)0)
763         overflow();
764     if((chars = (char *)malloc(MAXLIN)) == (char *)0)
765         overflow();
766     if((lower = (wchar_t *)malloc(MAXLIN*sizeof(wchar_t))) == (wchar_t *)0)
767         overflow();
768     if((upper = (wchar_t *)malloc(MAXLIN*sizeof(wchar_t))) == (wchar_t *)0)
769         overflow();
770     if((positions = (int *)malloc(MAXPOS*sizeof(int))) == (int *)0)
771         overflow();
772     maxlin = MAXLIN;
773     maxclin = MAXLIN;
774     maxwclin = MAXLIN;
775     maxpos = MAXPOS;
776     ypparse();
777
778     cfollline(-1);
779     cgtofn();
780     nfile = argc;
781     if (argc<=0) {
782         execute(0);
783     } else while (--argc >= 0) {
784         if (reinit == 1) clearg();
785         execute(*argv++);
786     }
787     return (badbatch ? 2 : nsucc==0);
788
789 void
790 execute(char *file)
791 {
792     char *p;
793     int cstat;
794     wchar_t c;
795     int t;
796     long count;
797     long count1, count2;
798     long nchars;
799     int succ;
800     char *ptr, *ptrend, *lastptr;
801     char *buf;
802     long lBufSiz;
803     FILE *f;
804     int nlflag;
805
806     lBufSiz = EBUFSIZ;
807     if ((buf = malloc (lBufSiz + EBUFSIZ)) == NULL) {
808         exit (2); /* out of memory - BAIL */
809     }
810
811     if (file) {
812         if ((f = fopen(file, "r")) == NULL) {
813             fprintf(stderr,
814                     gettext("egrep: can't open %s\n"), file);
815             badbatch=1;
816             return;
817         }
818     } else {
819         file = "<stdin>";
820         f = stdin;
821         file = STDIN_FILENAME;

```

```

821 #endif /* ! codereview */
822 }
823     inum = 1;
824     tln = 0;
825     if((count = read(fileno(f), buf, EBUFSIZ)) <= 0) {
826         fclose(f);
827
828         if (cflag && !qflag) {
829             if (Hflag || (nfile > 1 && !hflag))
830                 if (oflag) {
831                     if (nfile>1 && !hflag)
832                         fprintf(stdout, "%s:", file);
833                     fprintf(stdout, "%lld\n", tln);
834                 }
835             return;
836         }
837
838         blkno = count;
839         ptr = buf;
840         for(;;) {
841             if((ptrend = memchr(ptr, '\n', buf + count - ptr)) == NULL) {
842                 /*
843                  move the unused partial record to the head of th
844
845                 if (ptr > buf) {
846                     count = buf + count - ptr;
847                     memmove (buf, ptr, count);
848                     ptr = buf;
849
850                     /*
851                     Get a bigger buffer if this one is full
852
853                     if(count > lBufSiz) {
854                         /*
855                           expand the buffer
856
857                         lBufSiz += EBUFSIZ;
858                         if ((buf = realloc (buf, lBufSiz + EBUFSIZ)) ==
859                             exit (2); /* out of memory - BAIL */
860
861                         ptr = buf;
862
863                         p = buf + count;
864                         if((count1 = read(fileno(f), p, EBUFSIZ)) > 0) {
865                             count += count1;
866                             blkno += count1;
867                             continue;
868
869                         }
870                         ptrend = ptr + count;
871                         nlflag = 0;
872                     } else
873                         nlflag = 1;
874                     *ptrend = '\n';
875                     p = ptr;
876                     lastptr = ptr;
877                     cstat = istat;
878                     succ = 0;
879                     for(;;) {
880                         if(out[cstat]) {
881                             if(multibyte && p > ptr) {
882                                 wchar_t wchar;
883                                 int length;
884                                 char *endptr = p;

```

```

885         p = lastptr;
886         while(p < endptr) {
887             length = mbtowc(&wchar, p, MB_LE
888             if(length <= 1)
889                 p++;
890             else
891                 p += length;
892
893             if(p == endptr) {
894                 succ = !vflag;
895                 break;
896             }
897             cstat = 1;
898             length = mbtowc(&wchar, lastptr, MB_LEN_
899             if(length <= 1)
900                 lastptr++;
901             else
902                 lastptr += length;
903             p = lastptr;
904             continue;
905
906             succ = !vflag;
907             break;
908
909             c = (unsigned char)*p++;
910             if ((t = gotofn[cstat][c]) == 0)
911                 cstat = nxtst(cstat, c);
912             else
913                 cstat = t;
914             if(c == RIGHT) {
915                 if(out[cstat]) {
916                     succ = !vflag;
917                     break;
918                 }
919                 succ = vflag;
920                 break;
921             }
922
923             if (succ) {
924                 if(succ) {
925                     nsucc = 1;
926                     if (lflag || qflag) {
927                         if (!qflag)
928                             (void) printf("%s\n", file);
929                         if (cflag) tln++;
930                         else if (sflag)
931                             ; /* ugh */
932                         else if (lflag) {
933                             printf("%s\n", file);
934                             fclose(f);
935                             return;
936
937                     }
938                     if (cflag) {
939                         tln++;
940                     } else {
941                         if (Hflag || (nfile > 1 && !hflag))
942                             printf("%s:", file);
943                         else {
944                             if (nfile > 1 && !hflag)
945                                 printf(gettext("%s:"), file);
946                             if (bflag) {
947                                 nchars = blkno - (buf + count - ptrend)
948                                 if(nlflag)
949                                     nchars++;
950                                 printf("%lld:", nchars/BLKSIZE);
951                             }
952                         }
953                     }
954                 }
955             }

```

```
942         if (nflag)
943             printf("%lld:", lnum);
944         if(nlflag)
945             nchars = ptrend - ptr + 1;
946         else
947             nchars = ptrend - ptr;
948         fwrite(ptr, (size_t)1, (size_t)nchars, stdout);
949     }
950 }
951 if(!nlflag)
952     break;
953 ptr = ptrend + 1;
954 if(ptr >= buf + count) {
955     ptr = buf;
956     if((count = read(fileno(f), buf, EBUFSIZ)) <= 0)
957         break;
958     blkno += count;
959 }
960 lnum++;
961 if (reinit == 1)
962     clearg();
963 }
964 fclose(f);
965 if (cflag && !qflag) {
966     if (hflag || (nfile > 1 && !hflag))
967         printf("%s:", file);
968     if (cflag) {
969         if (nfile > 1 && !hflag)
970             printf(gettext("%s:"), file);
971         printf("%lld\n", tln);
972     }
973 }
```

unchanged_portion_omitted_

new/usr/src/cmd/fgrep/fgrep.c

```
*****
14443 Thu May 30 19:29:52 2013
new/usr/src/cmd/fgrep/fgrep.c
3737 grep does not support -H 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 */
26 /*
27 * Copyright (c) 1984, 1986, 1987, 1988, 1989 AT&T */
28 * All Rights Reserved */
29 /*
30 * Copyright (c) 1987, 1988 Microsoft Corporation */
31 * All Rights Reserved */
32 /*
33 * Copyright 2013 Damian Bogel. All rights reserved.
34 */
35 */
36 #pragma ident "%Z%%M% %I% %E% SMI"

37 /*
38 * fgrep -- print all lines containing any of a set of keywords
39 *
40 * status returns:
41 *      0 - ok, and some matches
42 *      1 - ok, but no matches
43 *      2 - some error
44 */

45 #include <stdio.h>
46 #include <ctype.h>
47 #include <sys/types.h>
48 #include <sys/stat.h>
49 #include <stdlib.h>
50 #include <string.h>
51 #include <locale.h>
52 #include <libintl.h>
53 #include <euc.h>
54 #include <fcntl.h>
55 #include <getwidth.h>

56 #include <getwidth.h>

57 #include <getwidth.h>

58 eucwidth_t WW;
59 #define WIDTH1 WW._eucwl
60 #define WIDTH2 WW._eucw2
61 #define WIDTH3 WW._eucw3
62 #define MULTI_BYTE WW._multibyte
63 #define GETONE(lc, p) \
64     cw = ISASCII(lc = (unsigned char)*p++) ? 1 : \
65         (ISSET2(lc) ? WIDTH2 : \
66         (ISSET3(lc) ? WIDTH3 : WIDTH1)); \
67     if (--cw > --ccount) { \
68         cw -= ccount; \
69         while (ccount--) \
70             lc = (lc << 7) | ((*p++) & 0177); \
71         if (p >= &buf[fw_lBufsiz + BUFSIZ]) { \
72             if (nlp == buf) { \
73                 /* Increase the buffer size */ \
74                 fw_lBufsiz += BUFSIZ; \
75                 if ((buf = realloc(buf, \
76                     fw_lBufsiz + BUFSIZ)) == NULL) { \
77                     exit(2); /* out of memory */ \
78                 } \
79                 nlp = buf; \
80                 p = &buf[fw_lBufsiz]; \
81             } else { \
82                 /* shift the buffer contents down */ \
83                 (void) memmove(buf, nlp, \
84                     &buf[fw_lBufsiz + BUFSIZ] - nlp); \
85                 p -= nlp - buf; \
86                 nlp = buf; \
87             } \
88         } \
89         if (p > &buf[fw_lBufsiz]) { \
90             if ((ccount = fread(p, sizeof (char), \
91                 &buf[fw_lBufsiz + BUFSIZ] - p, fptr)) \ 
92                 <= 0) break; \
93             } else if ((ccount = fread(p, \
94                 sizeof (char), BUFSIZ, fptr)) <= 0) \
95                 break; \
96             blkno += (long long)ccount; \
97         } \
98         ccount -= cw; \
99         while (cw--) \
100             lc = (lc << 7) | ((*p++) & 0177); \
101     } \
102     /* \
103      * The same() macro and letter() function were inserted to allow for \
104      * the -i option work for the multi-byte environment. \
105      */ \
106     wchar_t letter(); \
107 #define same(a, b) \
108     (a == b || iflag && (!MULTI_BYTE || ISASCII(a)) && (a ^ b) == ' ' && \
109     letter(a) == letter(b)) \
110 letter(a) == letter(b)

111 #define STDIN_FILENO gettext("(standard input)") \
112 #endif /* ! codereview */

113 #define QSIZE 400
114 struct words {
115     wchar_t inp;
116     char out;
117     struct words *nst;
118     struct words *link;
119     struct words *fail;
120 } *w = NULL, *smax, *q;
```

1

new/usr/src/cmd/fgrep/fgrep.c

```
61 #define WIDTH2 WW._eucw2
62 #define WIDTH3 WW._eucw3
63 #define MULTI_BYTE WW._multibyte
64 #define GETONE(lc, p) \
65     cw = ISASCII(lc = (unsigned char)*p++) ? 1 : \
66         (ISSET2(lc) ? WIDTH2 : \
67         (ISSET3(lc) ? WIDTH3 : WIDTH1)); \
68     if (--cw > --ccount) { \
69         cw -= ccount; \
70         while (ccount--) \
71             lc = (lc << 7) | ((*p++) & 0177); \
72         if (p >= &buf[fw_lBufsiz + BUFSIZ]) { \
73             if (nlp == buf) { \
74                 /* Increase the buffer size */ \
75                 fw_lBufsiz += BUFSIZ; \
76                 if ((buf = realloc(buf, \
77                     fw_lBufsiz + BUFSIZ)) == NULL) { \
78                     exit(2); /* out of memory */ \
79                 } \
80                 nlp = buf; \
81                 p = &buf[fw_lBufsiz]; \
82             } else { \
83                 /* shift the buffer contents down */ \
84                 (void) memmove(buf, nlp, \
85                     &buf[fw_lBufsiz + BUFSIZ] - nlp); \
86                 p -= nlp - buf; \
87                 nlp = buf; \
88             } \
89         } \
90         if (p > &buf[fw_lBufsiz]) { \
91             if ((ccount = fread(p, sizeof (char), \
92                 &buf[fw_lBufsiz + BUFSIZ] - p, fptr)) \ 
93                 <= 0) break; \
94             } else if ((ccount = fread(p, \
95                 sizeof (char), BUFSIZ, fptr)) <= 0) \
96                 break; \
97             blkno += (long long)ccount; \
98         } \
99         ccount -= cw; \
100         while (cw--) \
101             lc = (lc << 7) | ((*p++) & 0177); \
102     } \
103     /* \
104      * The same() macro and letter() function were inserted to allow for \
105      * the -i option work for the multi-byte environment. \
106      */ \
107     wchar_t letter(); \
108 #define same(a, b) \
109     (a == b || iflag && (!MULTI_BYTE || ISASCII(a)) && (a ^ b) == ' ' && \
110     letter(a) == letter(b)) \
111 letter(a) == letter(b)

112 #define STDIN_FILENO gettext("(standard input)") \
113 #endif /* ! codereview */

114 #define QSIZE 400
115 struct words {
116     wchar_t inp;
117     char out;
118     struct words *nst;
119     struct words *link;
120     struct words *fail;
121 } *w = NULL, *smax, *q;
```

2

```

127 int      Hflag, hflag, iflag;
110 int     bflag, cflag, lflag, fflag, nflag, vflag, xflag, eflag, sflag;
111 int     hflag, iflag;
128 int     retcode = 0;
129 int     nfile;
130 long long blkno;
131 int     nsucc;
132 long long tln;
133 FILE    *wordf;
134 char    *argptr;
135 off_t   input_size = 0;

137 void    execute(char *);
138 void    cgotofn(void);
139 void    overflo(void);
140 void    cfail(void);

142 static long fw_lBufsiz = 0;

144 int
145 main(int argc, char **argv)
146 {
147     int c;
148     int errflg = 0;
149     struct stat file_stat;

151     (void) setlocale(LC_ALL, "");
152 #if !defined(TEXT_DOMAIN) /* Should be defined by cc -D */
153 #define TEXT_DOMAIN "SYS_TEST" /* Use this only if it weren't */
154 #endif
155     (void) textdomain(TEXT_DOMAIN);

157     while ((c = getopt(argc, argv, "Hybcie:f:lnvxqs")) != EOF)
158     while ((c = getopt(argc, argv, "hybcie:f:lnvxs")) != EOF)
159         switch (c) {

160             case 'q':
161             case 's': /* Solaris: legacy option */
162                 qflag++;
163                 continue;
164             case 'H':
165                 Hflag++;
166                 hflag = 0;
167             case 's':
168                 sflag++;
169                 continue;
170             case 'h':
171 #endif /* ! codereview */
172                 hflag++;
173                 Hflag = 0;
174             case 'b':
175                 bflag++;
176                 continue;
177             case 'i':
178             case 'y':
179                 iflag++;
180                 continue;
181
182             case 'c':
183                 cflag++;
184                 continue;
185
186             case 'e':
187                 eflag++;

```

```

188             argptr = optarg;
189             input_size = strlen(argptr);
190             continue;

192         case 'f':
193             fflag++;
194             wordf = fopen(optarg, "r");
195             if (wordf == NULL) {
196                 (void) fprintf(stderr,
197                               gettext("fgrep: can't open %s\n"),
198                               optarg);
199                 exit(2);
200             }
202             if (fstat(fileno(wordf), &file_stat) == 0) {
203                 input_size = file_stat.st_size;
204             } else {
205                 (void) fprintf(stderr,
206                               gettext("fgrep: can't fstat %s\n"),
207                               optarg);
208                 exit(2);
209             }
211             continue;

213         case 'l':
214             lflag++;
215             continue;

217         case 'n':
218             nflag++;
219             continue;

221         case 'v':
222             vflag++;
223             continue;

225         case 'x':
226             xflag++;
227             continue;

