```
**********************************************************
   10801 Sat Sep 14 23:25:09 2013
new/usr/src/cmd/grep/grep.c
3546 add support for grep -o option
**********************************************************
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License, Version 1.0 only
 * (the "License").  You may not use this file except in compliance
 * with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */
/*
 * Copyright 2005 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 */

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

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

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

/*
 * Copyright 2013 Damian Bogel. All rights reserved.
 * Copyright (c) 2013 Andrew Stormont.  All rights reserved.
#endif /* ! codereview */
 */

/*
 * grep -- print lines matching (or not matching) a pattern
 *
 *      status returns:
 *              0 - ok, and some matches
 *              1 - ok, but no matches
 *              2 - some error
 */

#include <sys/types.h>

#include <ctype.h>
#include <fcntl.h>
#include <locale.h>
#include <memory.h>
#include <regexpr.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <ftw.h>
```

```
#include <limits.h>
#include <sys/param.h>

static const char *errstr[] = {
        "Range endpoint too large.",
        "Bad number.",
        "``\\digit'' out of range.",
        "No remembered search string.",
        "\\( \\) imbalance.",
        "Too many \\(.",
        "More than 2 numbers given in \\{ \\}.",
        "} expected after \\.",
        "First number exceeds second in \\{ \\}.",
        "[ ] imbalance.",
        "Regular expression overflow.",
        "Illegal byte sequence.",
        "Unknown regexp error code!!",
        NULL
};

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

#define errmsg(msg, arg)        (void) fprintf(stderr, gettext(msg), arg)
#define BLKSIZE 512
#define GBUFSIZ 8192
#define MAX_DEPTH       1000

static int      temp;
static long long        lnum;
static char     *linebuf;
static char     *prntbuf = NULL;
static long     fw_lPrntBufLen = 0;
static int      nflag;
static int      bflag;
static int      lflag;
static int      cflag;
static int      rflag;
static int      Rflag;
static int      vflag;
static int      sflag;
static int      iflag;
static int      wflag;
static int      hflag;
static int      Hflag;
static int      qflag;
static int      oflag;
#endif /* ! codereview */
static int      errflg;
static int      nfile;
static long long        tln;
static int      nsucc;
static int      outfn = 0;
static int      nlflag;
static char     *ptr, *ptrend;
static char     *expbuf;

static void     execute(const char *, int);
static void     regerr(int);
static void     prepare(const char *);
static int      recursive(const char *, const struct stat *, int, struct FTW *);
static int      succeed(const char *);

int
main(int argc, char **argv)
{
        int     c;
        char    *arg;
        extern int      optind;

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

        while ((c = getopt(argc, argv, "hHqblcnoRrsviyw")) != -1)
        while ((c = getopt(argc, argv, "hHqblcnRrsviyw")) != -1)
                switch (c) {
                /* based on options order h or H is set as in GNU grep */
                case ’h’:
                        hflag++;
                        Hflag = 0; /* h excludes H */
                        break;
                case ’H’:
                        if (!lflag) /* H is excluded by l */
                                Hflag++;
                        hflag = 0; /* H excludes h */
                        break;
                case ’q’:       /* POSIX: quiet: status only */
                        qflag++;
                        break;
                case ’v’:
                        vflag++;
                        break;
                case ’c’:
                        cflag++;
                        break;
                case ’n’:
                        nflag++;
                        break;
                case ’o’:
                        oflag++;
                        break;
#endif /* ! codereview */
                case ’R’:
                        Rflag++;
                        /* FALLTHROUGH */
                case ’r’:
                        rflag++;
                        break;
                case ’b’:
                        bflag++;
                        break;
                case ’s’:
                        sflag++;
                        break;
                case ’l’:
                        lflag++;
                        Hflag = 0; /* l excludes H */
                        break;
                case ’y’:
                case ’i’:
                        iflag++;
                        break;
                case ’w’:
                        wflag++;
                        break;
                case ’?’:
                        errflg++;
                }

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

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

        argv = &argv[optind];
        argc -= optind;
        nfile = argc - 1;

        if (strrchr(*argv, ’\n’) != NULL)
                regerr(41);

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

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

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

                (void) strcpy(wordbuf, "\\<");
                (void) strcat(wordbuf, *argv);
                (void) strcat(wordbuf, "\\>");
                *argv = wordbuf;
        }

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

        if (--argc == 0)
                execute(NULL, 0);
        else
                while (argc-- > 0)
                        prepare(*++argv);

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

______***unchanged_portion_omitted_***

