

**NAME**

mg – multi-line grep

**SYNOPSIS**

**mg** [ **options** ] *pattern* [ *file* ]

**DESCRIPTION**

*Mg* searches the specified pattern from files or standard input and prints lines which contain the search pattern. It has almost the same function as the Unix command *grep*(1) but is distinguished from *grep* because the matching is done across the line boundaries.

For example, to find a sentence “internet control message protocol” from many RFC texts, you can say

```
mg -i 'internet control message protocol' *.txt
```

Match will occur for sequence of words ‘internet’, ‘control’, ‘message’ and ‘protocol’ separated by any number of whitespace characters including newline and null string (–w option avoids null matching).

**[COMPRESSED FILE SEARCH]** If the file has ‘.Z’ or ‘.gz’ suffix, the data is automatically uncompressed on the fly before search. Use –Z option to suppress this operation. Because this mechanism is supported in generic framework, any kind of suffix and preprocessor pair can be defined by –z option, which also allows to give default input filter. Output filter is specified by –X option.

**[EMPHASIZING and BLOCK SEARCH]** To emphasize the matched part of text, –Q, –q, –u and –b options are supported. This is useful especially used with block search options. Option –c gives the number of surrounding lines to be shown along with matched line. Option –o prints the paragraph containing matched text. Containing page is with –g, and entire text is printed with –a option.

**[RECURSIVE SEARCH]** *Mg* supports recursive directory search. Use –R option to enable it. This option can be combined with –T (text only search), –B (binary data search), –P/V (file name pattern), –D (descending level), –F (symbolic link control). Option –dcd gives you ongoing status.

**[JAPANESE STRING SEARCH]** *Mg* is also useful to find a word from Japanese text because Japanese words are not separated by whitespaces, and newline character can be inserted at any place of the text. As a matter of fact, *mg* was made for Japanese string search originally. Any Japanese codes including JIS, Shift-JIS, EUC can be handled hopefully, but using JIS code disables some features.

User should be aware that *mg* doesn’t recognize character boundaries even if the search string is described in any of Japanese code. So some 2-byte character code may match with 2nd byte of one character and 1st byte of next one. Especially when it is encoded in JIS, it may match some sequence of ASCII string (it may be meaningless string in uencode or base64). From my experience, this is not a big problem when searching some meaningful strings, but you may get in trouble when looking for short string especially single character.

Use –j option to specify search string code. Use –z option if you want convert file code before search. Option –y enables to find JIS X 0208 representation of ASCII characters (only EUC).

**[ARCHIVE SEARCH]** If the file is archived file by *ar*(1) or *tar*(1) command, each file in the archive will be searched individually. The file name is shown like “(archive)file” for ar format and “[archive]file” for tar format. This function works for compressed file of course. Use –A option if you want to search from entire archived file.

If the file has ‘.zip’, ‘.zoo’ or ‘.lzh’ suffix, the file is treated as *zip*, *zoo* and *lha* archive respectively and each file contained in the archive will be the subject of search. Option –A disables this feature too. File name is shown as “{archive}file”.

**[BUFFERING POLICY]** *Mg* reads some amount of data at once, and the last portion of the chunk will be searched again with next chunk of data. Default size of data chunk varies depending of perl version; using perl4, it’s 30k and 512k on perl5. Search-again-data size is 2k for both. So if the matched segment size is more than 2K bytes, it may be truncated. This truncation happens only when the file size is more than data chunk size, of course. You can use –W option to read whole file contents at once. But this slows down the speed of execution for large file. Maximum read and search again size can be specified by –G option.

**[PERFORMANCE NOTICE]** This version of *mg* is free from using `$'`, `$&` and `$'` only when executed by *perl5*. Using these variables makes the program much slower when processing large chunk of data. Using *perl5* doubles the performance when getting many matches from large file.

## ENVIRONMENT

Environment variable `MGOPTS` is used as a default options.

## OPTIONS

### GREP COMPATIBLE OPTIONS:

- `-i` Ignore case.
- `-l` List filename only.
- `-n` Show line number.
- `-h` Do not display filenames even if multiple filenames are specified by command line.

