

### Make command line options

<code>target</code>	Select the target to run
<code>-f file</code>	Select which <i>file</i> to read
<code>-W file</code>	Mark <i>file</i> as 'out of date'
<code>-C</code>	Change directory before <i>making</i>
<code>-d</code>	Print all debug information
<code>-n</code>	Print actions without <i>making</i>
<code>-t</code>	Mark encountered targets as 'up of date'
<code>-p</code>	Expand makefile and print
<code>-b</code>	Consider all targets as 'out-of-date'

### Automatic variables

<code>\$\$</code>	Name of the target of the recipe being run*
<code>\$\$%</code>	The target member name, when the target is an archive member*
<code>\$\$&lt;</code>	Name of first prerequisite
<code>\$\$?</code>	Names of prerequisites newer than the target
<code>\$\$^</code>	Names of all prerequisites
<code>\$\$*</code>	Name of the stem

\* In case of foo.a(bar.o) \$\$ returns "foo.a" and \$\$% returns "bar.o"

### .PHONY

Certain targets can be marked as .PHONY. By doing this, you notify make that the target is not related to a specific filename. It will thus always be rebuilt.

**.PHONY: clean**

**clean:**

```
rm *.o temp
```

In the example given, clean will always be rebuilt, even if a file named "clean" is found

### Text manipulation functions

Syntax: `$(function arguments)`

`$(subst from,to,text)`

Substitute substring *from* to *to* in *text*

`$(patsubst pat,repl,text)`

Text substitutions using pattern *pat* in *text*

`$(text:pat=repl)`

Same effect as `patsubst`, but in different form

`$(strip string)`

Strip leading and trailing spaces from *string*

`$(findstring find,strings)`

Tries to find occurrence of *find* in *strings*. Returns '*find*' if successful, else it returns "

`$(filter patterns,text)`

Returns words in *text* that match *patterns*

`$(filter-out pattern...,text)`

Returns words in *text* that DO NOT match *patterns*

`$(sort list)`

Sort list *list* of strings in alphabetical order

`$(word n,text)`

Return the *n*<sup>th</sup> word in *text*

`$(wordlist s,e,text)`

Return sublist of words list *text* starting at index *s* and ending at index *e*

`$(words text)`

Returns the number of words in *text*

`$(firstword text)`

Returns the first word in *text*

`$(lastword text)`

Returns the last word in *text*

### General rule syntax

In general, a rule looks like this:

**targets** : *prerequisites*

recipe

...

or like this:

**targets** : *prerequisites* ; recipe

recipe

...

### Variable assignment

#### Recursively expanded variable

```
var = $(shell ls)
```

The expansion of `$(shell ls)` only happens when `var` is referenced

#### Simply expanded variable

```
var := $(shell ls)
```

```
var ::= $(shell ls)
```

The expansion of `$(shell ls)` is done immediately

#### Conditionally expanded variable

```
var ?= $(shell ls)
```

Assigns the variable recursively if it is not yet defined

#### Incremental assignment

```
var += $(shell ls)
```

Appends to the variable. Assignment (recursive/simple) depends on `var`

#### Shell assignment

```
var != ls
```

Executes the `ls` command immediately in the shell and assigns result to `var`

