Yale-NUS College - 2018-2019

Bruno Bodin

YSC2227: INTRO TO C

week 06.1.makefile (auto-generated)

THE MAKE COMMAND

- When you run the `make` program, it looks for a `Makefile`.
- `Makefile` is a list a recipes to generate files automatically.
- The syntax of a `makefile` is :

```
<TARGET1>: <DEPENDENCIES FOR TARGET1>
   <STEP 1 OF RECIPE>
   <STEP 2 OF RECIPE>
   <...>

<TARGET2>: <DEPENDENCIES FOR TARGET2>
   <STEP 1 OF RECIPE>
   <STEP 2 OF RECIPE>
   <...>
```

- For example :

```
program: program.c
        gcc program.c -o program
```

make, makefile, target, dependencies

MAKEFILE: TARGET SELECTION

- When a `Makefile` has more than one target, you need to specify which one needs to be executed. **If not, default is the first target**.

```
program1: program1.c
        gcc program1.c -o program1

program2: program2.c
        gcc program2.c -o program2
```

- Example:

```
> make program1
gcc program1.c -o program1

> make program2
gcc program2.c -o program2

> make
gcc program1.c -o program1
```

MAKEFILE: VARIABLES

- in order to be parametric, your `Makefile` can have variables :

```
COMPILER := gcc

prog: prog.c
	${COMPILER} prog.c -o prog
```

- Then you can reassign this variable when running `make` :

```
> make
gcc prog.c -o prog

> make COMPILER=llvm
llvm prog.c -o prog
```

MAKEFILE: SPECIAL VARIABLES - 1

- special variables can help saving space and time
- The target `$@` and dependencies `$^` variables are such that:

```
prog: prog.c
	${COMPILER} prog.c -o prog
```

is equivalent to

```
prog: prog.c
	${COMPILER} $^ -o $@
```

- Additionnaly, with the wild card `%` :

```
COMPILER := gcc

%: %.c
	${COMPILER} $^ -o $@
```

MAKEFILE: SPECIAL VARIABLES - 2

- When using the wild card, you can get its value with `$*`
- For example

```
test_%: prog_%
	./prog_$*

prog_%: prog_%.c
	${COMPILER} $^ -o $@
```

- Please note that there is no possible default target now, so you can specify one :

```
all : test_1 test_2
```

MAKEFILE: SHELL VARIABLE

- There is a special kind of variables that are active.
- for example one can execute shell command `$(shell ..)`

```
EXERCICES = $(shell ls exercice*.c)
```

- another one can be used to remove extensions in a list of file `$(basename EXERCICES)`
- others can add prefix or suffix to those files: `$(addsuffix .o, EXERCICES)` and `$(addprefix test_, EXERCICES)`

RESOURCES AND EXERCICES

- You can find more content at this address:
  `http://www.cs.colby.edu/maxwell/courses/tutorials/maketutor/`
- **Today's exercice** is to write a makefile to compile what we have done so far for codeabbey and to test it.

KEY POINTS

- *make, makefile, target, dependencies*
- *variable*
- *wildcard*

REFERENCES

- cook and magician:
  https://pixabay.com/en/users/graphicmama-team-2641041/