NAME

m4 - macro language processor

SYNOPSIS

m4 [**-EGgPs**] [**-D***name*[=*value*]] [**-d** [[+-]*flags*]] [**-I** *dirname*] [**-o** *filename*] [**-t** *macro*] [**-U***name*] [*file ...*]

DESCRIPTION

The **m4** utility is a macro processor that can be used as a front end to any language (e.g., C, ratfor, fortran, lex, and yacc). If no input files are given, **m4** reads from the standard input, otherwise files specified on the command line are processed in the given order. Input files can be regular files, files in the m4 include paths, or a single dash (‘-’), denoting standard input. **m4** writes the processed text to the standard output, unless told otherwise.

Macro calls have the form name(argument1[, argument2, ..., argumentN]).

There cannot be any space following the macro name and the open parenthesis (‘(’). If the macro name is not followed by an open parenthesis it is processed with no arguments.

Macro names consist of a leading alphabetic or underscore possibly followed by alphanumeric or underscore characters, e.g., valid macro names match the pattern "[a-zA-Z_][a-zA-Z0-9_]*".

In arguments to macros, leading unquoted space, tab, and newline (‘\n’) characters are ignored. To quote strings, use left and right single quotes (e.g., ‘ this is a string with a leading space’). You can change the quote characters with the **changequote** built-in macro.

Most built-ins do not make any sense without arguments, and hence are not recognized as special when not followed by an open parenthesis.

The options are as follows:

-D*name*[=*value*], **--define**=*name*[=*value*]
: Define the symbol *name* to have some value (or NULL).

-d [[+|-]*flags*], **--debug**=[[+|-]*flags*]
: Set or unset trace flags. The trace flags are as follows:

a	print macro arguments.
c	print macro expansion over several lines.

e print result of macro expansion.

f print filename location.

l print line number.

q quote arguments and expansion with the current quotes.

t start with all macros traced.

x number macro expansions.

V turn on all options.

If "+" or "-" is used, the specified flags are added to or removed from the set of active trace flags, respectively; otherwise, the specified flags replace the set of active trace flags.

Specifying this option without an argument is equivalent to specifying it with the argument "aeq".

By default, trace is set to "eq".

-E, **--fatal-warnings**
Set warnings to be fatal. When a single **-E** flag is specified, if warnings are issued, execution continues but **m4** will exit with a non-zero exit status. When multiple **-E** flags are specified, execution will halt upon issuing the first warning and **m4** will exit with a non-zero exit status. This behavior matches GNU m4 1.4.9 and later.

-G, **--traditional**
Disable GNU compatibility mode (see **-g** below).

-g, **--gnu**
Enable GNU compatibility mode. In this mode, **translit** handles simple character ranges (e.g., ‘a-z’), regular expressions mimic Emacs behavior, multiple **m4wrap** calls are handled as a stack, the number of diversions is unlimited, empty names for macro definitions are allowed, **undivert** can be used to include files, and **eval** understands ‘0rbase:value’ numbers.

-I *dirname*, **--include**=*dirname*
Add directory *dirname* to the include path.

-o *filename*, **--error-output**=*filename*
: Send trace output to *filename*.

-P, **--prefix-builtins**
: Prefix all built-in macros with ‘m4_’. For example, instead of writing **define**, use **m4_define**.

-s, **--synclines**
: Output line synchronization directives, suitable for cpp(1).

-t *macro*, **--trace**=*macro*
: Turn tracing on for *macro*.

-U*name*, **--undefine**=*name*
: Undefine the symbol *name*.