```
static void
execute(const char *file, int base)
{
        char    *lbuf, *p;
        long    count;
        long    offset = 0;
        char    *next_ptr = NULL;
        long    next_count = 0;

        tln = 0;

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

        if (file == NULL) {
                temp = 0;
                file = STDIN_FILENAME;
        } else if ((temp = open(file + base, O_RDONLY)) == -1) {
                if (!sflag)
                        errmsg("grep: can’t open %s\n", file);
                nsucc = 2;
                return;
        }

        /* read in first block of bytes */
        if ((count = read(temp, prntbuf, GBUFSIZ)) <= 0) {
                (void) close(temp);

                if (cflag && !qflag) {
                        if (Hflag || (nfile > 1 && !hflag))
                                (void) fprintf(stdout, "%s:", file);
                        if (!rflag)
                        (void) fprintf(stdout, "%lld\n", tln);
                }
                return;
        }

        lnum = 0;
        ptr = prntbuf;
        for (;;) {
                /* look for next newline */
                if ((ptrend = memchr(ptr + offset, ’\n’, count)) == NULL) {
                        offset += count;

                        /*
                         * shift unused data to the beginning of the buffer
                         */
                        if (ptr > prntbuf) {
                                (void) memmove(prntbuf, ptr, offset);
                                ptr = prntbuf;
                        }

                        /*
                         * re-allocate a larger buffer if this one is full
                         */
                        if (offset + GBUFSIZ > fw_lPrntBufLen) {
                                /*
                                 * allocate a new buffer and preserve the
                                 * contents...
                                 */
                                fw_lPrntBufLen += GBUFSIZ;
                                if ((prntbuf = realloc(prntbuf,
                                    fw_lPrntBufLen)) == NULL)
                                        exit(2);

                                /*
                                 * set up a bigger linebuffer (this is only used
                                 * for case insensitive operations). Contents do
                                 * not have to be preserved.
                                 */
                                free(linebuf);
                                if ((linebuf = malloc(fw_lPrntBufLen)) == NULL)
                                        exit(2);

                                ptr = prntbuf;
                        }
```

```

                                p = prntbuf + offset;
                                if ((count = read(temp, p, GBUFSIZ)) > 0)
                                        continue;

                                if (offset == 0)
                                        /* end of file already reached */
                                        break;

                                /* last line of file has no newline */
                                ptrend = ptr + offset;
                                nlflag = 0;
                        } else {
                                next_ptr = ptrend + 1;
                                next_count = offset + count - (next_ptr - ptr);
                                nlflag = 1;
                        }
                        lnum++;
                        *ptrend = ’\0’;

                        if (iflag) {
                                /*
                                 * Make a lower case copy of the record
                                 */
                                p = ptr;
                                for (lbuf = linebuf; p < ptrend; )
                                        *lbuf++ = (char)tolower((int)
                                            (unsigned char)*p++);
                                *lbuf = ’\0’;
                                lbuf = linebuf;
                        } else
                                /*
                                 * Use record as is
                                 */
                                lbuf = ptr;

                        /* lflag only once */
                        if (step(lbuf, expbuf) ^ vflag) {
                                if (oflag) {
                                        /*
                                         * Only store the matching bits
                                         */
                                        ptr = loc1;
                                        ptrend = loc2;
                                }
                                if (succeed(file) == 1)
                        if ((step(lbuf, expbuf) ^ vflag) && succeed(file) == 1)
                                        break;
                        }
#endif /* ! codereview */

                        if (!nlflag)
                                break;

                        ptr = next_ptr;
                        count = next_count;
                        offset = 0;
                }
                (void) close(temp);

                if (cflag && !qflag) {
                        if (Hflag || (!hflag && ((nfile > 1) ||
                            (rflag && outfn))))
                                (void) fprintf(stdout, "%s:", file);
                        (void) fprintf(stdout, "%lld\n", tln);
                }
```

