Make

compilebuild

Make1977C Make

MakeIsaac SchlueterMakefileGNU Make

Make

Make""Makea.txt

```
$ make a.txt
```

Makea.txt

a.txtb.txtc.txtcatmake

```
a.txt: b.txt c.txt
    cat b.txt c.txt > a.txt
```

```
make a.txtb.txtc.txtcat
```

MakefileMakeMakefilemakefile

```
$ make -f rules.txt
#
$ make --file=rules.txt
```

makerules.txt

makeShell

Makefile

MakefileMakeMakefile

2.1

Makefilerules

```
<target> : <prerequisites>
[tab]  <commands>
```

""target""prerequisitestab""commands

"""""""""

2.2 target

targetMakea.txt

""phony target

```
clean:
    rm *.o
```

clean""

```
$ make clean
```

cleanMakecleanrm

clean""

Content Menu

```
.PHONY: clean
clean:
    rm *.o temp
```

clean""makeclean.PHONY

MakeMakefile

```
$ make
```

Makefile

2.3 prerequisites

""last-modification""

```
result.txt: source.txt
    cp source.txt result.txt
```

result.txtsource.txt source.txt make result.txtsource.txt

```
source.txt:
    echo "this is the source" > source.txt
```

source.txtmake source.txt

```
$ make result.txt
$ make result.txt
```

make result.txtsource.txtresult.txtMakesource.txtresult.txtresult.txt

```
source: file1 file2 file3
```

source

```
$ make source
```

make sourcefile1file2file3

```
$ make file1
$ make file2
$ make file3
```

2.4 commands

commandsShell""

tab.RECIPEPREFIX

```
.RECIPEPREFIX = >
all:
> echo Hello, world
```

.RECIPEPREFIX>tabtab

> ⚠ **shellShell**

```
var-lost:
    export foo=bar
    echo "foo=[$$foo]"
```

```
make var-lostfoo
```

```
var-kept:
    export foo=bar; echo "foo=[$$foo]"
```

```
var-kept:
    export foo=bar; \
    echo "foo=[$$foo]"

.ONESHELL:
```

```
.ONESHELL:
var-kept:
    export foo=bar;
    echo "foo=[$$foo]"
```

Makefile

3.1

#Makefile

```
#
result.txt: source.txt
    #
    cp source.txt result.txt #
```

3.2 echoing

makeechoing

```
test:
    #
```

```
$ make test
#
```

@

```
test:
    @#
```

make test

echo@

```
test:
    @#
    @echo TODO
```

3.3

wildcardMakefileBash*?[...]*.oo

```
clean:
    rm -f *.o
```

3.4

Make%f1.cf2.c

```
%.o: %.c
```

```
f1.o: f1.c
f2.o: f2.c

%
```

3.5

Makefile

```
txt = Hello World
test:
    @echo $(txt)
```

txtHello World$( )

ShellMake

```
test:
    @echo $$HOME
```

```
v1 = $(v2)
```

v1v2v1v2

Makefile =:==+=StackOverflow

```
VARIABLE = value
#

VARIABLE := value
#

VARIABLE ?= value
#

VARIABLE += value
#
```

3.6 Implicit Variables

Make$(CC) $(MAKE) Make

```
output:
    $(CC) -o output input.c
```

3.7 Automatic Variables

Make

1$@

$@Makemake foo $@ foo

```
a.txt b.txt:
    touch $@
```

```
a.txt:
    touch a.txt
b.txt:
    touch b.txt
```

2$<

```
$< t: p1 p2$<p1
```

```
a.txt: b.txt c.txt
    cp $< $@
```

```
a.txt: b.txt c.txt
    cp b.txt a.txt
```

3$?

```
$? t: p1 p2p2t$?p2
```

4$^

```
$^  t: p1 p2$^p1 p2
```

5$*

```
$*%%f1.txtf1$*f1
```

6$(@D)$(@F)

```
$(@D) $(@F)  $@ $@src/input.c$(@D)src$(@F)input.c
```

7$(<D) $(<F)

```
$(<D) $(<F) $<
```

```
dest/%.txt: src/%.txt
    @[ -d dest ] || mkdir dest
    cp $< $@
```

```
srctxtdestdest$<src/%.txt $@ dest/%.txt
```

3.8

```
MakefileBash
```

```
ifeq ($(CC),gcc)
  libs=$(libs_for_gcc)
else
  libs=$(normal_libs)
endif
```

```
gcc
```

```
LIST = one two three
all:
    for i in $(LIST); do \
        echo $$i; \
    done

#

all:
    for i in one two three; do \
        echo $i; \
    done
```

```
one
two
three
```

3.9

Makefile

```
$(function arguments)
#
${function arguments}
```

Makefile.

1shell

shellshell

```
srcfiles := $(shell echo src/{00..99}.txt)
```

2wildcard

wildcardMakefileBash

```
srcfiles := $(wildcard src/*.txt)
```

3subst

subst

$(subst from,to,text)

"feet on the street""fEEt on the strEEt"

$(subst ee,EE,feet on the street)

```
comma:= ,
empty:=
# space
space:= $(empty) $(empty)
foo:= a b c
bar:= $(subst $(space),$(comma),$(foo))
# bar is now `a,b,c'.
```

4patsubst

patsubst

$(patsubst pattern,replacement,text)

"x.c.c bar.c""x.c.o bar.o"

```
$(patsubst %.c,%.o,x.c.c bar.c)
```

5

+ + patsubst

```
min: $(OUTPUT:.js=.min.js)
```

OUTPUT.js.min.js

Makefile

1

```
.PHONY: cleanall cleanobj cleandiff

cleanall : cleanobj cleandiff
    rm program

cleanobj :
    rm *.o

cleandiff :
    rm *.diff
```

cleanall

2C

```
edit : main.o kbd.o command.o display.o
    cc -o edit main.o kbd.o command.o display.o

main.o : main.c defs.h
    cc -c main.c
kbd.o : kbd.c defs.h command.h
    cc -c kbd.c
command.o : command.c defs.h command.h
    cc -c command.c
display.o : display.c defs.h
    cc -c display.c

clean :
     rm edit main.o kbd.o command.o display.o

.PHONY: edit clean
```

- http://www.ruanyifeng.com/blog/2015/02/make.html