SYNTAX

m4 provides the following built-in macros. They may be redefined, losing their original meaning. Return values are null unless otherwise stated.

builtin(*name*)
: Calls a built-in by its *name*, overriding possible redefinitions.

changecom(*startcomment, endcomment*)
: Changes the start comment and end comment sequences. Comment sequences may be up to five characters long. The default values are the hash sign and the newline character.

```
# This is a comment
```

With no arguments, comments are turned off. With one single argument, the end comment sequence is set to the newline character.

changequote(*beginquote*, *endquote*)
: Defines the open quote and close quote sequences. Quote sequences may be up to five characters long. The default values are the backquote character and the quote character.

```
‘Here is a quoted string’
```

With no arguments, the default quotes are restored. With one single argument, the close quote sequence is set to the newline character.

decr(*arg*) Decrements the argument *arg* by 1. The argument *arg* must be a valid numeric string.

define(*name*, *value*)
Define a new macro named by the first argument *name* to have the value of the second argument *value*. Each occurrence of ‘$n’ (where *n* is 0 through 9) is replaced by the *n*’th argument. ‘$0’ is the name of the calling macro. Undefined arguments are replaced by a null string. ‘$#’ is replaced by the number of arguments; ‘$*’ is replaced by all arguments comma separated; ‘$@’ is the same as ‘$*’ but all arguments are quoted against further expansion.

defn(*name*, *...*)
Returns the quoted definition for each argument. This can be used to rename macro definitions (even for built-in macros).

divert(*num*) There are 10 output queues (numbered 0-9). At the end of processing **m4** concatenates all the queues in numerical order to produce the final output. Initially the output queue is 0. The divert macro allows you to select a new output queue (an invalid argument passed to divert causes output to be discarded).

divnum Returns the current output queue number.

dnl Discard input characters up to and including the next newline.

dumpdef(*name*, *...*)
Prints the names and definitions for the named items, or for everything if no arguments are passed.

errprint(*msg*)
Prints the first argument on the standard error output stream.

esyscmd(*cmd*)
Passes its first argument to a shell and returns the shell’s standard output. Note that the shell shares its standard input and standard error with **m4**.

eval(*expr[,radix[,minimum]]*)
Computes the first argument as an arithmetic expression using 32-bit arithmetic. Operators are the standard C ternary, arithmetic, logical, shift, relational, bitwise, and parentheses operators. You can specify octal, decimal, and hexadecimal numbers as in C. The optional second argument *radix* specifies the radix for the result and the optional third argument *minimum* specifies the minimum number of digits in the result.

expr(*expr*) This is an alias for **eval**.

format(*formatstring*, *arg1*, *...*)
Returns *formatstring* with escape sequences substituted with *arg1* and following arguments, in a way similar to printf(3). This built-in is only available in GNU-m4 compatibility mode, and the only parameters implemented are there for autoconf compatibility: left-padding flag, an optional field width, a maximum field width, *-specified field widths, and the %s and %c data type.

ifdef(*name*, *yes*, *no*)
If the macro named by the first argument is defined then return the second argument, otherwise the third. If there is no third argument, the value is NULL. The word "unix" is predefined.

ifelse(*a*, *b*, *yes*, *...*)
If the first argument *a* matches the second argument *b* then **ifelse**() returns the third argument *yes*. If the match fails the three arguments are discarded and the next three arguments are used until there is zero or one arguments left, either this last argument or NULL is returned if no other matches were found.

include(*name*)
Returns the contents of the file specified in the first argument. If the file is not found as is, look through the include path: first the directories specified with **-I** on the command line, then the environment variable M4PATH, as a colon-separated list of directories. Include aborts with an error message if the file cannot be included.

incr(*arg*) Increments the argument by 1. The argument must be a valid numeric string.

index(*string*, *substring*)
Returns the index of the second argument in the first argument (e.g., **index(the quick brown fox jumped, fox)** returns 16). If the second argument is not found index returns -1.

indir(*macro*, *arg1*, *...*)
Indirectly calls the macro whose name is passed as the first argument, with the remaining arguments passed as first, ... arguments.

len(*arg*) Returns the number of characters in the first argument. Extra arguments are ignored.

m4exit(*code*)
Immediately exits with the return value specified by the first argument, 0 if none.

m4wrap(*todo*)
Allows you to define what happens at the final EOF, usually for cleanup purposes (e.g., **m4wrap("cleanup(tempfile)")** causes the macro cleanup to be invoked after all other processing is done).

