NAME

m4 – macro processor

SYNOPSIS

/usr/ccs/bin/m4 [**–e**] [**–s**] [**–B** *int*] [**–H** *int*] [**–S** *int*] [**–T** *int*]
[**–D***name* [=*val*]] . . . [**–U** *name*] . . . [*file* . . .]

/usr/xpg4/bin/m4 [**–e**] [**–s**] [**–B** *int*] [**–H** *int*] [**–S** *int*] [**–T** *int*]
[**–D***name* [=*val*]] . . . [**–U** *name*] . . . [*file* . . .]

AVAILABILITY

/usr/ccs/bin/m4 SUNWcsu

/usr/xpg4/bin/m4 SUNWxcu4

DESCRIPTION

The **m4** command is a macro processor intended as a front end for C, assembler, and other languages. Each of the argument files is processed in order; if there are no files, or if a file is –, the standard input is read. The processed text is written on the standard output.

Macro Syntax

Macro calls have the form:

*name***(***arg1*,*arg2*, . . ., *argn***)**

The **(** must immediately follow the name of the macro. If the name of a defined macro is not followed by a **(**, it is deemed to be a call of that macro with no arguments. Potential macro names consist of alphanumeric characters and underscore (_), where the first character is not a digit.

Leading unquoted blanks, TABs, and NEWLINEs are ignored while collecting arguments. Left and right single quotes are used to quote strings. The value of a quoted string is the string stripped of the quotes.

Macro Processing

When a macro name is recognized, its arguments are collected by searching for a matching right parenthesis. If fewer arguments are supplied than are in the macro definition, the trailing arguments are taken to be **NULL**. Macro evaluation proceeds normally during the collection of the arguments, and any commas or right parentheses that happen to turn up within the value of a nested call are as effective as those in the original input text. After argument collection, the value of the macro is pushed back onto the input stream and rescanned.

OPTIONS

The options and their effects are as follows:

–e Operate interactively. Interrupts are ignored and the output is unbuffered.

–s Enable line sync output for the C preprocessor (#line . . .)

–B *int* Change the size of the push-back and argument collection buffers from the default of **4,096**.

–H *int* Change the size of the symbol table hash array from the default of **199**. The size should be prime.

–S *int* Change the size of the call stack from the default of **100** slots. Macros take three slots, and non-macro arguments take one.

–T *int* Change the size of the token buffer from the default of **512** bytes.

To be effective, the above flags must appear before any file names and before any **–D** or **–U** flags:

–D *name*[**=***val*] Defines *name* to *val* or to **NULL** in *val*'s absence.

–U *name* Undefines *name*.

OPERANDS

The following operand is supported:

file A path name of a text file to be processed. If no *file* is given, or if it is –, the standard input is read.

USAGE

m4 makes available the following built-in macros. These macros may be redefined, but once this is done the original meaning is lost. Their values are **NULL** unless otherwise stated.

changequote Change quote symbols to the first and second arguments. The symbols may be up to five characters long. **changequote** without arguments restores the original values (that is, \` ´).

changecom Change left and right comment markers from the default **#** and NEWLINE. With no arguments, the comment mechanism is effectively disabled. With one argument, the left marker becomes the argument and the right marker becomes NEWLINE. With two arguments, both markers are affected. Comment markers may be up to five characters long.

decr Returns the value of its argument decremented by 1.

define The second argument is installed as the value of the macro whose name is the first argument. Each occurrence of **$***n* in the replacement text, where *n* is a digit, is replaced by the *n*-th argument. Argument 0 is the name of the macro; missing arguments are replaced by the null string; **$#** is replaced by the number of arguments; **$*** is replaced by a list of all the arguments separated by commas; **$@** is like **$***, but each argument is quoted (with the current quotes).

defn Returns the quoted definition of its argument(s). It is useful for renaming macros, especially built-ins.

divert **m4** maintains 10 output streams, numbered 0-9. The final output is the concatenation of the streams in numerical order; initially stream 0 is the current stream. The **divert** macro changes the current output stream to its (digit-string) argument. Output diverted to a stream other than 0 through 9 is discarded.

divnum Returns the value of the current output stream.

dnl Reads and discards characters up to and including the next NEWLINE.

dumpdef Prints current names and definitions, for the named items, or for all if no arguments are given.

errprint Prints its argument on the diagnostic output file.

