NAME

cgrep − search for a pattern using regular expressions under the shortest substring model

SYNOPSIS

cgrep [*option* ...] *pattern* [*filename* ...]
cgrep [**−help**]
cgrep [**−version**]

DESCRIPTION

cgrep searches files for a pattern specified by a regular expression and prints all occurrences of the pattern that do not themselves contain an occurrence of the pattern as a substring. Occurrences may overlap, but no two occurrences will nest. This approach to pattern matching is termed the *shortest substring model*.

In addition, **cgrep** allows a regular expression to be used to define a search universe and reports elements of the search universe that contain (or alternately do not contain) occurrences of the pattern. Useful search universes include email messages, news articles and similar components of structured documents.

The behavior of **cgrep** differs substantially from that of **grep**(1) and other related utilities. Those utilities perform matching only within a line, and only lines containing the pattern may be reported. The approach taken by **cgrep** increases the usefulness of regular expression search, allowing matching across lines and allowing non−text and binary files to be searched. Examples are given toward the end of this man page to illustrate some of the possibilities.

Regular expressions are written in a notation based on that of **egrep**(1) and POSIX 1003.2 (excluding internationalization features) with some additions. These additions include an intersection operator, escape sequences for non−printable characters, and a macro facility.

cgrep begins its execution by reading and processing macros defined in the file *$HOME/.cgreprc*. It then processes its command line arguments, reading and processing in order any macro definition files specified on the command line. Finally, each input file is read and searched, with occurrences of the pattern reported as they are encountered. As each occurrence of the pattern is printed it may be optionally delimited with user-defined start and end tags. If no input file is specified, standard input is read.

OPTIONS

−binary
Do not assume input and output are text files. This option has a minor effect on the matching rules and output format. It is not necessary to specify this option to search binary files. Under the default behavior, superfluous newline characters are stripped from the output, each match is terminated by a newline if one is not already present, and special matching rules are used for the start and end of lines. Under these rules, a ‘^’ matches both the start of the file and the newline character before the start of each line, and a ‘$’ matches the newline character at end of each line and the end of the file. When the **−binary** option is specified, ‘^’ and ‘$’ match only the start and end of file and no stripping or addition of newlines is performed.

−count Print a count of occurrences. If a search universe is specified (using the **-U** option) the number of elements of the universe containing the pattern are reported. If an anti-search universe is specified (using the **-V** option) the number of elements of the universe not containing the pattern are reported.

−c Short form of **−count.** Included for compatibility with older members of the **grep**(1) family.

−defs *filename*
Read and process macro definitions contained in the file.

−help Appearing alone on the command line, prints a short help message describing the options.

−version
Appearing alone on the command line, prints the version of the program.

–insensitive
Ignore upper/lower case distinctions during matching.

–i Short form of **–insensitive**.

–list Reports the names of files containing an occurrence of the pattern. File names are separated by new-lines. Does not repeat the name of a file if more than one occurrence is found. If a search universe is specified (using the **-U** option) the names of files containing an element of the universe containing the pattern are reported. If an anti-search universe is specified (using the **-V** option) the names of files containing an element of the universe not containing the pattern are reported.

–l Short form of **–list**.

–machine
Prints the non–deterministic finite automata (NFA) associated with the pattern. The NFA is represented as a sorted list of state transitions, printed one per line. Each state transition is a triple: the first element of the triple is the "from" state of the transition; the second element is the "to" state of the transition; the final element is the symbol on which the transition is taken.

–mfast *limit*
Set fast match state limit. (Unless you have serious performance concerns or requirements, you probably don't need to worry about this option.) **cgrep** uses two slightly different search algorithms representing different time-space trade-offs: The first ("fast") algorithm uses storage in proportion to the product of the number of states in the NFA and the number of symbols in the input alphabet (currently 256). The second ("slow") algorithm uses storage proportional to the sum of the number of states and the number of symbols. If the number of states is larger than the fast match limit, the slow algorithm is used. Otherwise, the fast algorithm is used. The default value for the fast match limit is 2048 states. If a value of 0 is specified the fast algorithm will always be used. If a value of 1 is specified the slow algorithm will always be used.

–range Print pairs representing the start and end positions of matching occurrences in the input. If standard input is not being read, each range will be preceded by the name of the file in which the match occurred.

–silent Enable silent mode. Suppresses the printing of error messages that occur during the processing of files.

–s Short form of **–silent**.

–tag *start-tag end-tag*
Start and end delimiters for tagging occurrences. An '@' appearing in a tag is replaced by the name of the file where the occurrence was found. Any escape sequence valid in a regular expression is valid in a tag.

-U *regular-expression*
Define a search universe. **cgrep** will report elements of the search universe that contain occurrences of the pattern.

-V *regular-expression*
Define an anti-search universe. **cgrep** will report elements of the search universe that do not contain occurrences of the pattern.