229         case '?':
230             errflg++;
231         }

233     argc -= optind;
234     if (errflg || ((argc <= 0) && !fflag && !eflag)) {
235         (void) printf(gettext("usage: fgrep [ -bcHilnqsvx ] "
236                               "[ -e exp ] [ -f file ] [ strings ] [ file ] ...\\n"));
237         exit(2);
238     }
239     if (!eflag && !fflag) {
240         argptr = argv[optind];
241         input_size = strlen(argptr);
242         input_size++;
243         optind++;
244         argc--;
245     }

247     /*
248     * Normally we need one struct words for each letter in the pattern
249     * plus one terminating struct words with outp = 1, but when -x option
250     * is specified we require one more struct words for '\n' character so we
251     * calculate the input_size as below. We add extra 1 because
252     * (input_size/2) rounds off odd numbers

```

new/usr/src/cmd/fgrep/fgrep.c

```

253 * /
254
255     if (xflag) {
256         input_size = input_size + (input_size/2) + 1;
257     }
258
259     input_size++;
260
261     w = (struct words *)calloc(input_size, sizeof (struct words));
262     if (w == NULL) {
263         (void) fprintf(stderr,
264                         gettext("fgrep: could not allocate "
265                                 "memory for wordlist\n"));
266         exit(2);
267     }
268
269     getwidth(&WW);
270     if ((WIDTH1 == 0) && (WIDTH2 == 0) &&
271         (WIDTH3 == 0)) {
272         /*
273          * If non EUC-based locale,
274          * assume WIDTH1 is 1.
275          */
276         WIDTH1 = 1;
277     }
278     WIDTH2++;
279     WIDTH3++;
280
281     cgotofn();
282     cfail();
283     nfile = argc;
284     argv = &argv[optind];
285     if (argc <= 0) {
286         execute((char *)NULL);
287     } else
288         while (--argc >= 0) {
289             execute(*argv);
290             argv++;
291         }
292
293     if (w != NULL) {
294         free(w);
295     }
296
297     return (retcode != 0 ? retcode : nsucc == 0);
298 }
299
300 void
301 execute(char *file)
302 {
303     char *p;
304     struct words *c;
305     int ccount;
306     static char *buf = NULL;
307     int failed;
308     char *nlp;
309     wchar_t lc;
310     int cw;
311
312     if (buf == NULL) {
313         fw_lBufsiz = BUFSIZ;
314         if ((buf = malloc(fw_lBufsiz + BUFSIZ)) == NULL) {
315             exit(2); /* out of memory */
316         }
317     }

```

5

```
new/usr/src/cmd/fgrep/fgrep.c
319         if (file) {
320             if ((fptr = fopen(file, "r")) == NULL) {
321                 (void) fprintf(stderr,
322                               gettext("fgrep: can't open %s\n"), file);
323                 retcode = 2;
324                 return;
325             }
326         } else {
327             file = "<stdin>";
328             fptr = stdin;
329             file = STDIN_FILENO;
330 #endif /* ! codereview */
331         }
332         ccount = 0;
333         failed = 0;
334         lnum = 1;
335         tln = 0;
336         blkno = 0;
337         p = buf;
338         nlp = p;
339         c = w;
340         for (;;) {
341             if (c == 0)
342                 break;
343             if (ccount <= 0) {
344                 if (p >= &buf[fw_lBufsiz + BUFSIZ]) {
345                     if (nlp == buf) {
346                         /* increase the buffer size */
347                         fw_lBufsiz += BUFSIZ;
348                         if ((buf = realloc(buf,
349                                         fw_lBufsiz + BUFSIZ)) == NULL) {
350                             exit(2); /* out of memory */
351                         }
352                         nlp = buf;
353                         p = &buf[fw_lBufsiz];
354                     } else {
355                         /* shift the buffer down */
356                         (void) memmove(buf, nlp,
357                                       &buf[fw_lBufsiz + BUFSIZ]
358                                       - nlp);
359                         p -= nlp - buf;
360                         nlp = buf;
361                     }
362                 }
363                 if (p > &buf[fw_lBufsiz]) {
364                     if ((ccount = fread(p, sizeof (char),
365                                         &buf[fw_lBufsiz + BUFSIZ] - p, fptr))
366                         <= 0)
367                         break;
368                     } else if ((ccount = fread(p, sizeof (char),
369                                         BUFSIZ, fptr)) <= 0)
370                         break;
371                     blkno += (long long) ccount;
372                 }
373                 GETONE(lc, p);
374             nstate:
375                 if (same(c->inp, lc)) {
376                     c = c->nst;
377                 } else if (c->link != 0) {
378                     c = c->link;
379                     goto nstate;
380                 } else {
381                     c = c->fail;
382                     failed = 1;
383                     if (c == 0) {
384                         if (failed)
385                             exit(1);
386                     }
387                 }
388             }
389         }
390     }
391     if (fptr != stdin)
392         fclose(fptr);
393     exit(0);
394 }
```

```

384         c = w;
385     istate:
386         if (same(c->inp, lc)) {
387             c = c->nst;
388         } else if (c->link != 0) {
389             c = c->link;
390             goto istate;
391         }
392         } else
393             goto nstate;
394     }
395
396     if (c == 0)
397         break;
398
399     if (c->out) {
400         while (lc != '\n') {
401             if (ccount <= 0) {
402                 if (p == &buf[fw_lBufsiz + BUFSIZ]) {
403                     if (nlp == buf) {
404                         /* increase buffer size */
405                         fw_lBufsiz += BUFSIZ;
406                         if ((buf = realloc(buf, fw_lBufsiz + BUFSIZ)) == NULL) {
407                             exit(2); /* out of memory */
408                         }
409                         nlp = buf;
410                         p = &buf[fw_lBufsiz];
411                     } else {
412                         /* shift buffer down */
413                         (void) memmove(buf, nlp, &buf[fw_lBufsiz + BUFSIZ] - nlp);
414                         p -= nlp - buf;
415                         nlp = buf;
416                     }
417                 }
418                 if (p > &buf[fw_lBufsiz]) {
419                     if ((ccount = fread(p, sizeof(char),
420                         &buf[fw_lBufsiz + BUFSIZ] - p, fptr)) <= 0) break;
421                 } else if ((ccount = fread(p, sizeof(char), BUFSIZ,
422                     fptr)) <= 0) break;
423                 blkno += (long long)ccount;
424             }
425             GETONE(lc, p);
426         }
427         if ((vflag && (failed == 0 || xflag == 0)) ||
428             (vflag == 0 && xflag && failed))
429             goto nomatch;
430     succeed:
431         nsucc = 1;
432         if (iflag || qflag) {
433             if (!qflag)
434                 if (cflag)
435                     tln++;
436             else if (lflag && !sflag) {
437                 (void) printf("%s\n", file);
438                 (void) fclose(fptr);
439                 return;
440             }
441             if (cflag) {
442                 tln++;
443             } else {
444                 if (Hflag || (nfile > 1 && !hflag))
445                     if (!sflag) {
446                         if (nfile > 1 && !hflag)
447                             (void) printf("%s:", file);
448                         if (bflag)
449                             (void) printf("%lld:",

```

```

445                                         (blkno - (long long)(ccount-1))
446                                         / BUFSIZ);
447         if (nflag)
448             (void) printf("%lld:", lnum);
449         if (p <= nlp) {
450             while (nlp < &buf[fw_lBufsiz + BUFSIZ])
451                 (void) putchar(*nlp++);
452             nlp = buf;
453         }
454         while (nlp < p)
455             (void) putchar(*nlp++);
456     }
457     nomatch:
458         lnum++;
459         nlp = p;
460         c = w;
461         failed = 0;
462         continue;
463     }
464     if (lc == '\n')
465         if (vflag)
466             goto succeed;
467         else {
468             lnum++;
469             nlp = p;
470             c = w;
471             failed = 0;
472         }
473     }
474     (void) fclose(fptr);
475     if (cflag && !qflag) {
476         if (Hflag || (nfile > 1 && !hflag))
477             if ((nfile > 1) && !hflag)
478                 (void) printf("%s:", file);
479             (void) printf("%lld\n", tln);
480     }

```

unchanged_portion_omitted

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

```
*****
10567 Thu May 30 19:29:52 2013
new/usr/src/cmd/grep/grep.c
3737 grep does not support -H 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 /*
28 * Copyright (c) 1984, 1986, 1987, 1988, 1989 AT&T */
29 * All Rights Reserved */  
30 /*
31 * Copyright (c) 1987, 1988 Microsoft Corporation */
32 * All Rights Reserved */  
33 /* Copyright 2012 Nexenta Systems, Inc. All rights reserved. */  
35 /*
36 * Copyright 2013 Damian Bogel. All rights reserved.
37 */  
39 /*
40 #endif /* ! codereview */
41 * grep -- print lines matching (or not matching) a pattern
42 *
43 *      status returns:
44 *          0 - ok, and some matches
45 *          1 - ok, but no matches
46 *          2 - some error
47 */  
49 #include <sys/types.h>  
51 #include <ctype.h>
52 #include <fcntl.h>
53 #include <locale.h>
54 #include <memory.h>
55 #include <regexp.h>
56 #include <stdio.h>
57 #include <stdlib.h>
58 #include <string.h>
59 #include <unistd.h>
60 #include <ftw.h>
61 #include <limits.h>
```

1

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

```
62 #include <sys/param.h>  
64 static const char *errstr[] = {
65     "Range endpoint too large.",
66     "Bad number.",
67     "'\\digit' out of range.",
68     "No remembered search string.",
69     "\\( \\) imbalance.",
70     "Too many \\(.",
71     "More than 2 numbers given in \\{ \\}.",
72     "} expected after \\.",
73     "First number exceeds second in \\{ \\}.",
74     "[ ] imbalance.",
75     "Regular expression overflow.",
76     "Illegal byte sequence.",
77     "Unknown regexp error code!!",
78     NULL
79 };  
81 #define STDIN_FILENO gettext("(standard input)")  
83 #endif /* ! codereview */
84 #define errmsg(msg, arg) (void) fprintf(stderr, gettext(msg), arg)
85 #define BLKSIZE 512
86 #define GBUFFSIZ 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 #endif /* ! codereview */
107 static int qflag;
108 static int errflg;
109 static int nfile;
110 static long long tln;
111 static int nsucc;
112 static int outfn = 0;
113 static int nlflag;
114 static char *ptr, *ptrend;
115 static char *expbuf;  
117 static void execute(const char *, int);
118 static void regerr(int);
119 static void prepare(const char *);
120 static int recursive(const char *, const struct stat *, int, struct FTW *);
121 static int succeed(const char *);  
123 int
124 main(int argc, char **argv)
125 {
126     int c;
127     char *arg;
```

2

```

128     extern int      optind;
129
130     (void) setlocale(LC_ALL, "");
131 #if !defined(TEXT_DOMAIN) /* Should be defined by cc -D */
132 #define TEXT_DOMAIN "SYS_TEST" /* Use this only if it weren't */
133 #endif
134     (void) textdomain(TEXT_DOMAIN);
135
136     while ((c = getopt(argv, argv, "hHqblcnRrsviyw")) != -1)
137         while ((c = getopt(argv, argv, "hgbclcnRrsviyw")) != -1)
138             switch (c) {
139 #endif /* ! codereview */
140             case 'h':
141                 hflag++;
142                 Hflag = 0; /* h excludes H */
143                 break;
144             case 'H':
145                 if (!lflag) /* H is excluded by l */
146                     Hflag++;
147                 hflag = 0; /* H excludes h */
148 #endif /* ! codereview */
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 'R':
163                 Rflag++;
164                 /* FALLTHROUGH */
165             case 'r':
166                 rflag++;
167                 break;
168             case 'b':
169                 bflag++;
170                 break;
171             case 's':
172                 sflag++;
173                 break;
174             case 'l':
175                 lflag++;
176                 Hflag = 0; /* l excludes H */
177 #endif /* ! codereview */
178                 break;
179             case 'y':
180             case 'i':
181                 iflag++;
182                 break;
183             case 'w':
184                 wflag++;
185                 break;
186             case '?':
187                 errflg++;
188             }
189
190     if (errflg || (optind >= argc)) {
191         errmsg("Usage: grep [-c|-l|-q] [-r|-R] -hHbnsviw "
192             "errmsg("Usage: grep [-c|-l|-q] [-r|-R] -hbnsiviw "
193             "errmsg("Usage: grep [-c|-l|-q] [-r|-R] -hbnsiviw "

```

```

192                                         "pattern file . . .\n",
193                                         (char *)NULL);
194                                         exit(2);
195 }
196
197     argv = &argv[optind];
198     argc -= optind;
199     nfile = argc - 1;
200
201     if (strchr(*argv, '\n') != NULL)
202         regerr(41);
203
204     if (iflag) {
205         for (arg = *argv; *arg != NULL; ++arg)
206             *arg = (char)tolower((int)((unsigned char)*arg));
207     }
208
209     if (wflag) {
210         unsigned int wordlen;
211         char        *wordbuf;
212
213         wordlen = strlen(*argv) + 5; /* '\\' '<' *argv '\\' '>' '\0' */
214         if ((wordbuf = malloc(wordlen)) == NULL) {
215             errmsg("grep: Out of memory for word\n", (char *)NULL);
216             exit(2);
217         }
218
219         (void) strcpy(wordbuf, "\\\<");
220         (void) strcat(wordbuf, *argv);
221         (void) strcat(wordbuf, "\\\>");
222         *argv = wordbuf;
223     }
224
225     expbuf = compile(*argv, (char *)0, (char *)0);
226     if (regerrno)
227         regerr(regerrno);
228
229     if (--argc == 0)
230         execute(NULL, 0);
231     else
232         while (argc-- > 0)
233             prepare(*++argv);
234
235     return (nsucc == 2 ? 2 : (nsucc == 0 ? 1 : 0));
236 }


---


237 unchanged_portion_omitted
238
239 static void
240 execute(const char *file, int base)
241 {
242     char    *lbuf, *p;
243     long    count;
244     long    offset = 0;
245     char    *next_ptr = NULL;
246     long    next_count = 0;
247
248     tln = 0;
249
250     if (prntbuf == NULL) {
251         fw_lPrntBufLen = GBUFSIZ + 1;
252         if ((prntbuf = malloc(fw_lPrntBufLen)) == NULL) {
253             exit(2); /* out of memory - BAIL */
254         }
255         if ((linebuf = malloc(fw_lPrntBufLen)) == NULL) {
256             exit(2); /* out of memory - BAIL */
257         }
258     }

```

```

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

```

```

384             }
385             p = prntbuf + offset;
386             if ((count = read(temp, p, GBUFSIZ)) > 0)
387                 continue;
388
389             if (offset == 0)
390                 /* end of file already reached */
391                 break;
392
393             /* last line of file has no newline */
394             ptrend = ptr + offset;
395             nlflag = 0;
396         } else {
397             next_ptr = ptrend + 1;
398             next_count = offset + count - (next_ptr - ptr);
399             nlflag = 1;
400         }
401         lnum++;
402         *ptr = '\0';
403
404         if (iflag) {
405             /*
406             * Make a lower case copy of the record
407             */
408             p = ptr;
409             for (lbuf = linebuf; p < ptrend; )
410                 *lbuf++ = (char)tolower((int)
411                               (unsigned char)*p++);
412             *lbuf = '\0';
413             lbuf = linebuf;
414         } else
415             /*
416             * Use record as is
417             */
418             lbuf = ptr;
419
420         /* lflag only once */
421         if ((step(lbuf, expbuf) ^ vflag) && succeed(file) == 1)
422             break;
423
424         if (!nlflag)
425             break;
426
427         ptr = next_ptr;
428         count = next_count;
429         offset = 0;
430     }
431     (void) close(temp);
432
433     if (cflag && !qflag) {
434         if (Hfflag || (!hfflag && ((nfile > 1) ||
435                         (rfflag && outfn))))
436             if (!hfflag && file && (nfile > 1) ||
437                         (rfflag && outfn))
438                 (void) fprintf(stdout, "%s:", file);
439             (void) fprintf(stdout, "%lld\n", tln);
440     }
441
442     static int
443     succeed(const char *f)
444     {
445         int nchars;
446         nsucc = (nsucc == 2) ? 2 : 1;

```