/usr/ccs/bin/m4

eval Evaluates its argument as an arithmetic expression, using 32-bit signed-integer arithmetic. The following operators are supported: parentheses, unary −, unary +, !, ˜, ∗, /, %, +, −, relationals, bitwise &, |, &&, and ||. Octal and hex numbers may be specified as in C. The second argument specifies the radix for the result; the default is 10. The third argument may be used to specify the minimum number of digits in the result.

/usr/xpg4/bin/m4

eval Evaluates its argument as an arithmetic expression, using 32-bit signed-integer arithmetic. The following operators are supported: parentheses, unary −, unary +, !, ˜, ∗, /, %, +, −, **<<**, **>>**, relationals, bitwise &, |, &&, and ||. Precedence and associativity are as in C. Octal and hex numbers may also be specified as in C. The second argument specifies the radix for the result; the default is 10. The third argument may be used to specify the minimum number of digits in the result.

ifdef If the first argument is defined, the value is the second argument, otherwise the third. If there is no third argument, the value is **NULL**. The word **unix** is predefined.

ifelse This macro has three or more arguments. If the first argument is the same string as the second, then the value is the third argument. If not, and if there are more than four arguments, the process is repeated with arguments 4, 5, 6 and 7. Otherwise, the value is either the fourth string, or, if it is not present, **NULL**.

include Returns the contents of the file named in the argument.

incr Returns the value of its argument incremented by 1. The value of the argument is calculated by interpreting an initial digit-string as a decimal number.

index Returns the position in its first argument where the second argument begins (zero origin), or −1 if the second argument does not occur.

len Returns the number of characters in its argument.

m4exit This macro causes immediate exit from **m4**. Argument 1, if given, is the exit code; the default is **0**.

m4wrap Argument 1 will be pushed back at final EOF; example: **m4wrap(\`cleanup( )´)**

maketemp Fills in a string of “**X**” characters in its argument with the current process ID.

popdef Removes current definition of its argument(s), exposing the previous one, if any.

pushdef Like **define**, but saves any previous definition.

shift Returns all but its first argument. The other 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 This macro is identical to **include**, except that it says nothing if the file is inaccessible.

substr Returns a substring of its first argument. The second argument is a zero origin number selecting the first character; the third argument indicates the length of the substring. A missing third argument is taken to be large enough to extend to the end of the first string.

syscmd This macro executes the command given in the first argument. No value is returned.

sysval This macro is the return code from the last call to **syscmd**.

translit Transliterates the characters in its first argument from the set given by the second argument to the set given by the third. No abbreviations are permitted.

traceon This macro with no arguments, turns on tracing for all macros (including built-ins). Otherwise, turns on tracing for named macros.

traceoff Turns off trace globally and for any macros specified. Macros specifically traced by **traceon** can be untraced only by specific calls to **traceoff**.

undefine Removes the definition of the macro named in its argument.

undivert This macro causes immediate output of text from diversions named as arguments, or all diversions if no argument. Text may be undiverted into another diversion. Undiverting discards the diverted text.

EXAMPLES

An example of a single **m4** input file capable of generating two output files follows. The file **file1.m4** could contain lines such as:

```
if(VER, 1, do_something)
if(VER, 2, do_something)
```

The makefile for the program might include:

```
file1.1.c :    file1.m4
               m4 -D VER=1 file1.m4 > file1.1.c
               ...
file1.2.c :    file1.m4
               m4 -D VER=2 file1.m4 > file1.2.c
               ...
```

The –**U** option can be used to undefine **VER**. If **file1.m4** contains:

```
if(VER, 1, do_something)
if(VER, 2, do_something)
ifndef(VER, do_something)
```

then the makefile would contain:

```
file1.0.c :    file1.m4
               m4 -U VER file1.m4 > file1.0.c
               ...
file1.1.c :    file1.m4
               m4 -D VER=1 file1.m4 > file1.1.c
               ...
file1.2.c :    file1.m4
               m4 -D VER=2 file1.m4 > file1.2.c
               ...
```

ENVIRONMENT

See **environ**(5) for descriptions of the following environment variables that affect the execution of **m4**: **LC_CTYPE**, **LC_MESSAGES**, and **NLSPATH**.

EXIT STATUS

The following exit values are returned:

0 Successful completion.

>0 An error occurred

If the **m4exit** macro is used, the exit value can be specified by the input file.

SEE ALSO

as(1), **environ**(5)