### SLIGHTLY DIFFERENT OPTIONS:

- `-w` Word sensitive. Match occurs only when both beginning and end of pattern is on word boundary. Also space characters in the pattern doesn't match to the null string.
- `-v pattern`  
Skip the line if matched with pattern. Don't print the matched line if it matched the pattern specified with this option. This option doesn't have any effect when used with `-a` or `-l` option.

### OTHER OPTIONS:

- `-d flags`  
Display informations. Various kind of debug, diagnostic, monitor information can be display by giving appropriate flag to `-d` option.

f: processing file name  
d: processing directory name  
c: count of processing files  
s: statistic information  
m: misc debug information

p: run 'ps' command before termination (on Unix)

You may want to use "`-dcd`" option to monitor what is going on during recursive search.

- `-N` Print byte offset of matched string. If multiple matching occurred in one line, only the first matching offset is printed. Use `-2` option to avoid it.
- `-e` Use the pattern as a regular expression in *perl*(1) but space is treated specially. With this option, you can use *mg* like *egrep*(1) like this:

```
mg -e 'foo bar|goo car|hoo dar' ...
```

See *perl*(1) for detail of regular expression. Slash characters (`/`) in expression don't have to be escaped.

Option `-w` puts `\b`'s at the beginning and the end of the pattern. So the pattern "`foo|bar`" becomes "`\bfoo|bar\b`". You probably want to use "`foo\b|\bbar`" instead.

- `-E` Use the pattern as regular expression in *perl* completely.
- `-r` Specify restriction pattern. A file becomes a subject for search only when the pattern specified by `-r` option is found in the file. Next two examples are equivalent.

```
mg -e ^Subject: 'mg -le "^From: lwall" *'
```

```
mg -er '^From: lwall' ^Subject: *
```

File will be swallowed at one time even if `-W` option is not supplied.

`-c n[,n]`

Print *n*-lines before/after matched line. Default *n* is 1, which means only one line is displayed. *N* is number of newlines surrounding matched pattern. If “`-c 3`” options is supplied, two lines before and after matched line will be displayed together. You can see only after two lines by “`-c 1,3`” option.

Option `-c0` displays matched string only. Next example is almost equivalent to `strings(1)` command with `-oa` option.

```
mg -NEc0 '[\t(040-176){4,}' file
```

`-o` Print the paragraph which contains the pattern. Each paragraph is delimited by two successive newline character by default. Be aware that an empty line is not paragraph delimiter if which contains space characters. Example:

```
mg -nQo 'setuid script' /usr/man/cat1/perl.1
```

```
mg -o sockaddr /usr/include/sys/socket.h
```

If `-c` option is also supplied and there are more lines in the paragraph, continuous mark is displayed.

`-O string`

Specify paragraph delimiter string as well as activate the paragraph mode. The contents of string is evaluated as is inside double-quote not as a regular expression. For example, you can get lines between troff macros like this:

```
mg -QoO "\n." 'setuid script' /usr/man/man1/perl.1
```

`-g lines`

Page mode. Provided number of lines for each pages, pages which contains the pattern will be displayed. For example, you can get pages which contain some strings from nroff’ed text like this:

```
mg -Q -g 66 'setuid script' /usr/man/cat1/perl.1
```

You can’t use `-c` with this option. Formfeed is not supported currently.

`-C chars`

Continuous character. If you want search sentence continued by other than white space characters, continuous characters can be set by this options. For example, next command finds a sentence even if it is quoted by ‘>’ or ‘|’ mark.

```
mg -C '>|' 'ninja shogun fujiyama' 'mhpah all'
```

To search a pattern in C style comments:

```
mg -C '/*' 'setuid scripts' perl.c
```

Note that continuous characters don’t have to be found only top of the string. So “foo bar” matches a string “foo>>bar” on the previous example.