```
294     if (f == NULL)
295         f = "<stdin>";
448     if (qflag) {
449         /* no need to continue */
450         return (1);
451     }
453     if (cflag) {
454         tln++;
455         return (0);
456     }
458     if (lflag) {
459         (void) fprintf(stdout, "%s\n", f);
460         return (1);
461     }
463     if (Hflag || (!hflag && (nfile > 1 || (rflag && outfn)))) {
464     if (!hflag && (nfile > 1 || (rflag && outfn))) {
465         /* print filename */
466         (void) fprintf(stdout, "%s:", f);
467
468     if (bflag)
469         /* print block number */
470         (void) fprintf(stdout, "%lld:", (offset_t)
471             ((lseek(temp, (off_t)0, SEEK_CUR) - 1) / BLKSIZE));
472
473     if (nflag)
474         /* print line number */
475         (void) fprintf(stdout, "%lld:", lnum);
476
477     if (nlflag) {
478         /* newline at end of line */
479         *ptrend = '\n';
480         nchars = ptrend - ptr + 1;
481     } else {
482         /* don't write sentinel \0 */
483         nchars = ptrend - ptr;
484     }
485
486     (void) fwrite(ptr, 1, nchars, stdout);
487     return (0);
488 }
```

unchanged portion omitted

```
*****
28372 Thu May 30 19:29:53 2013
new/usr/src/cmd/grep_xpg4/grep.c
3737 grep does not support -H 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 2004 Sun Microsystems, Inc. All rights reserved.
24 * Use is subject to license terms.
25 */

27 /*
28 * grep - pattern matching program - combined grep, egrep, and fgrep.
29 *      Based on MKS grep command, with XCU & Solaris mods.
30 */

32 /*
33 * Copyright 1985, 1992 by Mortice Kern Systems Inc. All rights reserved.
34 *
35 */

37 /* Copyright 2012 Nexenta Systems, Inc. All rights reserved. */

39 /*
40 * Copyright 2013 Damian Bogel. All rights reserved.
41 */

43 #endif /* ! codereview */
44 #include <string.h>
45 #include <stdlib.h>
46 #include <ctype.h>
47 #include <stdarg.h>
48 #include <regex.h>
49 #include <limits.h>
50 #include <sys/types.h>
51 #include <sys/stat.h>
52 #include <fcntl.h>
53 #include <stdio.h>
54 #include <locale.h>
55 #include <wchar.h>
56 #include <errno.h>
57 #include <unistd.h>
58 #include <wctype.h>
59 #include <ftw.h>
60 #include <sys/param.h>
```

```

62 #define STDIN_FILENAME gettext("(standard input)")

64 #endif /* ! codereview */
65 #define BSIZE 512 /* Size of block for -b */
66 #define BUFSIZE 8192 /* Input buffer size */
67 #define MAX_DEPTH 1000 /* how deep to recurse */

69 #define M_CSETSIZE 256 /* singlebyte chars */
70 static int bmglen; /* length of BMG pattern */
71 static char *bmgpat; /* BMG pattern */
72 static int bmgtab[M_CSETSIZE]; /* BMG delta table */

74 typedef struct _PATTERN {
75     char *pattern; /* original pattern */
76     wchar_t *wpattern; /* wide, lowercased pattern */
77     struct _PATTERN *next; /* compiled pattern */
78     regex_t re; /* regex_t re; */
79 } PATTERN;

81 static PATTERN *patterns;
82 static char errstr[128]; /* regerror string buffer */
83 static int regflags = 0; /* regcomp options */
84 static int matched = 0; /* return of the grep() */
85 static int errors = 0; /* count of errors */
86 static uchar_t fgrep = 0; /* Invoked as fgrep */
87 static uchar_t egrep = 0; /* Invoked as egrep */
88 static uchar_t nvflag = 1; /* Print matching lines */
89 static uchar_t cflag; /* Count of matches */
90 static uchar_t iflag; /* Case insensitive matching */
91 static uchar_t Hflag; /* Precede lines by file name */
92 #endif /* ! codereview */
93 static uchar_t hflag; /* Suppress printing of filename */
94 static uchar_t lflag; /* Print file names of matches */
95 static uchar_t nflag; /* Precede lines by line number */
96 static uchar_t rflag; /* Search directories recursively */
97 static uchar_t bflag; /* Precede matches by block number */
98 static uchar_t sflag; /* Suppress file error messages */
99 static uchar_t qflag; /* Suppress standard output */
100 static uchar_t wflag; /* Search for expression as a word */
101 static uchar_t xflag; /* Anchoring */
102 static uchar_t Eflag; /* Egrep or -E flag */
103 static uchar_t Fflag; /* Fgrep or -F flag */
104 static uchar_t Rflag; /* Like rflag, but follow symlinks */
105 static uchar_t outfn; /* Put out file name */
106 static char *cmdname; /* cmdname */

108 static int use_wchar, use_bmg, mblocale;
110 static size_t outbuflen, prntbuflen;
111 static char *prntbuf;
112 static wchar_t *outline;

114 static void addfile(const char *fn);
115 static void addpattern(char *s);
116 static void fixpatterns(void);
117 static void usage(void);
118 static int grep(int, const char *);
119 static void bmgcomp(char *, int);
120 static char *bmgexec(char *, char *);
121 static int recursive(const char *, const struct stat *, int, struct FTW *);
122 static void process_path(const char *);
123 static void process_file(const char *, int);

125 /*
126 * mainline for grep
127 */
```

```

128 int
129 main(int argc, char **argv)
130 {
131     char    *ap;
132     int      c;
133     int      fflag = 0;
134     int      i, n_pattern = 0, n_file = 0;
135     char    **pattern_list = NULL;
136     char    **file_list = NULL;

138     (void) setlocale(LC_ALL, "");
139 #if !defined(TEXT_DOMAIN)          /* Should be defined by cc -D */
140 #define TEXT_DOMAIN    "SYS_TEST"   /* Use this only if it weren't */
141 #endif
142     (void) textdomain(TEXT_DOMAIN);

144     /*
145      * true if this is running on the multibyte locale
146      */
147     mb.locale = (MB_CUR_MAX > 1);
148     /*
149      * Skip leading slashes
150      */
151     cmdname = argv[0];
152     if (ap = strrchr(cmdname, '/'))
153         cmdname = ap + 1;

155     ap = cmdname;
156     /*
157      * Detect egrep/fgrep via command name, map to -E and -F options.
158      */
159     if (*ap == 'e' || *ap == 'E') {
160         regflags |= REG_EXTENDED;
161         egrep++;
162     } else {
163         if (*ap == 'f' || *ap == 'F') {
164             fgrep++;
165         }
166     }

168     while ((c = getopt(argc, argv, "vwchHilnrbs:f:qxEFIR")) != EOF) {
169     while ((c = getopt(argc, argv, "vwchilnrbse:f:qxEFIR")) != EOF) {
170         switch (c) {
171             case 'v':           /* POSIX: negate matches */
172                 nvflag = 0;
173                 break;

174             case 'c':           /* POSIX: write count */
175                 cflag++;
176                 break;

178             case 'i':           /* POSIX: ignore case */
179                 iflag++;
180                 regflags |= REG_ICASE;
181                 break;

183             case 'l':           /* POSIX: Write filenames only */
184                 lflag++;
185                 break;

187             case 'n':           /* POSIX: Write line numbers */
188                 nflag++;
189                 break;

191             case 'r':           /* Solaris: search recursively */
192                 rflag++;

```

```

193                     break;
194
195     case 'b':           /* Solaris: Write file block numbers */
196         bflag++;
197         break;
198
199     case 's':           /* POSIX: No error msgs for files */
200         sflag++;
201         break;
202
203     case 'e':           /* POSIX: pattern list */
204         n_pattern++;
205         pattern_list = realloc(pattern_list,
206                                  sizeof (char *) * n_pattern);
207         if (pattern_list == NULL) {
208             (void) fprintf(stderr,
209                           gettext("%s: out of memory\n"),
210                           cmdname);
211             exit(2);
212         }
213         *(pattern_list + n_pattern - 1) = optarg;
214         break;
215
216     case 'f':           /* POSIX: pattern file */
217         fflag = 1;
218         n_file++;
219         file_list = realloc(file_list,
220                               sizeof (char *) * n_file);
221         if (file_list == NULL) {
222             (void) fprintf(stderr,
223                           gettext("%s: out of memory\n"),
224                           cmdname);
225             exit(2);
226         }
227         *(file_list + n_file - 1) = optarg;
228         break;
229
230     /* based on options order h or H is set as in GNU grep */
231 #endif /* ! codereview */
232     case 'h':           /* Solaris: suppress printing of file name */
233         hflag = 1;
234         Hflag = 0;
235         break;
236     /* Solaris: precede every matching with file name */
237     case 'H':
238         Hflag = 1;
239         hflag = 0;
240 #endif /* ! codereview */
241         break;
242
243     case 'q':           /* POSIX: quiet: status only */
244         qflag++;
245         break;
246
247     case 'w':           /* Solaris: treat pattern as word */
248         wflag++;
249         break;
250
251     case 'x':           /* POSIX: full line matches */
252         xflag++;
253         regflags |= REG_ANCHOR;
254         break;
255
256     case 'E':           /* POSIX: Extended RE's */
257         regflags |= REG_EXTENDED;
258         Eflag++;

```

[new/usr/src/cmd/grep_xpg4/grep.c](#)

5

```

259             break;
260
261     case 'F':           /* POSIX: strings, not RE's */
262         Fflag++;
263         break;
264
265     case 'R':           /* Solaris: like rflag, but follow symlinks */
266         Rflag++;
267         rflag++;
268         break;
269
270     default:
271         usage();
272     }
273 }
274 /*
275 * If we're invoked as egrep or fgrep we need to do some checks
276 */
277
278 if (egrep || fgrep) {
279     /*
280      * Use of -E or -F with egrep or fgrep is illegal
281      */
282     if (Eflag || Fflag)
283         usage();
284
285     /*
286      * Don't allow use of wflag with egrep / fgrep
287      */
288     if (wflag)
289         usage();
290
291     /*
292      * For Solaris the -s flag is equivalent to XCU -q
293      */
294     if (sflag)
295         qflag++;
296
297     /*
298      * done with above checks - set the appropriate flags
299      */
300     if (egrep)
301         Eflag++;
302     else
303         Fflag++; /* Else fgrep */
304
305     if (wflag && (Eflag || Fflag)) {
306         /*
307          * -w cannot be specified with grep -F
308          */
309         usage();
310     }
311
312     /*
313      * -E and -F flags are mutually exclusive - check for this
314      */
315     if (Eflag && Fflag)
316         usage();
317
318     /*
319      * -l overrides -H like in GNU grep
320      */
321     if (lflag)
322         Hflag = 0;
323
324     /*
325      * ! codereview */
326      * -c, -l and -q flags are mutually exclusive

```

new/usr/src/cmd/grep_xpg4/grep.c

```

325      * We have -c override -l like in Solaris.
326      * -q overrides -l & -c programmatically in grep() function.
327      */
328      if (cflag && lflag)
329          lflag = 0;
330
331      argv += optind - 1;
332      argc -= optind - 1;
333
334      /*
335      * Now handling -e and -f option
336      */
337      if (pattern_list) {
338          for (i = 0; i < n_pattern; i++) {
339              addpattern(pattern_list[i]);
340          }
341          free(pattern_list);
342      }
343      if (file_list) {
344          for (i = 0; i < n_file; i++) {
345              addfile(file_list[i]);
346          }
347          free(file_list);
348      }
349
350      /*
351      * No -e or -f? Make sure there is one more arg, use it as the pattern.
352      */
353      if (patterns == NULL && !fflag) {
354          if (argc < 2)
355              usage();
356          addpattern(argv[1]);
357          argc--;
358          argv++;
359      }
360
361      /*
362      * If -x flag is not specified or -i flag is specified
363      * with fgrep in a multibyte locale, need to use
364      * the wide character APIs. Otherwise, byte-oriented
365      * process will be done.
366      */
367      use_wchar = Fflag && mblocale && (!xflag || iflag);
368
369      /*
370      * Compile Patterns and also decide if BMG can be used
371      */
372      fixpatterns();
373
374      /* Process all files: stdin, or rest of arg list */
375      if (argc < 2) {
376          matched = grep(0, STDIN_FILENAME);
377          matched = grep(0, gettext("(standard input)"));
378      } else {
379          if (Hflag || (argc > 2 && hflag == 0))
380              if (argc > 2 && hflag == 0)
381                  outfn = 1; /* Print filename on match line */
382          for (argv++; *argv != NULL; argv++) {
383              process_path(*argv);
384          }
385      }
386      /*
387      * Return() here is used instead of exit
388      */
389      (void) fflush(stdout);

```

```

390     if (errors)
391         return (2);
392     return (matched ? 0 : 1);
393 }


---


unchanged_portion_omitted_

794 /*
795 * Do grep on a single file.
796 * Return true in any lines matched.
797 *
798 * We have two strategies:
799 * The fast one is used when we have a single pattern with
800 * a string known to occur in the pattern. We can then
801 * do a BMG match on the whole buffer.
802 * This is an order of magnitude faster.
803 * Otherwise we split the buffer into lines,
804 * and check for a match on each line.
805 */
806 static int
807 grep(int fd, const char *fn)
808 {
809     PATTERN *pp;
810     off_t data_len;      /* length of the data chunk */
811     off_t line_len;     /* length of the current line */
812     off_t line_offset;  /* current line's offset from the beginning */
813     long long lineno;
814     long long matches = 0; /* Number of matching lines */
815     int newlinep;        /* 0 if the last line of file has no newline */
816     char *ptr, *ptrend;
817
818     if (patterns == NULL)
819         return (0); /* no patterns to match -- just return */
820
821     pp = patterns;
822
823     if (use_bmg) {
824         bmgcomp(pp->pattern, strlen(pp->pattern));
825     }
826
827     if (use_wchar && outline == NULL) {
828         outbuflen = BUFSIZE + 1;
829         outline = malloc(sizeof(wchar_t) * outbuflen);
830         if (outline == NULL) {
831             (void) fprintf(stderr, gettext("%s: out of memory\n"),
832                           cmdname);
833             exit(2);
834         }
835     }
836
837     if (prntbuf == NULL) {
838         prntbuflen = BUFSIZE;
839         if ((prntbuf = malloc(prntbuflen + 1)) == NULL) {
840             (void) fprintf(stderr, gettext("%s: out of memory\n"),
841                           cmdname);
842             exit(2);
843         }
844     }
845
846     line_offset = 0;
847     lineno = 0;
848     newlinep = 1;
849     data_len = 0;
850     for ( ; ; ) {
851         long count;
852

```

