GNU Make

`http://www.gnu.org/software/make/make.html`

```
$ make
$ make install
$ make clean
```

Parametry příkazové řádky

`make [ -f makefile ] [ volby ] ...  cíl ...`

- `-f file --file=file` — výběr jiného „makefile“ souboru než GNUmakefile, makefile a Makefile
- `-j` *X* `--jobs=`*X* — kolik procesů se má spustit souběžně, na SMP strojích se 2 CPU můžeme dát např. `$ make -j 2`
- `-l` *X* `--max-load=`*X* — nepouštět více jak jeden proces při zatížení (load average) větším než *X*, např. `$ make -l 2.5`
- `-k --keep-going` — pokračování zpracování souborů po chybě
- `-n --just-print` — vytiskne příkazy nutné pro zpracování určitého cíle
- `-C` *dir* `--directory` *dir* — změna adresáře na *dir*

Psaní makefile

Vše co `make` potřebuje je soubor plný definic cílů (targets), předpokladů (prerequisites) a příkazů pro spracování jednotlivých cílů.

```
target ... : prerequisites ...
     command
     ...
     ...
```

Pravidla

- Každý příkaz na vlastním řádku nebo oddělený středníkem, lámat řádky lze pomocí „\“
- Řádek s příkazem musí uvozovat TAB

Jednoduchý příklad

```
programek : main.o boo.o
    gcc -O2 -Wall -g  -o programek main.o boo.o

main.o : main.c boo.h
    gcc -O2 -Wall -g -c main.c

boo.o : boo.c
    gcc -O2 -Wall -g -c boo.c
```

Makra

```
# Definice maker
CC=gcc
DEBUG = -g
CFLAGS= -O2 -Wall $(DEBUG)

programek : main.o boo.o; @echo Jsme ve finale
    $(CC) $(CFLAGS) -o programek main.o boo.o

main.o: main.c boo.h
    $(CC) $(CFLAGS) -c main.c

boo.o : boo.c
    $(CC) $(CFLAGS) -c boo.c

.PHONY: clean

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

Výsledek příkladů

```
$ make
gcc -O2 -Wall -g -c main.c
gcc -O2 -Wall -g -c boo.c
gcc -O2 -Wall -g  -o programek main.o boo.o
```

Co ještě vědět?

- Veškeré příkazy se před provedením vypíšou na terminál, zabráníme tomu znakem `@`
- Otevření subshellu pomocí `$(shell příkazy; ...)`
- Podpora nahrazování řětězců v názvech souborů – `*`, `?`, `[...]`
- Dvě příchutě proměných „=“ „:=“

```
CFLAGS = $(include_dirs) -O
include_dirs = -Ifoo -Ibar
```

```
x := foo
y := $(x) bar
x := later
```

  - `$(CFLAGS)` se rozvine na „`-Ifoo -Ibar -O`“
  - Druhý zápis s „:=“ odpovídá

```
y := foo bar
x := later
```

- Použití funkcí, např. `wildcard`, `patsubst`

```
clean:
    rm -f $(patsubst %.c,%.o,$(wildcard *.c))
```

- Požadavek pro zpracování jiného makefile

```
subsystem:
    $(MAKE) -C subdir
```

- . . . a spousta dalších vlastností, na které už nezbyl čas. RTFM!