- u Underline matched string. Makes a matched string underlined by precede each character by “\_H”.
- b Make bold matched string. Makes a matched string overstruck like “f^Hfo^Hoo^Ho”.
- q Quote matched string by like “>>matched<<”. This options is disabled when searching JIS file.
- Q Use a stand-out feature of the terminal to quote the matched string (not for JIS). This option is ignored when the standard output is not a terminal. -q and -Q are useful for long line. Try

```
echo $path | mg -Q mh
```

- a Print all contents of the file. This option makes sense only if used with options like -q, -Q, -u, -b, otherwise it behaves like *cat(1)*. Filenames and lines are not printed with this option.
- s When multiple files are specified, each matched line is preceded by “filename:” string. With this option, a newline character is inserted between the filename and matched line. This option is useful when the filenames are very long.
- 2 Usually only one line is displayed even if multiple matching occurs for the same line. With this option, each match will be displayed in different line.
- R Search recursively. Only files specified by command line arguments are searched by default. When invoked with this option and arguments contain a directory, it is searched recursively. Usually used with -P or -T option.

*-D level*

Descending directory level. This option specifies how many levels of directory is to be searched from top directory.

*-P pattern*

Search file pattern. When directories are searched recursively, only files which match the ‘pattern’ are searched. A ‘pattern’ is specified in wildcard format same as shell and ‘|’ character can be used for alternative in addition. For example, you can find a string “foobar” from all C source files and makefiles like this:

```
mg -RP '*.[Mm]akefile' foobar /usr/src
```

*-V pattern*

Exception file pattern. This is a counterpart of -P. Only files which DO NOT match the pattern will be searched.

```
mg -RV '*.[oas]' foobar /usr/src
```

Note that the -V option is also applied to a directory name while -P option has an effect only for a file. This means you can specify a directory name to skip, but can't specify a directory name to search.

- F Follow symbolic link of a directory. Doesn't follow by default.
- T Search text file only. Binary file is skipped with this option. Decision is made by original method, not by perl operator -T, so that Japanese SJIS and EUC text is taken as text file. See -B option.
- B Find string from binary file. Only printable characters surrounding matched pattern are printed. If both -T and -B are supplied, text file is searched in normal mode and binary file is searched in binary mode. Next example is almost the same as *what(1)*.

```
mg -Bc0,1 '@(#)' /bin/awk
```

- L Print formfeed between each matchings. Print the formfeed character before each matched line. This options is useful when used with `-c` option and piped to pager command.
- S Get filenames from standard input. Read standard input and use each line as a filename for searching. You can feed the output from other command like `find(1)` for `mg` with this option. Next example searches string from files modified within 7 days:

```
find . -mtime -7 -print | mg -S pattern
```

Next example find the files from the newest first order:

```
ls -t | mg -S pattern
```

- m Print matched line only when the pattern is across the line.
- M Print matched line only when multiple matching occurred for the same line.
- f *file* Specify the file which contains search pattern. When file contains multiple lines, patterns on each lines are search in OR context.
- p Specify search pattern. You don't have to use this option explicitly because the first argument after options will be treated as a pattern.
- A Disables archive mode search (ar, tar, zip, zoo).
- Z Disables automatic uncompress, gunzip.
- J *string* Convert newline character(s) found in matched string to specified *string*. Using `-J` with `-c0` option, you can collect searching sentence list in one per line form. This is almost useless for English text but sometimes useful for Japanese text. For example next command prints the list of KATAKANA words used in the Shift-JIS texts.

```
set kana='\203[\100-\226]|\201\133'
set p="($kana)($kana\s)*"
mg -Ec0 -J "" "$p" files | sort | uniq -c
```

Note that this command is confused when 2nd byte and 1st byte of next character matches KATAKANA pattern.

- 0*digits* Specifies the record separator as an octal number. It is almost same as perl option. But unlike perl, only the first record is read for search. Like perl, `-00` means paragraph mode. Actually I added this option only to use `-00` to search from mail and news header portion. When reading from archived file, `mg` emulates perl's `$/` behaviour.
  - W Slurp whole file at once.
  - G *maxreadsize*[,*keepsize*] Specify maximum read size and keep buffer size for next read. Default values for these sizes are 30K and 2K bytes. `Mg` tries to read a file upto maximum read size at a same time. Last part of the buffer, specified by keep buffer size, is not thrown away but will be used as a subject for search with next buffer. In arguments, you can use B, K, and M for block (512), kilo (1024) and mega (1024 \* 1024) respectively. Next example sets maximum read size for 100K and keep buffer size for 10K bytes.
- ```
mg -iBG 100K,10K unix /vmunix
```
- 1 Print first match only. This option doesn't work well with `-a` option.