```

853         off_t offset = 0;
854
855         if (data_len == 0) {
856             /* If no data in the buffer, reset ptr
857             */
858             ptr = prntbuf;
859         }
860         if (ptr == prntbuf) {
861             /* The current data chunk starts from prntbuf.
862             * This means either the buffer has no data
863             * or the buffer has no newline.
864             * So, read more data from input.
865             */
866             count = read(fd, ptr + data_len, prntbuflen - data_len);
867             if (count < 0) {
868                 /* read error */
869                 if (cflag) {
870                     if (outfn && !rflag) {
871                         (void) fprintf(stdout,
872                           "%s:", fn);
873                     }
874                 }
875                 if (!qflag && !rflag) {
876                     (void) fprintf(stdout, "%lld\n",
877                                   matches);
878                 }
879             }
880             return (0);
881         } else if (count == 0) {
882             /* no new data */
883             if (data_len == 0) {
884                 /* end of file already reached */
885                 break;
886             }
887             /* last line of file has no newline */
888             ptrend = ptr + data_len;
889             newlinep = 0;
890             goto L_start_process;
891         }
892         offset = data_len;
893         data_len += count;
894     }
895
896     /*
897     * Look for newline in the chunk
898     * between ptr + offset and ptr + data_len - offset.
899     */
900     ptrend = find_nl(ptr + offset, data_len - offset);
901     if (ptrend == NULL) {
902         /* no newline found in this chunk */
903         if (ptr > prntbuf) {
904             /* Move remaining data to the beginning
905             * of the buffer.
906             * Remaining data lie from ptr for
907             * data_len bytes.
908             */
909             (void) memmove(prntbuf, ptr, data_len);
910         }
911         if (data_len == prntbuflen) {
912             /* No enough room in the buffer
913             */
914             prntbuflen += BUFSIZE;
915             prntbuf = realloc(prntbuf, prntbuflen + 1);
916
917
918

```

```

919         if (prntbuf == NULL) {
920             (void) fprintf(stderr,
921                         gettext("%s: out of memory\n"),
922                         cmdname);
923             exit(2);
924         }
925     }
926     /* read the next input */
927     continue;
928 }
929 L_start_process:
930
/* Beginning of the chunk:      ptr
 * End of the chunk:          ptr + data_len
 * Beginning of the line:      ptr
 * End of the line:           ptrend
 */
932
933 if (use_bmgi) {
934     /*
935      * Use Boyer-Moore-Gosper algorithm to find out if
936      * this chunk (not this line) contains the specified
937      * pattern. If not, restart from the last line
938      * of this chunk.
939     */
940     char *bline;
941     bline = bmgeexec(ptr, ptr + data_len);
942     if (bline == NULL) {
943         /*
944          * No pattern found in this chunk.
945          * Need to find the last line
946          * in this chunk.
947        */
948     ptrend = rfind_nl(ptr, data_len);
949
950     /*
951      * When this chunk does not contain newline,
952      * ptrend becomes NULL, which should happen
953      * when the last line of file does not end
954      * with a newline. At such a point,
955      * newlinep should have been set to 0.
956      * Therefore, just after jumping to
957      * L_skip_line, the main for-loop quits,
958      * and the line_len value won't be
959      * used.
960     */
961     line_len = ptrend - ptr;
962     goto L_skip_line;
963 }
964 if (bline > ptrend) {
965     /*
966      * Pattern found not in the first line
967      * of this chunk.
968      * Discard the first line.
969    */
970     line_len = ptrend - ptr;
971     goto L_skip_line;
972 }
973 /*
974  * Pattern found in the first line of this chunk.
975  * Using this result.
976 */
977 *ptrend = '\0';
978 line_len = ptrend - ptr;
979 */
980
981
982
983
984

```

```

986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
/*
 * before jumping to L_next_line,
 * need to handle xflag if specified
 */
if (xflag && (line_len != bmglen ||
               strcmp(bmglpat, ptr) != 0)) {
    /* didn't match */
    pp = NULL;
} else {
    pp = patterns; /* to make it happen */
}
goto L_next_line;
lineno++;
/*
 * Line starts from ptr and ends at ptrend.
 * line_len will be the length of the line.
 */
*ptrend = '\0';
line_len = ptrend - ptr;
/*
 * From now, the process will be performed based
 * on the line from ptr to ptrend.
 */
if (use_wchar) {
    size_t len;
    if (line_len >= outbuflen) {
        outbuflen = line_len + 1;
        outline = realloc(outline,
                           sizeof (wchar_t) * outbuflen);
        if (outline == NULL) {
            (void) fprintf(stderr,
                          gettext("%s: out of memory\n"),
                          cmdname);
            exit(2);
        }
        len = mbstowcs(outline, ptr, line_len);
        if (len == (size_t)-1) {
            (void) fprintf(stderr, gettext(
                "%s: input file \"%s\": line %lld: invalid multibyte character\n"),
                cmdname, fn, lineno);
            /* never match a line with invalid sequence */
            goto L_skip_line;
        }
        outline[len] = L'\0';
    }
    if (iflag) {
        wchar_t *cp;
        for (cp = outline; *cp != '\0'; cp++) {
            *cp = towlower((wint_t)*cp);
        }
    }
    if (xflag) {
        for (pp = patterns; pp; pp = pp->next) {
            if (outline[0] == pp->wpattern[0] &&
                wcscmp(outline,
                       pp->wpattern) == 0) {
                /* matched */
                break;
            }
        }
    }
}

```

```

1051         }
1052     } else {
1053         for (pp = patterns; pp; pp = pp->next) {
1054             if (wcswcs(outline, pp->wpattern)
1055                 != NULL) {
1056                 /* matched */
1057                 break;
1058             }
1059         }
1060     } else if (Fflag) {
1061         /* fgrep in byte-oriented handling */
1062         char *fptr;
1063         if (iflag) {
1064             fptr = istrdup(ptr);
1065         } else {
1066             fptr = ptr;
1067         }
1068         if (xflag) {
1069             /* fgrep -x */
1070             for (pp = patterns; pp; pp = pp->next) {
1071                 if (fptr[0] == pp->pattern[0] &&
1072                     strcmp(fptr, pp->pattern) == 0) {
1073                     /* matched */
1074                     break;
1075                 }
1076             }
1077         } else {
1078             for (pp = patterns; pp; pp = pp->next) {
1079                 if (strstr(fptr, pp->pattern) != NULL) {
1080                     /* matched */
1081                     break;
1082                 }
1083             }
1084         }
1085     } else {
1086         /* grep or egrep */
1087         for (pp = patterns; pp; pp = pp->next) {
1088             int rv;
1089
1090             rv = regexec(&pp->re, ptr, 0, NULL, 0);
1091             if (rv == REG_OK) {
1092                 /* matched */
1093                 break;
1094             }
1095
1096             switch (rv) {
1097             case REG_NOMATCH:
1098                 break;
1099             case REG_ECHAR:
1100                 (void) fprintf(stderr, gettext(
1101                     "%s: input file \"%s\": line %lld: invalid multibyte character\n"),
1102                     cmdname, fn, lineno);
1103                 break;
1104             default:
1105                 (void) regerror(rv, &pp->re, errstr,
1106                                 sizeof (errstr));
1107                 (void) fprintf(stderr, gettext(
1108                     "%s: input file \"%s\": line %lld: %s\n"),
1109                     cmdname, fn, lineno, errstr);
1110                 exit(2);
1111             }
1112         }
1113     }
1114
1115 L_next_line:

```

```

1116
1117         /*
1118          * Here, if pp points to non-NULL, something has been matched
1119          * to the pattern.
1120          */
1121         if (nvflag == (pp != NULL)) {
1122             matches++;
1123             /*
1124              * Handle q, l, and c flags.
1125              */
1126             if (qflag) {
1127                 /* no need to continue */
1128                 /*
1129                  * End of this line is ptrend.
1130                  * We have read up to ptr + data_len.
1131                  */
1132                 off_t pos;
1133                 pos = ptr + data_len - (ptrend + 1);
1134                 (void) lseek(fd, -pos, SEEK_CUR);
1135                 exit(0);
1136             }
1137             if (lflag) {
1138                 (void) printf("%s\n", fn);
1139                 break;
1140             }
1141             if (!cflag) {
1142                 if (Hflag || outfn) {
1143                     if (outfn)
1144                         (void) printf("%s:", fn);
1145                     if (bfflag) {
1146                         (void) printf("%lld:", (offset_t)
1147                                     (line_offset / BSIZE));
1148                     }
1149                     if (nflag) {
1150                         (void) printf("%lld:", lineno);
1151                     }
1152                     *ptrend = '\n';
1153                     (void) fwrite(ptr, 1, line_len + 1, stdout);
1154                 }
1155                 if (ferror(stdout)) {
1156                     return (0);
1157                 }
1158             L_skip_line:
1159             if (!newlinep)
1160                 break;
1161
1162             data_len -= line_len + 1;
1163             line_offset += line_len + 1;
1164             ptr = ptrend + 1;
1165
1166         }
1167         if (cflag) {
1168             if (Hflag || outfn) {
1169                 if (outfn)
1170                     (void) printf("%s:", fn);
1171                 if (!qflag) {
1172                     (void) printf("%lld\n", matches);
1173                 }
1174             }
1175         }
1176     }
1177 }
1178
1179 */
1180 /* usage message for grep

```

```
1181  */
1182 static void
1183 usage(void)
1184 {
1185     if (egrep || fgrep) {
1186         (void) fprintf(stderr, gettext("Usage:\t%s", cmdname);
1187         (void) fprintf(stderr,
1188             gettext(" [-c|-l|-q] [-r|-R] [-bhHinsvx] "
1189             gettext(" [-c|-l|-q] [-r|-R] [-bhHinsvx] "
1190             "pattern_list [file ...]\n"));
1191
1192         (void) fprintf(stderr, "\t%s", cmdname);
1193         (void) fprintf(stderr,
1194             gettext(" [-c|-l|-q] [-r|-R] [-bhHinsvx] "
1195             gettext(" [-c|-l|-q] [-r|-R] [-bhHinsvx] "
1196             "[ -e pattern_list]... "
1197             "[ -f pattern_file]... [file...]\n"));
1198     } else {
1199         (void) fprintf(stderr, gettext("Usage:\t%s", cmdname);
1200         (void) fprintf(stderr,
1201             gettext(" [-c|-l|-q] [-r|-R] [-bhHinsvx] "
1202             gettext(" [-c|-l|-q] [-r|-R] [-bhHinsvx] "
1203             "[ -e pattern_list]... "
1204             "[ -f pattern_file]... [file...]\n"));
1205
1206         (void) fprintf(stderr, "\t%s", cmdname);
1207         (void) fprintf(stderr,
1208             gettext(" [-c|-l|-q] [-r|-R] [-bhHinsvx] "
1209             gettext(" [-c|-l|-q] [-r|-R] [-bhHinsvx] "
1210             "[ -e pattern_list]... "
1211             "[ -f pattern_file]... [file...]\n"));
1212
1213         (void) fprintf(stderr, "\t%s", cmdname);
1214         (void) fprintf(stderr,
1215             gettext(" -E [-c|-l|-q] [-r|-R] [-bhHinsvx] "
1216             gettext(" -E [-c|-l|-q] [-r|-R] [-bhHinsvx] "
1217             "[ -e pattern_list]... "
1218             "[ -f pattern_file]... [file...]\n"));
1219
1220         (void) fprintf(stderr, "\t%s", cmdname);
1221         (void) fprintf(stderr,
1222             gettext(" -F [-c|-l|-q] [-r|-R] [-bhHinsvx] "
1223             gettext(" -F [-c|-l|-q] [-r|-R] [-bhHinsvx] "
1224             "[ -e pattern_list]... "
1225             "[ -f pattern_file]... [file...]\n"));
1226
1227     }
1228 }
1229 exit(2);
1230 /* NOTREACHED */
1231 }
```

unchanged portion omitted

```

new/usr/src/man/man1/egrep.1                                         1
*****
9118 Thu May 30 19:29:53 2013
new/usr/src/man/man1/egrep.1
3737 grep does not support -H option
*****
1 '\\" te
2 '\\" Copyright 1989 AT&T
3 '\\" Copyright (c) 2006, 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 EGREP 1 "May 3, 2013"
12 .TH EGREP 1 "Mar 24, 2006"
13 .SH NAME
14 egrep \- search a file for a pattern using full regular expressions
15 .SH SYNOPSIS
16 .LP
17 .nf
18 \fB/usr/bin/egrep\fR [\fB-bcHilnqsv\fR] \fB-e\fR \fIpattern_list\fR [\fIfile...
18 \fB/usr/bin/egrep\fR [\fB-bchilnsv\fR] \fB-e\fR \fIpattern_list\fR [\fIfile...\\f
19 .fi

21 .LP
22 .nf
23 \fB/usr/bin/egrep\fR [\fB-bcHilnqsv\fR] \fB-f\fR \fIfile\fR [\fIfile...\fR]
23 \fB/usr/bin/egrep\fR [\fB-bchilnsv\fR] \fB-f\fR \fIfile\fR [\fIfile...\fR]
24 .fi

26 .LP
27 .nf
28 \fB/usr/bin/egrep\fR [\fB-bcHilnqsv\fR] \fIpattern\fR [\fIfile...\fR]
28 \fB/usr/bin/egrep\fR [\fB-bchilnsv\fR] \fIpattern\fR [\fIfile...\fR]
29 .fi

31 .LP
32 .nf
33 \fB/usr/xpg4/bin/egrep\fR [\fB-bcHilnqsvx\fR] \fB-e\fR \fIpattern_list\fR [\fB-
33 \fB/usr/xpg4/bin/egrep\fR [\fB-bchilnqsvx\fR] \fB-e\fR \fIpattern_list\fR [\fB-f
34 [\fIfile...\fR]
35 .fi

37 .LP
38 .nf
39 \fB/usr/xpg4/bin/egrep\fR [\fB-bcHilnqsvx\fR] [\fB-e\fR \fIpattern_list\fR] \fB-
39 \fB/usr/xpg4/bin/egrep\fR [\fB-bchilnqsvx\fR] [\fB-e\fR \fIpattern_list\fR] \fB-
40 [\fIfile...\fR]
41 .fi

43 .LP
44 .nf
45 \fB/usr/xpg4/bin/egrep\fR [\fB-bcHilnqsvx\fR] \fIpattern\fR [\fIfile...\fR]
45 \fB/usr/xpg4/bin/egrep\fR [\fB-bchilnqsvx\fR] \fIpattern\fR [\fIfile...\fR]
46 .fi

48 .SH DESCRIPTION
49 .sp
50 .LP
51 The \fBegrep\fR (\fBexpression grep\fR) utility searches files for a pattern of
52 characters and prints all lines that contain that pattern. \fBegrep\fR uses
53 full regular expressions (expressions that have string values that use the full
54 set of alphanumeric and special characters) to match the patterns. It uses a

```

