NAME

dwm_bison, **dwm_curpath**, **dwm_cwd**, **dwm_files**, **dwm_flex**, **dwm_funcs**, **dwm_include**, **dwm_pwd**, **dwm_relpath**, **dwm_relpwd**, **dwm_reverse**, **dwm_rgxmatch**, **dwm_rgxreplace**, **dwm_rgxsearch**, **dwm_rgxsubst**, **dwm_rotl**, **dwm_rotr**, **dwm_sort**, **dwm_subdirs**, **dwm_thisdir**, **dwm_uniqleft**, **dwm_uniqright**, **dwm_my** - GNU make extensions

LIBRARY

library "dwm_gmk.so"

SYNOPSIS

load /usr/local/lib/dwm_gmk.so(dwm_gmk_setup)

$(eval $(dwm_bison bison_source_file[,bison_args]))
$(dwm_curpath)
$(dwm_cwd)
$(dwm_files directory_path,regexp)
$(eval $(dwm_flex flex_source_file[,flex_args]))
$(dwm_funcs)
$(dwm_include filename)
$(dwm_include verbose,filename)
$(dwm_my expression)
$(my expression)
$(myns name)
$(dwm_pwd)
$(dwm_relpath from,to)
$(dwm_reverse text)
$(dwm_rgxmatch regexp,text)
$(dwm_rgxreplace regexp,format,text)
$(dwm_rgxsearch regexp,text)
$(dwm_rgxsubst regexp,format,text)
$(dwm_rotl text)
$(dwm_rotr text)
$(dwm_sort text)
$(dwm_subdirs)
$(dwm_thisdir)
$(dwm_thisdirabs)
$(dwm_uniqleft text)
$(dwm_uniqright text)

DESCRIPTION

dwm_gmk.so contains a set of utility functions that can be loaded into GNU make via its *load* directive. Once loaded, these functions can be used in the same manner as GNU make built-in functions.

dwm_bison

Usage: **$(eval $(dwm_bison bison_source_file[,bison_args]))**

Creates a rule to run bison(1) on the given *bison_source_file*. For example, if you add this line to a Makefile:

```
$(eval $(dwm_bison parse.y,-d))
```

The following will be generated and evaluated by GNU make:

```
BISONTARGETS += parse.cc parse.hh
parse.cc parse.hh &: parse.y
	bison -d -oparse.cc $<
```

Note that we add to *BISONTARGETS* so that a 'clean' rule can use it to remove files generated by bison.

dwm_curpath

Usage: **$(dwm_curpath)**

Returns the current path, using std::filesystem::current_path() from the standard C++ library (C++17 and later). This is the same as the result of getcwd(3) from POSIX.

Note that this is different than the result of *dwm_pwd*; GNU make does not set the PWD environment variable when invoked as 'make -C somedir ...'.

dwm_files

Usage: **$(dwm_files directory_path,regexp)**

Returns a list of files contained in *directory_path* whose name matches the regular expression *regexp*. Note that *regexp* is an ECMAScript regular expression.

dwm_flex

Usage: **$(eval $(dwm_flex flex_source_file[,flex_args]))**

Creates a rule to run flex(1) on the given *flex_source_file*. For example, if you add this line to a Makefile:

```
$(eval $(dwm_flex lexer.lex,-o lexer.cc))
```

The following will be generated and evaluated by GNU make:

```
FLEXTARGETS += lexer.cc
lexer.cc: lexer.lex
	flex -o lexer.cc $<
```

Note that we add to ‘FLEXTARGETS’ so that a ‘clean’ rule can use it to remove files generated by flex.

dwm_include

Usage: **$(dwm_include files...)**
Usage: **$(dwm_include verbose,files...)**

Similar to GNU make’s built-in *include* directive, but will push each included file onto an internal stack while it is being processed and pop it from the stack when done. This allows *dwm_thisdir* to continue to work correctly after including a file.

In the second form, *dwm_include* will print a line showing which file is being included. This can be useful when trying to reduce duplicate inclusions.

The internal stack of Makefiles maintained by *dwm_include* can be accessed via the *dwm_incstack* variable, with the current Makefile (the one being processed) being:

```
$(firstword $(dwm_incstack))
```

dwm_my

Usage: **$(dwm_my varname)**
Usage: **$(dwm_my varname := value)**
Usage: **$(dwm_my varname = value)**

The intent of *dwm_my* is to set or get variable values from a namespace that is separate from the global GNU make namespace. This allows one to create variables that do not conflict with global variables despite sharing the same name. Such variables are ony accessible using *dwm_my*.

varname can be fully qualified by using the form *namespace.varname*. For example:

```
$(dwm_my lib.srcs := foo.c bar.c orange.c)
```

```
$(dwm_my lib.objs := $(patsubst %.c,%.o,$(dwm_my lib.srcs)))
libfoo.a: $(dwm_my lib.objs)
        ar rv $@ $^
```