```
}

static int
succeed(const char *f)
{
        int nchars;
        nsucc = (nsucc == 2) ? 2 : 1;

        if (qflag) {
                /* no need to continue */
                return (1);
        }

        if (cflag) {
                tln++;
                return (0);
        }

        if (lflag) {
                (void) fprintf(stdout, "%s\n", f);
                return (1);
        }

        if (Hflag || (!hflag && (nfile > 1 || (rflag && outfn)))) {
                /* print filename */
                (void) fprintf(stdout, "%s:", f);
        }

        if (bflag)
                /* print block number */
                (void) fprintf(stdout, "%lld:", (offset_t)
                    ((lseek(temp, (off_t)0, SEEK_CUR) - 1) / BLKSIZE));

        if (nflag)
                /* print line number */
                (void) fprintf(stdout, "%lld:", lnum);

        if (nlflag) {
                /* newline at end of line */
                *ptrend = '\n';
                nchars = ptrend - ptr + 1;
        } else {
                /* don't write sentinel \0 */
                nchars = ptrend - ptr;
        }

        (void) fwrite(ptr, 1, nchars, stdout);
        return (0);
}

static void
regerr(int err)
{
        errmsg("grep: RE error %d: ", err);
        switch (err) {
                case 11:
                        err = 0;
                        break;
                case 16:
                        err = 1;
                        break;
                case 25:
                        err = 2;
                        break;
                case 41:
                        err = 3;
```

```
                        break;
                case 42:
                        err = 4;
                        break;
                case 43:
                        err = 5;
                        break;
                case 44:
                        err = 6;
                        break;
                case 45:
                        err = 7;
                        break;
                case 46:
                        err = 8;
                        break;
                case 49:
                        err = 9;
                        break;
                case 50:
                        err = 10;
                        break;
                case 67:
                        err = 11;
                        break;
                default:
                        err = 12;
                        break;
        }

        errmsg("%s\n", gettext(errstr[err]));
        exit(2);
}
```

```
**********************************************************
   14225 Sat Sep 14 23:25:09 2013
new/usr/src/man/man1/grep.1
3546 add support for grep -o option
**********************************************************
'\" te
.\" Copyright (c) 2013 Andrew Stormont.  All rights reserved.
#endif /* ! codereview */
.\" Copyright 2012 Nexenta Systems, Inc. All rights reserved.
.\" Copyright 1989 AT&T
.\" Copyright (c) 2008, Sun Microsystems, Inc.  All Rights Reserved
.\" Portions Copyright (c) 1992, X/Open Company Limited  All Rights Reserved
.\" Sun Microsystems, Inc. gratefully acknowledges The Open Group for permission
.\" http://www.opengroup.org/bookstore/.
.\" The Institute of Electrical and Electronics Engineers and The Open Group, ha
.\"  This notice shall appear on any product containing this material.
.\" The contents of this file are subject to the terms of the Common Development
.\" You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE or http:
.\" When distributing Covered Code, include this CDDL HEADER in each file and in
.TH GREP 1 "May 3, 2013"
.SH NAME
grep \- search a file for a pattern
.SH SYNOPSIS
.LP
.nf
\fB/usr/bin/grep\fR [\fB-c\fR | \fB-l\fR | \fB-q\fR | \fB-o\fR] [\fB-r\fR | \fB-
\fB/usr/bin/grep\fR [\fB-c\fR | \fB-l\fR |\fB-q\fR] [\fB-r\fR | \fB-R\fR] [\fB-b
    \fIlimited-regular-expression\fR [\fIfilename\fR]...
.fi