```

new/usr/src/man/man1/egrep.1                                         2
*****
55 fast deterministic algorithm that sometimes needs exponential space.
56 .sp
57 .LP
58 If no files are specified, \fBegrep\fR assumes standard input. Normally, each
59 line found is copied to the standard output. The file name is printed before
60 each line found if there is more than one input file.
61 .SS "/usr/bin/egrep"
62 .sp
63 .LP
64 The \fB/usr/bin/egrep\fR utility accepts full regular expressions as described
65 on the \fBegrep\fR(5) manual page, except for \fB\|(\fR and \fB\b|\fR,
66 \fB\b|(\fR and \fB\b|\e)\fR, \fB\b|\e\fR and \fB\b|\e\fR, \fB\b|\e<\fR and \fB\b|\e>\fR, and
67 \fB\b|\en\fR, and with the addition of:
68 .RS +4
69 .TP
70 1.
71 A full regular expression followed by \fB+\fR that matches one or more
72 occurrences of the full regular expression.
73 .RE
74 .RS +4
75 .TP
76 2.
77 A full regular expression followed by \fB?\fR that matches 0 or 1
78 occurrences of the full regular expression.
79 .RE
80 .RS +4
81 .TP
82 3.
83 Full regular expressions separated by | or by a \fBNEWLINE\fR that match
84 strings that are matched by any of the expressions.
85 .RE
86 .RS +4
87 .TP
88 4.
89 A full regular expression that can be enclosed in parentheses \fB()\fR for
90 grouping.
91 .RE
92 .sp
93 .LP
94 Be careful using the characters \fB$\fR, \fB*\fR, \fB[\fR, \fB^|\fR, |, \fB(\fR,
95 \fB)\fR, and \fB\b|\e\fR in \fIfull regular expression\fR, because they are also
96 meaningful to the shell. It is safest to enclose the entire \fIfull regular
97 expression\fR in single quotes (\fBa'\fR\fBa'\fR).
98 .sp
99 .LP
100 The order of precedence of operators is \fB[\|]\fR, then \fB*\|\?|\+\fR, then
101 concatenation, then | and NEWLINE.
102 .SS "/usr/xpg4/bin/egrep"
103 .sp
104 .LP
105 The \fB/usr/xpg4/bin/egrep\fR utility uses the regular expressions described in
106 the \fBEXTENDED REGULAR EXPRESSIONS\fR section of the \fBegrep\fR(5) manual
107 page.
108 .SH OPTIONS
109 .sp
110 .LP
111 The following options are supported for both \fB/usr/bin/egrep\fR and
112 \fB/usr/xpg4/bin/egrep\fR:
113 .sp
114 .ne 2
115 .na
116 \fB\fB-b\fR\fR
117 .ad
118 .RS 19n
119 Precede each line by the block number on which it was found. This can be useful
120 in locating block numbers by context (first block is 0).

```

```

121 .RE
123 .sp
124 .ne 2
125 .na
126 \fB\fB-c\fR\fR
127 .ad
128 .RS 19n
129 Print only a count of the lines that contain the pattern.
130 .RE

132 .sp
133 .ne 2
134 .na
135 \fB\fB-e\fR \fIpattern_list\fR\fR
136 .ad
137 .RS 19n
138 Search for a \fIpattern_list\fR (\fIfull regular expression\fR that begins with
139 a \fB\.(mi\fR).
140 .RE

142 .sp
143 .ne 2
144 .na
145 \fB\fB-f\fR \fIfile\fR\fR
146 .ad
147 .RS 19n
148 Take the list of \fIfull\fR \fIregular\fR \fIexpressions\fR from \fIfile\fR.
149 .RE

151 .sp
152 .ne 2
153 .na
154 \fB\fB-H\fR\fR
155 .ad
156 .RS 19n
157 Precedes each line by the name of the file containing the matching line.
158 .RE

160 .sp
161 .ne 2
162 .na
163 #endif /* ! codereview */
164 \fB\fB-h\fR\fR
165 .ad
166 .RS 19n
167 Suppress printing of filenames when searching multiple files.
168 .RE

170 .sp
171 .ne 2
172 .na
173 \fB\fB-i\fR\fR
174 .ad
175 .RS 19n
176 Ignore upper/lower case distinction during comparisons.
177 .RE

179 .sp
180 .ne 2
181 .na
182 \fB\fB-l\fR\fR
183 .ad
184 .RS 19n
185 Print the names of files with matching lines once, separated by NEWLINES. Does
186 not repeat the names of files when the pattern is found more than once.

```

```

187 .RE
189 .sp
190 .ne 2
191 .na
192 \fB\fB-n\fR\fR
193 .ad
194 .RS 19n
195 Precede each line by its line number in the file (first line is 1).
196 .RE

198 .sp
199 .ne 2
200 .na
201 \fB\fB-q\fR\fR
202 .ad
203 .RS 19n
204 Quiet. Does not write anything to the standard output, regardless of matching
205 lines. Exits with zero status if an input line is selected.
206 .RE

208 .sp
209 .ne 2
210 .na
211 #endif /* ! codereview */
212 \fB\fB-s\fR\fR
213 .ad
214 .RS 19n
215 Legacy equivalent of \fB-q\fR.
216 Work silently, that is, display nothing except error messages. This is useful
217 for checking the error status.
218 .RE

220 .na
221 \fB\fB-v\fR\fR
222 .ad
223 .RS 19n
224 Print all lines except those that contain the pattern.
225 .RE

227 .SS "/usr/xpg4/bin/egrep"
228 .sp
229 .LP
230 The following options are supported for \fB/usr/xpg4/bin/egrep\fR only:
231 .sp
232 .ne 2
233 .na
234 \fB\fB-q\fR\fR
235 .ad
236 .RS 6n
237 Quiet. Does not write anything to the standard output, regardless of matching
238 lines. Exits with zero status if an input line is selected.
239 .RE

241 .sp
242 .ne 2
243 .na
244 \fB\fB-x\fR\fR
245 .ad
246 .RS 6n
247 Consider only input lines that use all characters in the line to match an
248 entire fixed string or regular expression to be matching lines.
249 .RE

```

```

241 .SH OPERANDS
242 .sp
243 .LP
244 The following operands are supported:
245 .sp
246 .ne 2
247 .na
248 \fB\fIfile\fR\fR
249 .ad
250 .RS 8n
251 A path name of a file to be searched for the patterns. If no \fIfile\fR
252 operands are specified, the standard input is used.
253 .RE

255 .SS "/usr/bin/egrep"
256 .sp
257 .ne 2
258 .na
259 \fB\fIpattern\fR\fR
260 .ad
261 .RS 1ln
262 Specify a pattern to be used during the search for input.
263 .RE

265 .SS "/usr/xpg4/bin/egrep"
266 .sp
267 .ne 2
268 .na
269 \fB\fIpattern\fR\fR
270 .ad
271 .RS 1ln
272 Specify one or more patterns to be used during the search for input. This
273 operand is treated as if it were specified as \fB-e\fR\fIpattern_list.\fR.
274 .RE

276 .SH USAGE
277 .sp
278 .LP
279 See \fBlargefile\fR(5) for the description of the behavior of \fBegrep\fR when
280 encountering files greater than or equal to 2 Gbyte ( 231 bytes).
281 .SH ENVIRONMENT VARIABLES
282 .sp
283 .LP
284 See \fBenviron\fR(5) for descriptions of the following environment variables
285 that affect the execution of \fBegrep\fR: \fBLC_COLLATE\fR, \fBLC_CTYPE\fR,
286 \fBLC_MESSAGES\fR, and \fBNLSPATH\fR.
287 .SH EXIT STATUS
288 .sp
289 .LP
290 The following exit values are returned:
291 .sp
292 .ne 2
293 .na
294 \fB\fB0\fR\fR
295 .ad
296 .RS 5n
297 If any matches are found.
298 .RE

300 .sp
301 .ne 2
302 .na
303 \fB\fB1\fR\fR
304 .ad
305 .RS 5n
306 If no matches are found.

```

```

307 .RE
309 .sp
310 .ne 2
311 .na
312 \fB\fB2\fR\fR
313 .ad
314 .RS 5n
315 For syntax errors or inaccessible files (even if matches were found).
316 .RE

318 .SH ATTRIBUTES
319 .sp
320 .LP
321 See \fBattributes\fR(5) for descriptions of the following attributes:
322 .SS "/usr/bin/egrep"
323 .sp

325 .sp
326 .TS
327 box;
328 c | c
329 l | l .
330 ATTRIBUTE TYPE ATTRIBUTE VALUE
331 -
332 CSI Not Enabled
333 .TE

335 .SS "/usr/xpg4/bin/egrep"
336 .sp

338 .sp
339 .TS
340 box;
341 c | c
342 l | l .
343 ATTRIBUTE TYPE ATTRIBUTE VALUE
344 -
345 CSI Enabled
346 .TE

348 .SH SEE ALSO
349 .sp
350 .LP
351 \fBfgrep\fR(1), \fBgrep\fR(1), \fBsed\fR(1), \fBsh\fR(1), \fBattributes\fR(5),
352 \fBenviron\fR(5), \fBlargefile\fR(5), \fBregex\fR(5), \fBregexp\fR(5),
353 \fBXPG4\fR(5)
354 .SH NOTES
355 .sp
356 .LP
357 Ideally there should be only one \fBgrep\fR command, but there is not a single
358 algorithm that spans a wide enough range of space-time trade-offs.
359 .sp
360 .LP
361 Lines are limited only by the size of the available virtual memory.
362 .SS "/usr/xpg4/bin/egrep"
363 .sp
364 .LP
365 The \fB/usr/xpg4/bin/egrep\fR utility is identical to \fB/usr/xpg4/bin/grep\fR
366 \fB-E\fR. See \fBgrep\fR(1). Portable applications should use
367 \fB/usr/xpg4/bin/grep\fR \fB-E\fR.

```

```
new/usr/src/man/man1/fgrep.1
```

1

```
*****  
7742 Thu May 30 19:29:53 2013  
new/usr/src/man/man1/fgrep.1  
3737 grep does not support -H option  
*****  
1 '\\" te  
2 '\\" Copyright 1989 AT&T  
3 '\\" Copyright (c) 2006, 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 FGREP 1 "May 3, 2013"  
12 .TH FGREP 1 "Mar 24, 2006"  
13 .SH NAME  
14 fgrep \- search a file for a fixed-character string  
15 .SH SYNOPSIS  
16 .LP  
17 .nf  
18 \fB/usr/bin/fgrep\fR [\fB-bcHilnqsvx\fR] \fB-e\fR \fIpattern_list\fR [\fIfile...  
19 .fi  
21 .LP  
22 .nf  
23 \fB/usr/bin/fgrep\fR [\fB-bcHilnqsvx\fR] \fB-f\fR \fIfile\fR [\fIfile...\fR]  
23 \fB/usr/bin/fgrep\fR [\fB-bchilnsvx\fR] \fB-f\fR \fIfile\fR [\fIfile...\fR]  
24 .fi  
26 .LP  
27 .nf  
28 \fB/usr/bin/fgrep\fR [\fB-bcHilnqsvx\fR] \fIpattern\fR [\fIfile...\fR]  
28 \fB/usr/bin/fgrep\fR [\fB-bchilnsvx\fR] \fIpattern\fR [\fIfile...\fR]  
29 .fi  
31 .LP  
32 .nf  
33 \fB/usr/xpg4/bin/fgrep\fR [\fB-bcHilnqsvx\fR] \fB-e\fR \fIpattern_list\fR [\fB-  
33 \fB/usr/xpg4/bin/fgrep\fR [\fB-bchilnqsvx\fR] \fB-e\fR \fIpattern_list\fR [\fB-f  
34 [\fIfile...\fR]  
35 .fi  
37 .LP  
38 .nf  
39 \fB/usr/xpg4/bin/fgrep\fR [\fB-bcHilnqsvx\fR] [\fB-e\fR \fIpattern_list\fR] \fB-  
39 \fB/usr/xpg4/bin/fgrep\fR [\fB-bchilnqsvx\fR] [\fB-e\fR \fIpattern_list\fR] \fB-  
40 [\fIfile...\fR]  
41 .fi  
43 .LP  
44 .nf  
45 \fB/usr/xpg4/bin/fgrep\fR [\fB-bcHilnqsvx\fR] \fIpattern\fR [\fIfile...\fR]  
45 \fB/usr/xpg4/bin/fgrep\fR [\fB-bchilnqsvx\fR] \fIpattern\fR [\fIfile...\fR]  
46 .fi  
48 .SH DESCRIPTION  
49 .sp  
50 .LP  
51 The \fBfgrep\fR (fast \fBgrep\fR) utility searches files for a character string  
52 and prints all lines that contain that string. \fBfgrep\fR is different from  
53 \fBgrep\fR(1) and from \fBgrep\fR(1) because it searches for a string, instead  
54 of searching for a pattern that matches an expression. \fBfgrep\fR uses a fast
```

```
new/usr/src/man/man1/fgrep.1
```

2

```
55 and compact algorithm.  
56 .sp  
57 .LP  
58 The characters \fB$\fR, \fB*\fR, \fB[\fR, \fB\fR, |, \fB(\fR, \fB)\fR, and  
59 \fB)e\fR are interpreted literally by \fBfgrep\fR, that is, \fBfgrep\fR does  
60 not recognize full regular expressions as does \fBgrep\fR. These characters  
61 have special meaning to the shell. Therefore, to be safe, enclose the entire  
62 \fIstring\fR within single quotes (\fBa\'\fR).  
63 .sp  
64 .LP  
65 If no files are specified, \fBfgrep\fR assumes standard input. Normally, each  
66 line that is found is copied to the standard output. The file name is printed  
67 before each line that is found if there is more than one input file.  
68 .SH OPTIONS  
69 .sp  
70 .LP  
71 The following options are supported for both \fB/usr/bin/fgrep\fR and  
72 \fB/usr/xpg4/bin/fgrep\fR:  
73 .sp  
74 .ne 2  
75 .na  
76 \fB\fB-b\fR\fR  
77 .ad  
78 .RS 19n  
79 Precedes each line by the block number on which the line was found. This can be  
80 useful in locating block numbers by context. The first block is 0.  
81 .RE  
83 .sp  
84 .ne 2  
85 .na  
86 \fB\fB-c\fR\fR  
87 .ad  
88 .RS 19n  
89 Prints only a count of the lines that contain the pattern.  
90 .RE  
92 .sp  
93 .ne 2  
94 .na  
95 \fB\fB-e\fR \fIpattern_list\fR\fR  
96 .ad  
97 .RS 19n  
98 Searches for a \fIstring\fR in \fIpattern_list\fR. This is useful when the  
99 \fIstring\fR begins with a \fB(\fR\&.  
100 .RE  
102 .sp  
103 .ne 2  
104 .na  
105 \fB\fB-f\fR \fIpattern-file\fR\fR  
106 .ad  
107 .RS 19n  
108 Takes the list of patterns from \fIpattern-file\fR.  
109 .RE  
111 .sp  
112 .ne 2  
113 .na  
114 \fB\fB-H\fR\fR  
115 .ad  
116 .RS 19n  
117 Precedes each line by the name of the file containing the matching line.  
118 .RE  
120 .sp
```

```

121 .ne 2
122 .na
123 #endif /* ! codereview */
124 \fB\fB-h\fR\fR
125 .ad
126 .RS 19n
127 Suppresses printing of files when searching multiple files.
128 .RE

130 .sp
131 .ne 2
132 .na
133 \fB\fB-i\fR\fR
134 .ad
135 .RS 19n
136 Ignores upper/lower case distinction during comparisons.
137 .RE

139 .sp
140 .ne 2
141 .na
142 \fB\fB-l\fR\fR
143 .ad
144 .RS 19n
145 Prints the names of files with matching lines once, separated by new-lines.
146 Does not repeat the names of files when the pattern is found more than once.
147 .RE

149 .sp
150 .ne 2
151 .na
152 \fB\fB-n\fR\fR
153 .ad
154 .RS 19n
155 Precedes each line by its line number in the file. The first line is 1.
156 .RE

158 .sp
159 .ne 2
160 .na
161 \fB\fB-q\fR\fR
114 \fB\fB-s\fR\fR
162 .ad
163 .RS 19n
164 Quiet. Does not write anything to the standard output, regardless of matching
165 lines. Exits with zero status if an input line is selected.
117 Works silently, that is, displays nothing except error messages. This is useful
118 for checking the error status.
166 .RE

168 .sp
169 .ne 2
170 .na
171 \fB\fB-s\fR\fR
124 \fB\fB-v\fR\fR
172 .ad
173 .RS 19n
174 Legacy equivalent of \fB-q\fR.
127 Prints all lines except those that contain the pattern.
175 .RE