- j *code* If you have to use different Japanese codes for search pattern and target file, target code can be specified by -j option. The code should be one of 'jis', 'sjis' and 'euc'. Perl library 'jcode.pl' has to be installed to use this option.
- y When option -y is specified, JIS X 0208 representation of ASCII string is also searched even if the pattern is supplied in ASCII. Currently only EUC encoding is supported, and '-j euc' option is required to enable this option. If the pattern is specified in JIS X 0208 string, only JIS X 0208 character is searched.
- z *filter* (or *EXP:filter:EXP:filter:...*)  
You can specify filter command which is applied to each files before search. If filter information include multiple fields separated by colons, first field is perl expression to check the filename saved in variable \$\_. These expression and command list can be repeated. If only one filter command is specified, it is applied to all files. Examples:

```
mg -z 'dd conv=ascii' string spoiled_files
mg -z '\.tar$/:tar tvf -' pattern *
```

Filters for compressed and gzipped file is set by default unless -Z option is given. Default action is:

```
mg -z 's\.\Z$//:zcat:s\.\g?z$//:gunzip -c'
```

- x *exp* You can specify the any Perl expression to preprocess input data. Some subroutine will be available for this purpose but currently only "&mime" is prepared. If "require" operator is included in *exp*, it is executed only once. So you can include your special perl library and use the subroutine defined in it.

&mime Subroutine "mime" decodes encoded string based on RFC1342 MIME header encoding but current implementation handles only ISO-2022-JP encoding. Example:

```
mg -x '&mime' -00 From ~/Mail/inbox/*
```

Note that, this process is done just before a search for the output from input filter if -z option is specified. So in the above example, MIME encoded string is converted into ISO-2022-JP even if the input filter was specified to convert the all data into EUC.

## APPENDIX

You may want to use *mg(1)* instead of *grep(1)* from GNU emacs. In that case please add following program segment in your .emacs file.

```
(defun mg (command)
  "Run mg instead of grep."
  (interactive "sRun mg (with args): ")
  (require 'compile)
  (compile1 (concat "mg -n " command " /dev/null")
            "No more mg hits" "mg"))
```

If you are using version 19, use this.

```
(defun mg (command-args)
  "Run mg instead of grep."
  (require 'compile)
  (interactive
   (list (read-from-minibuffer "Run mg (like this): "
                               "mg -n " nil nil 'grep-history)))
  (compile-internal (concat command-args " " null-filename)
                    "No more mg hits" "mg"
                    ;; Give it a simpler regexp to match.
                    nil grep-regexp-alist))
```

You have to visit uninterested line by (next-error) when surrounding lines are displayed by `-c` or `-o` option. Use `-s` option to avoid this.

#### AUTHOR

Kazumasa Utashiro <utashiro@ij.ad.jp>  
 Internet Initiative Japan Inc.  
 3-13 Kanda Nishiki-cho, Chiyoda-ku, Tokyo 101-0054, Japan

#### SEE ALSO

grep(1), perl(1)

#### BUGS

When using perl older than version 5.6, actual pattern is enclosed by parentheses, and it confuses the order of subexpressions if it contains back-references. The order is fixed automatically but you may have some problem for certain patterns. Use `-dm` option to check the actual pattern for search when you doubt the behavior of this command.

Hyphenation is not supported.

No capability to find words separated by nroff footer and header.

Very long JIS code pattern may not be processed because of a limitation of regular expression engine.

Not enough space for new option (try undocumented option `-U...`, oops, documented here :-).

#### LICENSE

Copyright (c) 1994-2001 Kazumasa Utashiro

Use and redistribution for ANY PURPOSE are granted as long as all copyright notices are retained. Redistribution with modification is allowed provided that you make your modified version obviously distinguishable from the original one. THIS SOFTWARE IS PROVIDED BY THE AUTHOR "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES ARE DISCLAIMED.