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

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

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

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

```
If no files are specified, \fBgrep\fR assumes standard input. Normally, each
line found is copied to standard output. The file name is printed before each
line found if there is more than one input file.
.SS "/usr/bin/grep"
.sp
.LP
The \fB/usr/bin/grep\fR utility uses limited regular expressions like those
described on the \fBregexp\fR(5) manual page to match the patterns.
.SS "/usr/xpg4/bin/grep"
.sp
.LP
The options \fB-E\fR and \fB-F\fR affect the way \fB/usr/xpg4/bin/grep\fR
interprets \fIpattern_list\fR. If \fB-E\fR is specified,
\fB/usr/xpg4/bin/grep\fR interprets \fIpattern_list\fR as a full regular
expression (see \fB-E\fR for description).  If \fB-F\fR is specified,
\fBgrep\fR interprets \fIpattern_list\fR as a fixed string. If neither are
specified, \fBgrep\fR interprets \fIpattern_list\fR as a basic regular
expression as described on \fBregex\fR(5) manual page.
.SH OPTIONS
.sp
.LP
The following options are supported for both \fB/usr/bin/grep\fR and
\fB/usr/xpg4/bin/grep\fR:
.sp
.ne 2
.na
\fB\fB-b\fR\fR
.ad
.RS 6n
Precedes each line by the block number on which it was found. This can be
useful in locating block numbers by context (first block is 0).
.RE

.sp
.ne 2
.na
\fB\fB-c\fR\fR
.ad
.RS 6n
Prints only a count of the lines that contain the pattern.
.RE

.sp
.ne 2
.na
\fB\fB-H\fR\fR
.ad
.RS 6n
Precedes each line by the name of the file containing the matching line.
.RE

.sp
.ne 2
.na
\fB\fB-h\fR\fR
.ad
.RS 6n
Prevents the name of the file containing the matching line from being prepended
to that line.  Used when searching multiple files.
.RE

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

```
.RS 6n
Ignores upper/lower case distinction during comparisons.
.RE

.sp
.ne 2
.na
\fB\fB-l\fR\fR
.ad
.RS 6n
Prints only the names of files with matching lines, separated by NEWLINE
characters.  Does not repeat the names of files when the pattern is found more
than once.
.RE

.sp
.ne 2
.na
\fB\fB-n\fR\fR
.ad
.RS 6n
Precedes each line by its line number in the file (first line is 1).
.RE

.sp
.ne 2
.na
\fB\fB-o\fR\fR
.ad
.RS 6n
Print only the matching part of the line.
.RE

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

.sp
.ne 2
.na
\fB\fB-R\fR\fR
.ad
.RS 6n
Read all files under each directory, recursively, following all symbolic links.
.RE

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

.sp
.ne 2
```

```
.na
\fB\fB-s\fR\fR
.ad
.RS 6n
Suppresses error messages about nonexistent or unreadable files.
.RE

.sp
.ne 2
.na
\fB\fB-v\fR\fR
.ad
.RS 6n
Prints all lines except those that contain the pattern.
.RE

.sp
.ne 2
.na
\fB\fB-w\fR\fR
.ad
.RS 6n
Searches for the expression as a word as if surrounded by \fB\e<\fR and
\fB\e>\fR\&.
.RE

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

.sp
.ne 2
.na
\fB\fB-E\fR\fR
.ad
.RS 19n
Matches using full regular expressions. Treats each pattern specified as a full
regular expression. If any entire full regular expression pattern matches an
input line, the line is matched. A null full regular expression matches every
line. Each pattern is interpreted as a full regular expression as described on
the \fBregex\fR(5) manual page, except for \fB\e(\fR and \fB\e)\fR, and
including:
.RS +4
.TP
1.
A full regular expression followed by \fB+\fR that matches one or more
occurrences of the full regular expression.
.RE
.RS +4
.TP
2.
```

```
A full regular expression followed by \fB?\fR that matches 0 or 1
occurrences of the full regular expression.
.RE
.RS +4
.TP
3.
Full regular expressions separated by | or by a new-line that match strings
that are matched by any of the expressions.
.RE
.RS +4
.TP
4.
A full regular expression that is enclosed in parentheses \fB()\fR for
grouping.
.RE
The order of precedence of operators is \fB[\|]\fR, then \fB*\|?\|+\fR, then
concatenation, then | and new-line.
.RE

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

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

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

.SH OPERANDS
.sp
.LP
The following operands are supported:
.sp
.ne 2
.na
\fB\fIfile\fR\fR
.ad
.RS 8n
A path name of a file to be searched for the patterns. If no \fIfile\fR
operands are specified, the standard input is used.
```