177 .sp
178 .ne 2
179 .na
180 \fB\fB-v\fR\fR
133 \fB\fB-x\fR\fR

```

```

181 .ad
182 .RS 19n
183 Prints all lines except those that contain the pattern.
136 Prints only lines that are matched entirely.
184 .RE

139 .SS "/usr/xpg4/bin/fgrep"
140 .sp
141 .LP
142 The following options are supported for \fB/usr/xpg4/bin/fgrep\fR only:
186 .sp
187 .ne 2
188 .na
189 \fB\fB-x\fR\fR
146 \fB\fB-q\fR\fR
190 .ad
191 .RS 19n
192 Prints only lines that are matched entirely.
148 .RS 6n
149 Quiet. Does not write anything to the standard output, regardless of matching
150 lines. Exits with zero status if an input line is selected.
193 .RE

195 .SH OPERANDS
196 .sp
197 .LP
198 The following operands are supported:
199 .sp
200 .ne 2
201 .na
202 \fB\fIfile\fR\fR
203 .ad
204 .RS 8n
205 Specifies a path name of a file to be searched for the patterns. If no
206 \fIfile\fR operands are specified, the standard input will be used.
207 .RE

209 .SS "/usr/bin/fgrep"
210 .sp
211 .ne 2
212 .na
213 \fB\fIpattern\fR\fR
214 .ad
215 .RS 11n
216 Specifies a pattern to be used during the search for input.
217 .RE

219 .SS "/usr/xpg4/bin/fgrep"
220 .sp
221 .ne 2
222 .na
223 \fB\fIpattern\fR\fR
224 .ad
225 .RS 11n
226 Specifies one or more patterns to be used during the search for input. This
227 operand is treated as if it were specified as \fB-e\fR \fIpattern_list\fR.
228 .RE

230 .SH USAGE
231 .sp
232 .LP
233 See \fBlargefile\fR(5) for the description of the behavior of \fBfgrep\fR when
234 encountering files greater than or equal to 2 Gbyte ( 2^31 bytes).
235 .SH ENVIRONMENT VARIABLES
236 .sp
237 .LP

```

```

238 See \fBenviron\fR(5) for descriptions of the following environment variables
239 that affect the execution of \fBfgrep\fR: \fBLC_COLLATE\fR, \fBLC_CTYPE\fR,
240 \fBLC_MESSAGES\fR, and \fBNLSPATH\fR.
241 .SH EXIT STATUS
242 .sp
243 .LP
244 The following exit values are returned:
245 .sp
246 .ne 2
247 .na
248 \fB\fB0\fR\fR
249 .ad
250 .RS 5n
251 If any matches are found
252 .RE

254 .sp
255 .ne 2
256 .na
257 \fB\fB1\fR\fR
258 .ad
259 .RS 5n
260 If no matches are found
261 .RE

263 .sp
264 .ne 2
265 .na
266 \fB\fB2\fR\fR
267 .ad
268 .RS 5n
269 For syntax errors or inaccessible files, even if matches were found.
270 .RE

272 .SS "/usr/xpg4/bin/fgrep"
273 .sp

275 .SH ATTRIBUTES
276 .sp
277 .LP
278 See \fBattributes\fR(5) for descriptions of the following attributes:
279 .sp
280 .TS
281 box;
282 c | c
283 l | l .
284 ATTRIBUTE TYPE ATTRIBUTE VALUE
285
286 CSI Enabled
287 .TE

289 .SH SEE ALSO
290 .sp
291 .LP
292 \fB\bcd\fR(1), \fB\egrep\fR(1), \fB\grep\fR(1), \fB\sed\fR(1), \fB\sh\fR(1),
293 \fB\attributes\fR(5), \fB\environ\fR(5), \fB\largefile\fR(5), \fB\XPG4\fR(5)
294 .SH NOTES
295 .sp
296 .LP
297 Ideally, there should be only one \fBgrep\fR command, but there is not a single
298 algorithm that spans a wide enough range of space-time tradeoffs.
299 .sp
300 .LP
301 Lines are limited only by the size of the available virtual memory.
302 .SS "/usr/xpg4/bin/fgrep"
303 .sp

```

```

304 .LP
305 The \fB/usr/xpg4/bin/fgrep\fR utility is identical to \fB/usr/xpg4/bin/grep\fR
306 \fB-F\fR (see \fBgrep\fR(1)). Portable applications should use
307 \fB/usr/xpg4/bin/grep\fR \fB-F\fR.

```

```
new/usr/src/man/man1/grep.1
```

1

```
*****
14031 Thu May 30 19:29:53 2013
new/usr/src/man/man1/grep.1
3737 grep does not support -H option
*****
1 .\" te
2 .\" Copyright 2012 Nexenta Systems, Inc. All rights reserved.
3 .\" Copyright 1989 AT&T
4 .\" Copyright (c) 2008, Sun Microsystems, Inc. All Rights Reserved
5 .\" Portions Copyright (c) 1992, X/Open Company Limited All Rights Reserved
6 .\" Sun Microsystems, Inc. gratefully acknowledges The Open Group for permission
7 .\" http://www.opengroup.org/bookstore/.
8 .\" The Institute of Electrical and Electronics Engineers and The Open Group, ha
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 .\" You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE or http:
12 .\" When distributing Covered Code, include this CDDL HEADER in each file and in
13 .TH GREP 1 "May 3, 2013"
13 .TH GREP 1 "Feb 26, 2008"
14 .SH NAME
15 grep \- search a file for a pattern
16 .SH SYNOPSIS
17 .LP
18 .nf
19 \fB/usr/bin/grep\fR [\fB-c\fR | \fB-l\fR | \fB-q\fR] [\fB-r\fR | \fB-R\fR] [\fB-b
19 \fB/usr/bin/grep\fR [\fB-c\fR | \fB-l\fR | \fB-q\fR] [\fB-r\fR | \fB-R\fR] [\fB-b
20 \fIlimited-regular-expression\fR [\fIfilename\fR]...
21 .fi

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

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

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

43 .SH DESCRIPTION
44 .sp
45 .LP
46 The \fBgrep\fR utility searches text files for a pattern and prints all lines
47 that contain that pattern. It uses a compact non-deterministic algorithm.
48 .sp
49 .LP
50 Be careful using the characters \fB$\fR, \fB*\fR, \fB[\fR, \fB\fR, \fB\fR, \fB|\fR,
51 \fB(\fR, \fB)\fR, and \fB\|e\fR in the \fIpattern_list\fR because they are also
52 meaningful to the shell. It is safest to enclose the entire \fIpattern_list\fR
53 in single quotes \fBa\fR&... \fBa\fR&.
54 .sp
55 .LP
56 If no files are specified, \fBgrep\fR assumes standard input. Normally, each
```

```
new/usr/src/man/man1/grep.1
```

2

```
57 line found is copied to standard output. The file name is printed before each
58 line found if there is more than one input file.
59 .SS "/usr/bin/grep"
60 .sp
61 .LP
62 The \fB/usr/bin/grep\fR utility uses limited regular expressions like those
63 described on the \fBrexexp\fR(5) manual page to match the patterns.
64 .SS "/usr/xpg4/bin/grep"
65 .sp
66 .LP
67 The options \fB-E\fR and \fB-F\fR affect the way \fB/usr/xpg4/bin/grep\fR
68 interprets \fIpattern_list\fR. If \fB-E\fR is specified,
69 \fB/usr/xpg4/bin/grep\fR interprets \fIpattern_list\fR as a full regular
70 expression (see \fB-E\fR for description). If \fB-F\fR is specified,
71 \fBgrep\fR interprets \fIpattern_list\fR as a fixed string. If neither are
72 specified, \fBgrep\fR interprets \fIpattern_list\fR as a basic regular
73 expression as described on \fBrexexp\fR(5) manual page.
74 .SH OPTIONS
75 .sp
76 .LP
77 The following options are supported for both \fB/usr/bin/grep\fR and
78 \fB/usr/xpg4/bin/grep\fR:
79 .sp
80 .ne 2
81 .na
82 \fB\fB-b\fR\fR\fR
83 .ad
84 .RS 6n
85 Precedes each line by the block number on which it was found. This can be
86 useful in locating block numbers by context (first block is 0).
87 .RE

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

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

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

118 .sp
119 .ne 2
120 .na
121 \fB\fB-i\fR\fR\fR
122 .ad
```

```

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

127 .sp
128 .ne 2
129 .na
130 \fB\fB-1\fR\fR
131 .ad
132 .RS 6n
133 Prints only the names of files with matching lines, separated by NEWLINE
134 characters. Does not repeat the names of files when the pattern is found more
135 than once.
136 .RE

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

147 .sp
148 .ne 2
149 .na
150 \fB\fB-r\fR\fR
151 .ad
152 .RS 6n
153 Read all files under each directory, recursively. Follow symbolic links on
154 the command line, but skip symlinks that are encountered recursively. If file
155 is a device, FIFO, or socket, skip it.
156 .RE

158 .sp
159 .ne 2
160 .na
161 \fB\fB-R\fR\fR
162 .ad
163 .RS 6n
164 Read all files under each directory, recursively, following all symbolic links.
165 .RE

167 .sp
168 .ne 2
169 .na
170 \fB\fB-q\fR\fR
171 .ad
172 .RS 6n
173 Quiet. Does not write anything to the standard output, regardless of matching
174 lines. Exits with zero status if an input line is selected.
175 .RE

177 .sp
178 .ne 2
179 .na
180 \fB\fB-s\fR\fR
181 .ad
182 .RS 6n
183 Suppresses error messages about nonexistent or unreadable files.
184 .RE

186 .sp
187 .ne 2
188 .na

```

```

189 \fB\fB-v\fR\fR
190 .ad
191 .RS 6n
192 Prints all lines except those that contain the pattern.
193 .RE

195 .sp
196 .ne 2
197 .na
198 \fB\fB-w\fR\fR
199 .ad
200 .RS 6n
201 Searches for the expression as a word as if surrounded by \fB\<\fR and
202 \fB\>\fR&.
203 .RE

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

224 .sp
225 .ne 2
226 .na
227 \fB\fB-E\fR\fR
228 .ad
229 .RS 19n
230 Matches using full regular expressions. Treats each pattern specified as a full
231 regular expression. If any entire full regular expression pattern matches an
232 input line, the line is matched. A null full regular expression matches every
233 line. Each pattern is interpreted as a full regular expression as described on
234 the \fBregex\fR(5) manual page, except for \fB\<\fR and \fB\>\fR, and
235 including:
236 .RS +4
237 .TP
238 1.
239 A full regular expression followed by \fB+\fR that matches one or more
240 occurrences of the full regular expression.
241 .RE
242 .RS +4
243 .TP
244 2.
245 A full regular expression followed by \fB?\fR that matches 0 or 1
246 occurrences of the full regular expression.
247 .RE
248 .RS +4
249 .TP
250 3.
251 Full regular expressions separated by | or by a new-line that match strings
252 that are matched by any of the expressions.
253 .RE
254 .RS +4

```

```

255 .TP
256 4.
257 A full regular expression that is enclosed in parentheses \fB()\fR for
258 grouping.
259 .RE
260 The order of precedence of operators is \fB[|\]\fR, then \fB*\|?\|+\fR, then
261 concatenation, then | and new-line.
262 .RE

264 .sp
265 .ne 2
266 .na
267 \fB\fB-f\fR \fIpattern_file\fR\fR
268 .ad
269 .RS 19n
270 Reads one or more patterns from the file named by the path name
271 \fIpattern_file\fR. Patterns in \fIpattern_file\fR are terminated by a NEWLINE
272 character. A null pattern can be specified by an empty line in
273 \fIpattern_file\fR. Unless the \fB-E\fR or \fB-F\fR option is also specified,
274 each pattern is treated as a basic regular expression.
275 .RE

277 .sp
278 .ne 2
279 .na
280 \fB\fB-F\fR\fR
281 .ad
282 .RS 19n
283 Matches using fixed strings. Treats each pattern specified as a string instead
284 of a regular expression. If an input line contains any of the patterns as a
285 contiguous sequence of bytes, the line is matched. A null string matches every
286 line. See \fBfgrep\fR(1) for more information.
287 .RE

289 .sp
290 .ne 2
291 .na
292 \fB\fB-x\fR\fR
293 .ad
294 .RS 19n
295 Considers only input lines that use all characters in the line to match an
296 entire fixed string or regular expression to be matching lines.
297 .RE

299 .SH OPERANDS
300 .sp
301 .LP
302 The following operands are supported:
303 .sp
304 .ne 2
305 .na
306 \fB\fIfile\fR\fR
307 .ad
308 .RS 8n
309 A path name of a file to be searched for the patterns. If no \fIfile\fR
310 operands are specified, the standard input is used.
311 .RE

313 .SS "/usr/bin/grep"
314 .sp
315 .ne 2
316 .na
317 \fB\fIpattern\fR\fR
318 .ad
319 .RS 1ln
320 Specifies a pattern to be used during the search for input.

```

```

321 .RE
323 .SS "/usr/xpg4/bin/grep"
324 .sp
325 .ne 2
326 .na
327 \fB\fIpattern\fR\fR
328 .ad
329 .RS 1ln
330 Specifies one or more patterns to be used during the search for input. This
331 operand is treated as if it were specified as \fB-e\fR \fIpattern_list\fR.
332 .RE