If *varname* is not fully qualified, the last namespace set with *dwm_myns* will be used as the namespace.

dwm_myns
Usage: **($dwm_myns namespace)**

Sets the current namespace used by *dwm_my*.

dwm_pwd
Usage: **$(dwm_pwd)**

Returns the value of the PWD environment variable. Note that GNU make does not modify PWD. A recursive make using **make -C ...** will not change PWD. Hence this function can be used to determine the working directory from which the user ran GNU make.

dwm_relpath
Usage: **$(dwm_relpath from,to)**

dwm_rgxmatch
Usage: **$(dwm_rgxmatch regexp,text)**

Example:
$(dwm_rgxmatch ^d(.+),dwm)
=> dwm wm

dwm_rgxreplace
Usage: **$(dwm_rgxreplace regexp,format,text)**

Example:
$(dwm_rgxreplace a|e|i|o|u,[$$&],Quick brown fox)
=> Q[u][i]ck br[o]wn f[o]x

dwm_rgxsearch
Usage: **$(dwm_rgxsearch regexp,text)**

Searches *text* for matches to the regular expression *regexp*. Returns all matches as a space-separated list.

dwm_rgxsubst

Usage: **$(dwm_rgxsubst regexp,format,text)**

Returns a string where each instance of the regular expression *regexp* in each word of *test* is replaced with *format*. Note that *regexp* is an ECMAScript regular expression. Under the hood, this function uses std::regex_replace() from the standard C++ library, executing it on each word in *text*. Note that the escaped '$1' is due to the C++ library using '$' instead of '\' for backreferences.

Example:

$(dwm_rgxsubst (.+)\.cc$$,$$1.o,foo.cc bar.cc foobar.cc))

=> foo.o bar.o foobar.o

dwm_relpwd

Usage: **$(dwm_relpwd path)**

dwm_reverse

Usage: **$(dwm_reverse text)**

Returns the words in *text* in reverse order.

Example:

$(dwm_reverse cat dog cow horse hen)

=> hen horse cow dog cat

dwm_rotl

Usage: **$(dwm_rotl text[,count])**

Returns the words in *text* rotated to the left *count* positions. If *count* is not given, 1 will be used.

Example:

$(dwm_rotl cat dog cow horse hen,3)

=> horse hen cat dog cow

dwm_rotr

Usage: **$(dwm_rotr text[,positions])**

Returns the words in *text* rotated to the right *count* positions. If *count* is not given, 1 will be used.

Example:

$(dwm_rotr cat dog cow horse hen,3)

=> cow horse hen cat dog

dwm_sort
Usage: **$(dwm_sort text)**

Returns a sorted version of the words in *text* (lexicographical). Unlike GNU make's built-in *sort*, does not remove duplicates.

Example:
$(dwm_sort d c b a c b a b a b a)
=> a a a a b b b b c c d

dwm_subdirs
Usage: **$(dwm_subdirs directory_path,regexp)**

Returns a list of subdirectories of *directory_path* whose name matches the regular expression *regexp*. Note that *regexp* is an ECMAScript regular expression.

dwm_thisdir
Usage: **$(dwm_thisdir)**

Returns the directory in which the current Makefile lives, assuming it is called at the top of the Makefile before any *include* directives or that *dwm_include* is always used in place of *include*. Note that the returned value is relative to the working directory (as would be returned by getcwd(3) from POSIX).

dwm_thisdirabs
Usage: **$(dwm_thisdirabs)**

Returns the directory in which the current Makefile lives, assuming it is called at the top of the Makefile before any *include* directives or that *dwm_include* is always used instead of *include*. The returned value is an absolute path.

dwm_uniqleft
Usage: **$(dwm_uniqleft text)**
Returns a copy of *text* with rightmost duplicates removed (leftmost retained). Unlike GNU make's built-in *sort*, does not sort. Useful for removing duplicate complier and linker flags or filenames without changing the order.

Example:
$(dwm_uniqleft a b d d b a)
=> a b d

dwm_uniqright

Usage: **$(dwm_uniqright text)**

Returns a copy of *text* with leftmost duplicates removed (rightmost retained). Unlike GNU make's built-in *sort*, does not sort. Useful for removing duplicate compiler and linker flags or filenames without changing the order.

Example:

$(dwm_uniqleft a b d d b a)

=> d b a

SEE ALSO

See **https://en.cppreference.com/w/cpp/regex/ecmascript** for the details of the modified ECMAScript regular expression grammar.

See **https://262.ecma-international.org/5.1/#sec-15.5.4.11** for information on ECMAScript backreferences.

See **https://en.cppreference.com/w/cpp/regex/regex_replace** for inforomation on std::regex_replace() from the standard C++ library.

AUTHORS

Daniel W. McRobb *dwm@mcplex.net*