```
.RE

.SS "/usr/bin/grep"
.sp
.ne 2
.na
\fB\fIpattern\fR\fR
.ad
.RS 11n
Specifies a pattern to be used during the search for input.
.RE

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

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

.sp
.in +2
.nf
example% \fB/usr/bin/grep -i -n posix text.mm\fR
.fi
.in -2
.sp
```

```
.LP
\fBExample 2 \fRFinding All Empty Lines
.sp
.LP
To find all empty lines in the standard input:

.sp
.in +2
.nf
example% \fB/usr/bin/grep ^$\fR
.fi
.in -2
.sp

.sp
.LP
or

.sp
.in +2
.nf
example% \fB/usr/bin/grep -v .\fR
.fi
.in -2
.sp

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

.sp
.in +2
.nf
example% \fB/usr/xpg4/bin/grep 'abc
def'\fR
example% \fB/usr/xpg4/bin/grep -e 'abc
def'\fR
example% \fB/usr/xpg4/bin/grep -e 'abc' -e 'def'\fR
example% \fB/usr/xpg4/bin/grep -E 'abc|def'\fR
example% \fB/usr/xpg4/bin/grep -E -e 'abc|def'\fR
example% \fB/usr/xpg4/bin/grep -E -e 'abc' -e 'def'\fR
example% \fB/usr/xpg4/bin/grep -E 'abc
def'\fR
example% \fB/usr/xpg4/bin/grep -E -e 'abc
def'\fR
example% \fB/usr/xpg4/bin/grep -F -e 'abc' -e 'def'\fR
example% \fB/usr/xpg4/bin/grep -F 'abc
def'\fR
example% \fB/usr/xpg4/bin/grep -F -e 'abc
def'\fR
.fi
.in -2
.sp

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

.sp
.in +2
```

```
.nf
example% \fB/usr/xpg4/bin/grep -E '^abc$ ^def$'\fR
example% \fB/usr/xpg4/bin/grep -F -x 'abc def'\fR
.fi
.in -2
.sp

.SH ENVIRONMENT VARIABLES
.sp
.LP
See \fBenviron\fR(5) for descriptions of the following environment variables
that affect the execution of \fBgrep\fR: \fBLANG\fR, \fBLC_ALL\fR,
\fBLC_COLLATE\fR, \fBLC_CTYPE\fR, \fBLC_MESSAGES\fR, and \fBNLSPATH\fR.
.SH EXIT STATUS
.sp
.LP
The following exit values are returned:
.sp
.ne 2
.na
\fB\fB0\fR\fR
.ad
.RS 5n
One or more matches were found.
.RE

.sp
.ne 2
.na
\fB\fB1\fR\fR
.ad
.RS 5n
No matches were found.
.RE

.sp
.ne 2
.na
\fB\fB2\fR\fR
.ad
.RS 5n
Syntax errors or inaccessible files (even if matches were found).
.RE

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

.sp
.TS
box;
c | c
l | l .
ATTRIBUTE TYPE  ATTRIBUTE VALUE
_
CSI     Not Enabled
.TE

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

.sp
.TS
```

```
box;
c | c
l | l .
ATTRIBUTE TYPE	ATTRIBUTE VALUE
_
CSI	Enabled
_
Interface Stability	Committed
_
Standard	See \fBstandards\fR(5).
.TE

.SH SEE ALSO
.sp
.LP
\fBegrep\fR(1), \fBfgrep\fR(1), \fBsed\fR(1), \fBsh\fR(1), \fBattributes\fR(5),
\fBenviron\fR(5), \fBlargefile\fR(5), \fBregex\fR(5), \fBregexp\fR(5),
\fBstandards\fR(5)
.SH NOTES
.SS "/usr/bin/grep"
.sp
.LP
Lines are limited only by the size of the available virtual memory. If there is
a line with embedded nulls, \fBgrep\fR only matches up to the first null. If
the line matches, the entire line is printed.
.SS "/usr/xpg4/bin/grep"
.sp
.LP
The results are unspecified if input files contain lines longer than
\fBLINE_MAX\fR bytes or contain binary data. \fBLINE_MAX\fR is defined in
\fB/usr/include/limits.h\fR.
```