REGULAR EXPRESSIONS

The regular expression syntax is based on that of POSIX 1003.2 extended regular expressions (excluding internationalization features). This regular expression syntax is similar to that of **egrep**(1). More explicitly, **cgrep** supports POSIX 1003.2 *character classes*, but does not support POSIX 1003.2 multi–character collating symbols or equivalence class expressions. Back references, the '\(' and '\)' available in **ed**(1), **grep**(1), and POSIX 1003.2 basic regular expressions, are also not supported.

In addition to the standard POSIX 1003.2 operators, we accept '&' for the intersection of two regular expressions. The precedence of the intersection operator is the same as that of union ('|'). The union and intersect operators associate left to right.

The characters ‘<’ and ‘>’ may be used to match the beginning and end of file respectively.

We make one addition to the character classes defined by POSIX 1003.2: Within a bracket expression, the sequence ‘[:print:]’ matches any printable character. Character class membership is based on the **ctype**(3) macros.

Escape sequences for non−printable characters follow the syntax of ANSI C, including the sequences for hexadecimal and octal constants. Escape sequences undefined by ANSI C represent the literal character following the '\'. In particular, an escape consisting of a ‘\’ followed by any punctuation character may be used to represent the literal punctuation mark, avoiding any special meaning of the character.

Support for macros is provided. Macros calls come in two flavors: *fast* and *tedious*. A fast call consists of an ‘@’ character followed by an single alphabetic character. A tedious macro call has the form:

[@*name*(*parameter0*, *parameter1*, ...)]

where each of the up to 9 parameters is a regular expression. If the macro requires no parameters, the bracket-enclosed parameter list is omitted completely. Be careful not to put any extra whitespace in the parameter list, this extra whitespace will be counted as part of the parameter.

MACRO DEFINITIONS

Fast and tedious macros are defined in the same way. Any un−parameterized, single−letter macro is automatically usable as either a fast macro or a tedious macro.

An un-parameterized macro definition has the form:

name=*regular-expression*

and parameterized macro definition has the form:

name#*n*=*regular-expression*

where the number of parameters is indicated by a single digit following the ‘#’ character. Within the body of a parameterized macro, the actual parameters may be referenced as ‘#1’ through ‘#9’. A macro name must start with a alphabetic character, and may include only alphanumeric characters and the character ‘_’. Be careful not put any extra whitespace after the '='; this whitespace counts as part of the regular expression.

EXAMPLES

One use of **cgrep** is to find occurrences of a phase broken across two or more lines. For each appearance of the country's name, the command

```
cgrep '^.*United[[:space:]]*States.*$' constitution.txt
```

will print the lines of text that contain it.

The command

```
cgrep -list 'the\nthe' *.txt
```

checks for a typing error that's hard to spot visually and prints the names of the files that contain it.

The command

```
cgrep -insensitive -U '/\*.*\*/' POSIX cgrep.c
```

prints all the comments in the C source file *cgrep.c*, that contain the the string ‘‘posix’’ in any combination of lower and upper case letters (under some mild assumptions).

The command

```
cgrep '[^[:print:]][[:print:]]{4,}[\n\0]' a.out
```

reports strings of four or more printable characters ending in a newline or null character that appear in the executable file *a.out*. Each match is printed on a separate line. If the **–binary** flag were specified, the resulting matches would be run together without separating newlines. Each match is started by an unprintable character and may contain superfluous null characters. The output could be piped to

```
cgrep -binary '[[:print:]\n]'
```

to strip these unprintable characters (or the **tr**(1) command could be used for the same purpose).

As a final example, **cgrep** may be used to to search a mail file and extract mail based on patterns in the sender or subject lines, or in other parts of the header and body. Standard macros for handling mail may be defined in the *$HOME/.cgreprc* file:

```
Mail=^From .*(^From |>)
From#1=^From:[^$]*#1
Re#1=^Subject:[^$]*#1
```

The command

```
cgrep -U '[@Mail]' '(.*[@From([Cc]owan)].*)&(.*[@Re(brewpubs)].*)' mbox
```

would then extract all mail messages in the file *mbox* that are from Cowan and are on the subject of brewpubs. It’s then necessary to pipe the output through

```
sed '/^From $/d'
```

or equivalently

```
cgrep -V '^.*$' '^From $'
```

to strip out the extra characters needed to detect the end of each mail message and create a validly formatted mail file.

AUTHOR

Charlie Clarke (claclark@plg.uwaterloo.ca)

SEE ALSO

egrep(1), **grep**(1)

POSIX 1003.2, section 2.8 (Regular Expression Notation).

Charles L. A. Clarke and Gordon V. Cormack. *On the use of Regular Expressions for Searching Text.* University of Waterloo Computer Science Department Technical Report number CS-95-07, University of Waterloo, Waterloo, Ontario N2L 3G7, Canada. February 1995. ftp://plg.uwaterloo.ca/pub/mt/TechReports/CS-95-07/regexp.ps

FILES

$HOME/.cgreprc start−up macro definition file

BUGS

Because of limits on internal buffering, matches longer than one megabyte in length may not be reported when reading from standard input.

The syntax for macros is ugly. An undefined macro is reported as a syntax error.

This man page needs to be extended with a complete and precise description of the regular expression format.

The software is an alpha release. Report bugs to mt@plg.uwaterloo.ca.