Multiple calls to **m4wrap**() get inserted in sequence at the final EOF.

maketemp(*template*)
Like **mkstemp**.

mkstemp(*template*)
Invokes mkstemp(3) on the first argument, and returns the modified string. This can be used to create unique temporary file names.

paste(*file*) Includes the contents of the file specified by the first argument without any macro processing. Aborts with an error message if the file cannot be included.

patsubst(*string*, *regexp*, *replacement*)
Substitutes a regular expression in a string with a replacement string. Usual substitution patterns apply: an ampersand (‘&’) is replaced by the string matching the regular expression. The string ‘\#’, where ‘#’ is a digit, is replaced by the corresponding back-reference.

popdef(*arg*, *...*)
Restores the **pushdef**ed definition for each argument.

pushdef(*macro*, *def*)
Takes the same arguments as **define**, but it saves the definition on a stack for later retrieval by **popdef**().

regexp(*string*, *regexp*, *replacement*)
Finds a regular expression in a string. If no further arguments are given, it returns the first match position or -1 if no match. If a third argument is provided, it returns the replacement string, with sub-patterns replaced.

shift(*arg1*, *...*)
Returns all but the first argument, the remaining arguments are quoted and pushed back with commas in between. The quoting nullifies the effect of the extra scan that will subsequently be performed.

sinclude(*file*)
Similar to **include**, except it ignores any errors.

spaste(*file*) Similar to **paste**(), except it ignores any errors.

substr(*string*, *offset*, *length*)
Returns a substring of the first argument starting at the offset specified by the second argument and the length specified by the third argument. If no third argument is present it returns the rest of the string.

syscmd(*cmd*)
Passes the first argument to the shell. Nothing is returned.

sysval Returns the return value from the last **syscmd**.

traceon(*arg*, ...)
Enables tracing of macro expansions for the given arguments, or for all macros if no argument is given.

traceoff(*arg*, ...)
Disables tracing of macro expansions for the given arguments, or for all macros if no argument is given.

translit(*string*, *mapfrom*, *mapto*)
Transliterate the characters in the first argument from the set given by the second argument to the set given by the third. You cannot use tr(1) style abbreviations.

undefine(*name1*, ...)
Removes the definition for the macros specified by its arguments.

undivert(*arg*, ...)
Flushes the named output queues (or all queues if no arguments).

unix A pre-defined macro for testing the OS platform.

__line__ Returns the current file's line number.

__file__ Returns the current file's name.

EXIT STATUS

The **m4** utility exits 0 on success, and >0 if an error occurs.

But note that the **m4exit** macro can modify the exit status, as can the **-E** flag.

SEE ALSO

B. W. Kernighan and D. M. Ritchie, *The M4 Macro Processor, AT&T Bell Laboratories*, Computing Science Technical Report, 59, July 1977.

STANDARDS

The **m4** utility is compliant with the IEEE Std 1003.1-2008 ("POSIX.1") specification.

The flags [**-dEGgIPot**] and the macros **builtin**, **esyscmd**, **expr**, **format**, **indir**, **paste**, **patsubst**, **regexp**, **spaste**, **unix**, **__line__**, and **__file__** are extensions to that specification.

maketemp is not supposed to be a synonym for **mkstemp**, but instead to be an insecure temporary file name creation function. It is marked by IEEE Std 1003.1-2008 ("POSIX.1") as being obsolescent and should not be used if portability is a concern.

The output format of **traceon** and **dumpdef** are not specified in any standard, are likely to change and should not be relied upon. The current format of tracing is closely modelled on **gnu-m4**, to allow **autoconf** to work.

The built-ins **pushdef** and **popdef** handle macro definitions as a stack. However, **define** interacts with the stack in an undefined way. In this implementation, **define** replaces the top-most definition only. Other implementations may erase all definitions on the stack instead.

All built-ins do expand without arguments in many other **m4**.

Many other **m4** have dire size limitations with respect to buffer sizes.

AUTHORS

Ozan Yigit <*oz@sis.yorku.ca*> and Richard A. O'Keefe <*ok@goanna.cs.rmit.OZ.AU*>.

GNU-m4 compatibility extensions by Marc Espie <*espie@cvs.openbsd.org*>.