334 .SH USAGE
335 .sp
336 .LP
337 The \fB-e\fR \fIpattern_list\fR option has the same effect as the
338 \fIpattern_list\fR operand, but is useful when \fIpattern_list\fR begins with
339 the hyphen delimiter. It is also useful when it is more convenient to provide
340 multiple patterns as separate arguments.
341 .sp
342 .LP
343 Multiple \fB-e\fR and \fB-f\fR options are accepted and \fBgrep\fR uses all of
344 the patterns it is given while matching input text lines. Notice that the order
345 of evaluation is not specified. If an implementation finds a null string as a
346 pattern, it is allowed to use that pattern first, matching every line, and
347 effectively ignore any other patterns.
348 .sp
349 .LP
350 The \fB-q\fR option provides a means of easily determining whether or not a
351 pattern (or string) exists in a group of files. When searching several files,
352 it provides a performance improvement (because it can quit as soon as it finds
353 the first match) and requires less care by the user in choosing the set of
354 files to supply as arguments (because it exits zero if it finds a match even if
355 \fBgrep\fR detected an access or read error on earlier file operands).
356 .SS "Large File Behavior"
357 .sp
358 .LP
359 See \fBlargefile\fR(5) for the description of the behavior of \fBgrep\fR when
360 encountering files greater than or equal to 2 Gbyte ( 2^31 bytes).
361 .SH EXAMPLES
362 .LP
363 \fBExample 1\fR Finding All Uses of a Word
364 .sp
365 .LP
366 To find all uses of the word "\fBPosix\fR" (in any case) in the file
367 \fBtext.mm\fR, and write with line numbers:
368 .sp
369 .in +2
370 .nf
371 example% \fB/usr/bin/grep -i -n posix text.mm\fR
372 .fi
373 .in -2
374 .sp
375 .LP
376 \fBExample 2\fR Finding All Empty Lines
377 .sp
378 To find all empty lines in the standard input:
379 .in +2
380 .nf
381 example% \fB/usr/bin/grep '^$\fR
382 .sp
383 .in +2
384 .nf
385 example% \fB/usr/bin/grep '^$\fR
386 .sp
387 .in +2
388 .nf
389 example% \fB/usr/bin/grep '^$\fR
390 .sp
391 .in +2
392 .nf
393 example% \fB/usr/bin/grep '^$\fR
394 .sp
395 .in +2
396 .nf
397 example% \fB/usr/bin/grep '^$\fR
398 .sp
399 .in +2
400 .nf
401 example% \fB/usr/bin/grep '^$\fR
402 .sp
403 .in +2
404 .nf
405 example% \fB/usr/bin/grep '^$\fR
406 .sp
407 .in +2
408 .nf
409 example% \fB/usr/bin/grep '^$\fR
410 .sp
411 .in +2
412 .nf
413 example% \fB/usr/bin/grep '^$\fR
414 .sp
415 .in +2
416 .nf
417 example% \fB/usr/bin/grep '^$\fR
418 .sp
419 .in +2
420 .nf
421 example% \fB/usr/bin/grep '^$\fR
422 .sp
423 .in +2
424 .nf
425 example% \fB/usr/bin/grep '^$\fR
426 .sp
427 .in +2
428 .nf
429 example% \fB/usr/bin/grep '^$\fR
430 .sp
431 .in +2
432 .nf
433 example% \fB/usr/bin/grep '^$\fR
434 .sp
435 .in +2
436 .nf
437 example% \fB/usr/bin/grep '^$\fR
438 .sp
439 .in +2
440 .nf
441 example% \fB/usr/bin/grep '^$\fR
442 .sp
443 .in +2
444 .nf
445 example% \fB/usr/bin/grep '^$\fR
446 .sp
447 .in +2
448 .nf
449 example% \fB/usr/bin/grep '^$\fR
450 .sp
451 .in +2
452 .nf
453 example% \fB/usr/bin/grep '^$\fR
454 .sp
455 .in +2
456 .nf
457 example% \fB/usr/bin/grep '^$\fR
458 .sp
459 .in +2
460 .nf
461 example% \fB/usr/bin/grep '^$\fR
462 .sp
463 .in +2
464 .nf
465 example% \fB/usr/bin/grep '^$\fR
466 .sp
467 .in +2
468 .nf
469 example% \fB/usr/bin/grep '^$\fR
470 .sp
471 .in +2
472 .nf
473 example% \fB/usr/bin/grep '^$\fR
474 .sp
475 .in +2
476 .nf
477 example% \fB/usr/bin/grep '^$\fR
478 .sp
479 .in +2
480 .nf
481 example% \fB/usr/bin/grep '^$\fR
482 .sp
483 .in +2
484 .nf
485 example% \fB/usr/bin/grep '^$\fR
486 .sp
487 .in +2
488 .nf
489 example% \fB/usr/bin/grep '^$\fR
490 .sp
491 .in +2
492 .nf
493 example% \fB/usr/bin/grep '^$\fR
494 .sp
495 .in +2
496 .nf
497 example% \fB/usr/bin/grep '^$\fR
498 .sp
499 .in +2
500 .nf
501 example% \fB/usr/bin/grep '^$\fR
502 .sp
503 .in +2
504 .nf
505 example% \fB/usr/bin/grep '^$\fR
506 .sp
507 .in +2
508 .nf
509 example% \fB/usr/bin/grep '^$\fR
510 .sp
511 .in +2
512 .nf
513 example% \fB/usr/bin/grep '^$\fR
514 .sp
515 .in +2
516 .nf
517 example% \fB/usr/bin/grep '^$\fR
518 .sp
519 .in +2
520 .nf
521 example% \fB/usr/bin/grep '^$\fR
522 .sp
523 .in +2
524 .nf
525 example% \fB/usr/bin/grep '^$\fR
526 .sp
527 .in +2
528 .nf
529 example% \fB/usr/bin/grep '^$\fR
530 .sp
531 .in +2
532 .nf
533 example% \fB/usr/bin/grep '^$\fR
534 .sp
535 .in +2
536 .nf
537 example% \fB/usr/bin/grep '^$\fR
538 .sp
539 .in +2
540 .nf
541 example% \fB/usr/bin/grep '^$\fR
542 .sp
543 .in +2
544 .nf
545 example% \fB/usr/bin/grep '^$\fR
546 .sp
547 .in +2
548 .nf
549 example% \fB/usr/bin/grep '^$\fR
550 .sp
551 .in +2
552 .nf
553 example% \fB/usr/bin/grep '^$\fR
554 .sp
555 .in +2
556 .nf
557 example% \fB/usr/bin/grep '^$\fR
558 .sp
559 .in +2
560 .nf
561 example% \fB/usr/bin/grep '^$\fR
562 .sp
563 .in +2
564 .nf
565 example% \fB/usr/bin/grep '^$\fR
566 .sp
567 .in +2
568 .nf
569 example% \fB/usr/bin/grep '^$\fR
570 .sp
571 .in +2
572 .nf
573 example% \fB/usr/bin/grep '^$\fR
574 .sp
575 .in +2
576 .nf
577 example% \fB/usr/bin/grep '^$\fR
578 .sp
579 .in +2
580 .nf
581 example% \fB/usr/bin/grep '^$\fR
582 .sp
583 .in +2
584 .nf
585 example% \fB/usr/bin/grep '^$\fR
586 .sp
587 .in +2
588 .nf
589 example% \fB/usr/bin/grep '^$\fR
590 .sp
591 .in +2
592 .nf
593 example% \fB/usr/bin/grep '^$\fR
594 .sp
595 .in +2
596 .nf
597 example% \fB/usr/bin/grep '^$\fR
598 .sp
599 .in +2
600 .nf
601 example% \fB/usr/bin/grep '^$\fR
602 .sp
603 .in +2
604 .nf
605 example% \fB/usr/bin/grep '^$\fR
606 .sp
607 .in +2
608 .nf
609 example% \fB/usr/bin/grep '^$\fR
610 .sp
611 .in +2
612 .nf
613 example% \fB/usr/bin/grep '^$\fR
614 .sp
615 .in +2
616 .nf
617 example% \fB/usr/bin/grep '^$\fR
618 .sp
619 .in +2
620 .nf
621 example% \fB/usr/bin/grep '^$\fR
622 .sp
623 .in +2
624 .nf
625 example% \fB/usr/bin/grep '^$\fR
626 .sp
627 .in +2
628 .nf
629 example% \fB/usr/bin/grep '^$\fR
630 .sp
631 .in +2
632 .nf
633 example% \fB/usr/bin/grep '^$\fR
634 .sp
635 .in +2
636 .nf
637 example% \fB/usr/bin/grep '^$\fR
638 .sp
639 .in +2
640 .nf
641 example% \fB/usr/bin/grep '^$\fR
642 .sp
643 .in +2
644 .nf
645 example% \fB/usr/bin/grep '^$\fR
646 .sp
647 .in +2
648 .nf
649 example% \fB/usr/bin/grep '^$\fR
650 .sp
651 .in +2
652 .nf
653 example% \fB/usr/bin/grep '^$\fR
654 .sp
655 .in +2
656 .nf
657 example% \fB/usr/bin/grep '^$\fR
658 .sp
659 .in +2
660 .nf
661 example% \fB/usr/bin/grep '^$\fR
662 .sp
663 .in +2
664 .nf
665 example% \fB/usr/bin/grep '^$\fR
666 .sp
667 .in +2
668 .nf
669 example% \fB/usr/bin/grep '^$\fR
670 .sp
671 .in +2
672 .nf
673 example% \fB/usr/bin/grep '^$\fR
674 .sp
675 .in +2
676 .nf
677 example% \fB/usr/bin/grep '^$\fR
678 .sp
679 .in +2
680 .nf
681 example% \fB/usr/bin/grep '^$\fR
682 .sp
683 .in +2
684 .nf
685 example% \fB/usr/bin/grep '^$\fR
686 .sp
687 .in +2
688 .nf
689 example% \fB/usr/bin/grep '^$\fR
690 .sp
691 .in +2
692 .nf
693 example% \fB/usr/bin/grep '^$\fR
694 .sp
695 .in +2
696 .nf
697 example% \fB/usr/bin/grep '^$\fR
698 .sp
699 .in +2
700 .nf
701 example% \fB/usr/bin/grep '^$\fR
702 .sp
703 .in +2
704 .nf
705 example% \fB/usr/bin/grep '^$\fR
706 .sp
707 .in +2
708 .nf
709 example% \fB/usr/bin/grep '^$\fR
710 .sp
711 .in +2
712 .nf
713 example% \fB/usr/bin/grep '^$\fR
714 .sp
715 .in +2
716 .nf
717 example% \fB/usr/bin/grep '^$\fR
718 .sp
719 .in +2
720 .nf
721 example% \fB/usr/bin/grep '^$\fR
722 .sp
723 .in +2
724 .nf
725 example% \fB/usr/bin/grep '^$\fR
726 .sp
727 .in +2
728 .nf
729 example% \fB/usr/bin/grep '^$\fR
730 .sp
731 .in +2
732 .nf
733 example% \fB/usr/bin/grep '^$\fR
734 .sp
735 .in +2
736 .nf
737 example% \fB/usr/bin/grep '^$\fR
738 .sp
739 .in +2
740 .nf
741 example% \fB/usr/bin/grep '^$\fR
742 .sp
743 .in +2
744 .nf
745 example% \fB/usr/bin/grep '^$\fR
746 .sp
747 .in +2
748 .nf
749 example% \fB/usr/bin/grep '^$\fR
750 .sp
751 .in +2
752 .nf
753 example% \fB/usr/bin/grep '^$\fR
754 .sp
755 .in +2
756 .nf
757 example% \fB/usr/bin/grep '^$\fR
758 .sp
759 .in +2
760 .nf
761 example% \fB/usr/bin/grep '^$\fR
762 .sp
763 .in +2
764 .nf
765 example% \fB/usr/bin/grep '^$\fR
766 .sp
767 .in +2
768 .nf
769 example% \fB/usr/bin/grep '^$\fR
770 .sp
771 .in +2
772 .nf
773 example% \fB/usr/bin/grep '^$\fR
774 .sp
775 .in +2
776 .nf
777 example% \fB/usr/bin/grep '^$\fR
778 .sp
779 .in +2
780 .nf
781 example% \fB/usr/bin/grep '^$\fR
782 .sp
783 .in +2
784 .nf
785 example% \fB/usr/bin/grep '^$\fR
786 .sp
787 .in +2
788 .nf
789 example% \fB/usr/bin/grep '^$\fR
790 .sp
791 .in +2
792 .nf
793 example% \fB/usr/bin/grep '^$\fR
794 .sp
795 .in +2
796 .nf
797 example% \fB/usr/bin/grep '^$\fR
798 .sp
799 .in +2
800 .nf
801 example% \fB/usr/bin/grep '^$\fR
802 .sp
803 .in +2
804 .nf
805 example% \fB/usr/bin/grep '^$\fR
806 .sp
807 .in +2
808 .nf
809 example% \fB/usr/bin/grep '^$\fR
810 .sp
811 .in +2
812 .nf
813 example% \fB/usr/bin/grep '^$\fR
814 .sp
815 .in +2
816 .nf
817 example% \fB/usr/bin/grep '^$\fR
818 .sp
819 .in +2
820 .nf
821 example% \fB/usr/bin/grep '^$\fR
822 .sp
823 .in +2
824 .nf
825 example% \fB/usr/bin/grep '^$\fR
826 .sp
827 .in +2
828 .nf
829 example% \fB/usr/bin/grep '^$\fR
830 .sp
831 .in +2
832 .nf
833 example% \fB/usr/bin/grep '^$\fR
834 .sp
835 .in +2
836 .nf
837 example% \fB/usr/bin/grep '^$\fR
838 .sp
839 .in +2
840 .nf
841 example% \fB/usr/bin/grep '^$\fR
842 .sp
843 .in +2
844 .nf
845 example% \fB/usr/bin/grep '^$\fR
846 .sp
847 .in +2
848 .nf
849 example% \fB/usr/bin/grep '^$\fR
850 .sp
851 .in +2
852 .nf
853 example% \fB/usr/bin/grep '^$\fR
854 .sp
855 .in +2
856 .nf
857 example% \fB/usr/bin/grep '^$\fR
858 .sp
859 .in +2
860 .nf
861 example% \fB/usr/bin/grep '^$\fR
862 .sp
863 .in +2
864 .nf
865 example% \fB/usr/bin/grep '^$\fR
866 .sp
867 .in +2
868 .nf
869 example% \fB/usr/bin/grep '^$\fR
870 .sp
871 .in +2
872 .nf
873 example% \fB/usr/bin/grep '^$\fR
874 .sp
875 .in +2
876 .nf
877 example% \fB/usr/bin/grep '^$\fR
878 .sp
879 .in +2
880 .nf
881 example% \fB/usr/bin/grep '^$\fR
882 .sp
883 .in +2
884 .nf
885 example% \fB/usr/bin/grep '^$\fR
886 .sp
887 .in +2
888 .nf
889 example% \fB/usr/bin/grep '^$\fR
890 .sp
891 .in +2
892 .nf
893 example% \fB/usr/bin/grep '^$\fR
894 .sp
895 .in +2
896 .nf
897 example% \fB/usr/bin/grep '^$\fR
898 .sp
899 .in +2
900 .nf
901 example% \fB/usr/bin/grep '^$\fR
902 .sp
903 .in +2
904 .nf
905 example% \fB/usr/bin/grep '^$\fR
906 .sp
907 .in +2
908 .nf
909 example% \fB/usr/bin/grep '^$\fR
910 .sp
911 .in +2
912 .nf
913 example% \fB/usr/bin/grep '^$\fR
914 .sp
915 .in +2
916 .nf
917 example% \fB/usr/bin/grep '^$\fR
918 .sp
919 .in +2
920 .nf
921 example% \fB/usr/bin/grep '^$\fR
922 .sp
923 .in +2
924 .nf
925 example% \fB/usr/bin/grep '^$\fR
926 .sp
927 .in +2
928 .nf
929 example% \fB/usr/bin/grep '^$\fR
930 .sp
931 .in +2
932 .nf
933 example% \fB/usr/bin/grep '^$\fR
934 .sp
935 .in +2
936 .nf
937 example% \fB/usr/bin/grep '^$\fR
938 .sp
939 .in +2
940 .nf
941 example% \fB/usr/bin/grep '^$\fR
942 .sp
943 .in +2
944 .nf
945 example% \fB/usr/bin/grep '^$\fR
946 .sp
947 .in +2
948 .nf
949 example% \fB/usr/bin/grep '^$\fR
950 .sp
951 .in +2
952 .nf
953 example% \fB/usr/bin/grep '^$\fR
954 .sp
955 .in +2
956 .nf
957 example% \fB/usr/bin/grep '^$\fR
958 .sp
959 .in +2
960 .nf
961 example% \fB/usr/bin/grep '^$\fR
962 .sp
963 .in +2
964 .nf
965 example% \fB/usr/bin/grep '^$\fR
966 .sp
967 .in +2
968 .nf
969 example% \fB/usr/bin/grep '^$\fR
970 .sp
971 .in +2
972 .nf
973 example% \fB/usr/bin/grep '^$\fR
974 .sp
975 .in +2
976 .nf
977 example% \fB/usr/bin/grep '^$\fR
978 .sp
979 .in +2
980 .nf
981 example% \fB/usr/bin/grep '^$\fR
982 .sp
983 .in +2
984 .nf
985 example% \fB/usr/bin/grep '^$\fR
986 .sp
987 .in +2
988 .nf
989 example% \fB/usr/bin/grep '^$\fR
990 .sp
991 .in +2
992 .nf
993 example% \fB/usr/bin/grep '^$\fR
994 .sp
995 .in +2
996 .nf
997 example% \fB/usr/bin/grep '^$\fR
998 .sp
999 .in +2
1000 .nf
1001 example% \fB/usr/bin/grep '^$\fR
1002 .sp
1003 .in +2
1004 .nf
1005 example% \fB/usr/bin/grep '^$\fR
1006 .sp
1007 .in +2
1008 .nf
1009 example% \fB/usr/bin/grep '^$\fR
1010 .sp
1011 .in +2
1012 .nf
1013 example% \fB/usr/bin/grep '^$\fR
1014 .sp
1015 .in +2
1016 .nf
1017 example% \fB/usr/bin/grep '^$\fR
1018 .sp
1019 .in +2
1020 .nf
1021 example% \fB/usr/bin/grep '^$\fR
1022 .sp
1023 .in +2
1024 .nf
1025 example% \fB/usr/bin/grep '^$\fR
1026 .sp
1027 .in +2
1028 .nf
1029 example% \fB/usr/bin/grep '^$\fR
1030 .sp
1031 .in +2
1032 .nf
1033 example% \fB/usr/bin/grep '^$\fR
1034 .sp
1035 .in +2
1036 .nf
1037 example% \fB/usr/bin/grep '^$\fR
1038 .sp
1039 .in +2
1040 .nf
1041 example% \fB/usr/bin/grep '^$\fR
1042 .sp
1043 .in +2
1044 .nf
1045 example% \fB/usr/bin/grep '^$\fR
1046 .sp
1047 .in +2
1048 .nf
1049 example% \fB/usr/bin/grep '^$\fR
1050 .sp
1051 .in +2
1052 .nf
1053 example% \fB/usr/bin/grep '^$\fR
1054 .sp
1055 .in +2
1056 .nf
1057 example% \fB/usr/bin/grep '^$\fR
1058 .sp
1059 .in +2
1060 .nf
1061 example% \fB/usr/bin/grep '^$\fR
1062 .sp
1063 .in +2
1064 .nf
1065 example% \fB/usr/bin/grep '^$\fR
1066 .sp
1067 .in +2
1068 .nf
1069 example% \fB/usr/bin/grep '^$\fR
1070 .sp
1071 .in +2
1072 .nf
1073 example% \fB/usr/bin/grep '^$\fR
1074 .sp
1075 .in +2
1076 .nf
1077 example% \fB/usr/bin/grep '^$\fR
1078 .sp
1079 .in +2
1080 .nf
1081 example% \fB/usr/bin/grep '^$\fR
1082 .sp
1083 .in +2
1084 .nf
1085 example% \fB/usr/bin/grep '^$\fR
1086 .sp
1087 .in +2
1088 .nf
1089 example% \fB/usr/bin/grep '^$\fR
1090 .sp
1091 .in +2
1092 .nf
1093 example% \fB/usr/bin/grep '^$\fR
1094 .sp
1095 .in +2
1096 .nf
1097 example% \fB/usr/bin/grep '^$\fR
1098 .sp
1099 .in +2
1100 .nf
1101 example% \fB/usr/bin/grep '^$\fR
1102 .sp
1103 .in +2
1104 .nf
1105 example% \fB/usr/bin/grep '^$\fR
1106 .sp
1107 .in +2
1108 .nf
1109 example% \fB/usr/bin/grep '^$\fR
1110 .sp
1111 .in +2
1112 .nf
1113 example% \fB/usr/bin/grep '^$\fR
1114 .sp
1115 .in +2
1116 .nf
1117 example% \fB/usr/bin/grep '^$\fR
1118 .sp
1119 .in +2
1120 .nf
1121 example% \fB/usr/bin/grep '^$\fR
1122 .sp
1123 .in +2
1124 .nf
1125 example% \fB/usr/bin/grep '^$\fR
1126 .sp
1127 .in +2
1128 .nf
1129 example% \fB/usr/bin/grep '^$\fR
1130 .sp
1131 .in +2
1132 .nf
1133 example% \fB/usr/bin/grep '^$\fR
1134 .sp
1135 .in +2
1136 .nf
1137 example% \fB/usr/bin/grep '^$\fR
1138 .sp
1139 .in +2
1140 .nf
1141 example% \fB/usr/bin/grep '^$\fR
1142 .sp
1143 .in +2
1144 .nf
1145 example% \fB/usr/bin/grep '^$\fR
1146 .sp
1147 .in +2
1148 .nf
1149 example% \fB/usr/bin/grep '^$\fR
1150 .sp
1151 .in +2
1152 .nf
1153 example% \fB/usr/bin/grep '^$\fR
1154 .sp
1155 .in +2
1156 .nf
1157 example% \fB/usr/bin/grep '^$\fR
1158 .sp
1159 .in +2
1160 .nf
1161 example% \fB/usr/bin/grep '^$\fR
1162 .sp
1163 .in +2
1164 .nf
1165 example% \fB/usr/bin/grep '^$\fR
1166 .sp
1167 .in +2
1168 .nf
1169 example% \fB/usr/bin/grep '^$\fR
1170 .sp
1171 .in +2
1172 .nf
1173 example% \fB/usr/bin/grep '^$\fR
1174 .sp
1175 .in +2
1176 .nf
1177 example% \fB/usr/bin/grep '^$\fR
1178 .sp
1179 .in +2
1180 .nf
1181 example% \fB/usr/bin/grep '^$\fR
1182 .sp
1183 .in +2
1184 .nf
1185 example% \fB/usr/bin/grep '^$\fR
1186 .sp
1187 .in +2
1188 .nf
1189 example% \fB/usr/bin/grep '^$\fR
1190 .sp
1191 .in +2
1192 .nf
1193 example% \fB/usr/bin/grep '^$\fR
1194 .sp
1195 .in +2
1196 .nf
1197 example% \fB/usr/bin/grep '^$\fR
1198 .sp
1199 .in +2
1200 .nf
1201 example% \fB/usr/bin/grep '^$\fR
1202 .sp
1203 .in +2
1204 .nf
1205 example% \fB/usr/bin/grep '^$\fR
1206 .sp
1207 .in +2
1208 .nf
1209 example% \fB/usr/bin/grep '^$\fR
1210 .sp
1211 .in +2
1212 .nf
1213 example% \fB/usr/bin/grep '^$\fR
1214 .sp
1215 .in +2
1216 .nf
1217 example% \fB/usr/bin/grep '^$\fR
1218 .sp
1219 .in +2
1220 .nf
1221 example% \fB/usr/bin/grep '^$\fR
1222 .sp
1223 .in +2
1224 .nf
1225 example% \fB/usr/bin/grep '^$\fR
1226 .sp
1227 .in +2
1228 .nf
1229 example% \fB/usr/bin/grep '^$\fR
1230 .sp
1231 .in +2
1232 .nf
1233 example% \fB/usr/bin/grep '^$\fR
1234 .sp
1235 .in +2
1236 .nf
1237 example% \fB/usr/bin/grep '^$\fR
1238 .sp
1239 .in +2
1240 .nf
1241 example% \fB/usr/bin/grep '^$\fR
1242 .sp
1243 .in +2
1244 .nf
1245 example% \fB/usr/bin/grep '^$\fR
1246 .sp
1247 .in +2
1248 .nf
1249 example% \fB/usr/bin/grep '^$\fR
1250 .sp
1251 .in +2
1252 .nf
1253 example% \fB/usr/bin/grep '^$\fR
1254 .sp
1255 .in +2
1256 .nf
1257 example% \fB/usr/bin/grep '^$\fR
1258 .sp
1259 .in +2
1260 .nf
1261 example% \fB/usr/bin/grep '^$\fR
1262 .sp
1263 .in +2
1264 .nf
1265 example% \fB/usr/bin/grep '^$\fR
1266 .sp
1267 .in +2
1268 .nf
1269 example% \fB/usr/bin/grep '^$\fR
1270 .sp
1271 .in +2
1272 .nf
1273 example% \fB/usr/bin/grep '^$\fR
1274 .sp
1275 .in +2
1276 .nf
1277 example% \fB/usr/bin/grep '^$\fR
1278 .sp
1279 .in +2
1280 .nf
1281 example% \fB/usr/bin/grep '^$\fR
1282 .sp
1283 .in +2
1284 .nf
1285 example% \fB/usr/bin/grep '^$\fR
1286 .sp
1287 .in +2
1288 .nf
1289 example% \fB/usr/bin/grep '^$\fR
1290 .sp
1291 .in +2
1292 .nf
1293 example% \fB/usr/bin/grep '^$\fR
1294 .sp
1295 .in +2
1296 .nf
1297 example% \fB/usr/bin/grep '^$\fR
1298 .sp
1299 .in +2
1300 .nf
1301 example% \fB/usr/bin/grep '^$\fR
1302 .sp
1303 .in +2
1304 .nf
1305 example% \fB/usr/bin/grep '^$\fR
1306 .sp
1307 .in +2
1308 .nf
1309 example% \fB/usr/bin/grep '^$\fR
1310 .sp
1311 .in +2
1312 .nf
1313 example% \fB/usr/bin/grep '^$\fR
1314 .sp
1315 .in +2
1316 .nf
1317 example% \fB/usr/bin/grep '^$\fR
1318 .sp
1319 .in +2
1320 .nf
1321 example% \fB/usr/bin/grep '^$\fR
1322 .sp
1323 .in +2
1324 .nf
1325 example% \fB/usr/bin/grep '^$\fR
1326 .sp
1327 .in +2
1328 .nf
1329 example% \fB/usr/bin/grep '^$\fR
1330 .sp
1331 .in +2
1332 .nf
1333 example% \fB/usr/bin/grep '^$\fR
1334 .sp
1335 .in +2
1336 .nf
1337 example% \fB/usr/bin/grep '^$\fR
1338 .sp
1339 .in +2
1340 .nf
1341 example% \fB/usr/bin/grep '^$\fR
1342 .sp
1343 .in +2
1344 .nf
1345 example% \fB/usr/bin/grep '^$\fR
1346 .sp
1347 .in +2
1348 .nf
1349 example% \fB/usr/bin/grep '^$\fR
1350 .sp
1351 .in +2
1352 .nf
1353 example% \fB/usr/bin/grep '^$\fR
1354 .sp
1355 .in +2
1356 .nf
1357 example% \fB/usr/bin/grep '^$\fR
1358 .sp
1359 .in +2
1360 .nf
1361 example% \fB/usr/bin/grep '^$\fR
1362 .sp
1363 .in +2
1364 .nf
1365 example% \fB/usr/bin/grep '^$\fR
1366 .sp
1367 .in +2
1368 .nf
1369 example% \fB/usr/bin/grep '^$\fR
1370 .sp
1371 .in +2
1372 .nf
1373 example% \fB/usr/bin/grep '^$\fR
1374 .sp
1375 .in +2
1376 .nf
1377 example% \fB/usr/bin/grep '^$\fR
1378 .sp
1379 .in +2
1380 .nf
1381 example% \fB/usr/bin/grep '^$\fR
1382 .sp
1383 .in +2
1384 .nf
1385 example% \fB/usr/bin/grep '^$\fR
1386 .sp
1387 .in +2
1388 .nf
1389 example% \fB/usr/bin/grep '^$\fR
1390 .sp
1391 .in +2
1392 .nf
1393 example% \fB
```

