
make: very very simple intro

make: what is it?

- is an interpreter of programs. These:
 - should be in a text file, *Makefile*
 - use a very special language (similar to *shell*'s)
 - be written in blocks similar to culinary recipes:
 - final dish, ingredients, cooking instructions

...make (cont.)

make: how does it work?

- it operates based on
 - dependency rules between ingredients and products (dishes)
 - *to bake a cake, the necessary ingredients should be available*
 - comparison of age between ingredients and products (dishes)
 - *a new cake is baked, only if “fresher” ingredients are available*

make: what is it used for?

- for repeating tasks (both simple and complex) in a smart way
 - for example: if something does not need to be redone, it will not be!
 - common use: preparation, update and installation of computer programs, documentation, etc.

...make (cont.)

make: how is it used?

- `shell> make`
 - *does whatever is first specified in a file called Makefile*
- `shell> make something`
 - *does whatever is specified in part something of Makefile*
- `shell> make -f anyOtherMakefile`
 - *does whatever is first specified in a file called anyOtherMakefile*
- ...examples follow...

...make: how is it used? (cont.)

Toy Makefile example:

```
# baking a chocolate cake ; cleaning the kitchen
chococake: chocolate flour banana oil recipient
    put chocolate, flour, oil and banana in recipient
    mix everything
    put recipient in oven @ 180° for 30 minutes

cleankitchen:
    wash with soap recipient, utensils and sink

# Note: each instruction line begins with TAB character!
```

- `shell> make`
 - *prepares chococake*
- `shell> make`
 - *only prepares and substitutes current chococake if one or more fresher ingredients (chocolate or flour or banana or oil or recipient) become available*
- `shell> make cleankitchen`
 - *does operations described under cleankitchen*

...make: how is it used? (cont.)

Simple Makefile example:

```
# Makefile very simple example.
## Two executables are to be created: exe , exe.stat
## Their source code is: exe.c , exe.h
### (Note: exe.h is included in exe.c)

all: exe exe.stat

exe: exe.c exe.h
    cc exe.c -o exe

exe.stat: exe.c exe.h
    cc -static exe.c -o exe.stat

clean:
    rm -f exe exe.stat
```

...make: how is it used? (cont. of Simple Makefile example)

- **shell> make**
 - *does whatever is first specified in Makefile; in this case, builds both executables exe and exe.stat;*
 - *seeks further down in Makefile for instructions on how to build exe and exe.stat*
- **shell> make**
 - *only builds new exe and exe.stat if exe.c or exe.h were changed since last build of the executables*
- **shell> make exe.stat**
 - *builds exe.stat, only if it is absent or if exe.c or exe.h were changed since last build*
- **shell> make clean**
 - *unconditionally removes named files (no dependencies)*

Some references:

- GNU's Make Manual
 - Make manual: www.gnu.org/software/make/manual/
- Ben Yoshino's Make Tutorial
 - Make - a tutorial: web.eng.hawaii.edu/Tutor/Make/
- P.F.Souto (@FEUP) Make Intro:
 - MIEIC's Computer Labs: make:
web.fe.up.pt/~pfs/aulas/lcom2012/at/11make.pdf
- JMCruz (@FEUP) simple Makefile examples:
 - Simple Makefile examples, web.fe.up.pt/~jmcruz/etc/unix/make/makefile-ex.tar.gz