```

387 .fi
388 .in -2
389 .sp
390 .sp
391 .sp
392 .LP
393 or
394 .sp
395 .in +2
396 .nf
397 example% \fB/usr/bin/grep -v .\fR
398 .fi
399 .in -2
400 .sp
401 .sp

403 .LP
404 \fBExample 3\fR Finding Lines Containing Strings
405 .sp
406 .LP
407 All of the following commands print all lines containing strings \fBabc\fR or
408 \fBdef\fR or both:
409 .sp
410 .in +2
411 .nf
412 example% \fB/usr/xpg4/bin/grep 'abc
413 def'\fR
414 example% \fB/usr/xpg4/bin/grep -e 'abc
415 def'\fR
416 example% \fB/usr/xpg4/bin/grep -e 'abc' -e 'def'\fR
417 example% \fB/usr/xpg4/bin/grep -E 'abc|def'\fR
418 example% \fB/usr/xpg4/bin/grep -E -e 'abc|def'\fR
419 example% \fB/usr/xpg4/bin/grep -E -e 'abc' -e 'def'\fR
420 example% \fB/usr/xpg4/bin/grep -E -e 'abc' -e 'def'\fR
421 example% \fB/usr/xpg4/bin/grep -E 'abc
422 def'\fR
423 example% \fB/usr/xpg4/bin/grep -E -e 'abc
424 def'\fR
425 example% \fB/usr/xpg4/bin/grep -F -e 'abc' -e 'def'\fR
426 example% \fB/usr/xpg4/bin/grep -F 'abc
427 def'\fR
428 example% \fB/usr/xpg4/bin/grep -F -e 'abc
429 def'\fR
430 .fi
431 .in -2
432 .sp

434 .LP
435 \fBExample 4\fR Finding Lines with Matching Strings
436 .sp
437 .LP
438 Both of the following commands print all lines matching exactly \fBabc\fR or
439 \fBdef\fR:
440 .sp
441 .in +2
442 .nf
443 example% \fB/usr/xpg4/bin/grep -E '^abc$ ^def$'\fR
444 example% \fB/usr/xpg4/bin/grep -F -x 'abc def'\fR
445 .fi
446 .in -2
447 .sp
448 .sp

450 .SH ENVIRONMENT VARIABLES
451 .sp
452 .LP

```

```

453 See \fBenvironment\fR(5) for descriptions of the following environment variables
454 that affect the execution of \fBgrep\fR: \fBLANG\fR, \fBLC_ALL\fR,
455 \fBLC_COLLATE\fR, \fBLC_CTYPE\fR, \fBLC_MESSAGES\fR, and \fBNLSPATH\fR.
456 .SH EXIT STATUS
457 .sp
458 .LP
459 The following exit values are returned:
460 .sp
461 .ne 2
462 .na
463 \fB\fBO\fR
464 .ad
465 .RS 5n
466 One or more matches were found.
467 .RE

469 .sp
470 .ne 2
471 .na
472 \fB\fB1\fR
473 .ad
474 .RS 5n
475 No matches were found.
476 .RE

478 .sp
479 .ne 2
480 .na
481 \fB\fB2\fR
482 .ad
483 .RS 5n
484 Syntax errors or inaccessible files (even if matches were found).
485 .RE

487 .SH ATTRIBUTES
488 .sp
489 .LP
490 See \fBattributes\fR(5) for descriptions of the following attributes:
491 .SS "/usr/bin/grep"
492 .sp

494 .sp
495 .TS
496 box;
497 c | c
498 l | l .
499 ATTRIBUTE TYPE ATTRIBUTE VALUE
500 -
501 CSI Not Enabled
502 .TE

504 .SS "/usr/xpg4/bin/grep"
505 .sp

507 .sp
508 .TS
509 box;
510 c | c
511 l | l .
512 ATTRIBUTE TYPE ATTRIBUTE VALUE
513 -
514 CSI Enabled
515 -
516 Interface Stability Committed
517 -
518 Standard See \fBstandards\fR(5).

```

```
519 .TE
521 .SH SEE ALSO
522 .sp
523 .LP
524 \fBgrep\fR(1), \fBfgrep\fR(1), \fBsed\fR(1), \fBsh\fR(1), \fBattributes\fR(5),
525 \fBenvir\fR(5), \fBlargefile\fR(5), \fBregex\fR(5), \fBregexp\fR(5),
526 \fBstandards\fR(5)
527 .SH NOTES
528 .SS "/usr/bin/grep"
529 .sp
530 .LP
531 Lines are limited only by the size of the available virtual memory. If there is
532 a line with embedded nulls, \fBgrep\fR only matches up to the first null. If
533 the line matches, the entire line is printed.
534 .SS "/usr/xpg4/bin/grep"
535 .sp
536 .LP
537 The results are unspecified if input files contain lines longer than
538 \fBLINE_MAX\fR bytes or contain binary data. \fBLINE_MAX\fR is defined in
539 \fB/usr/include/limits.h\fR.
```