Makefile Basics

System Programming 2022 Fall

[Tutorial] Makefile basics+demo - YouTube

Why we need Makefile?

A project could exist with more than one file besides the main file. Without Makefile, we have to check the dependencies for every file before compiling it.

Hence, defining the dependencies in advance simplifies the compiling process. With one make command, we could compile and run a project efficiently.

Rule Unit

A rule unit is consist of Target, Requirements (Dependencies) and Instruction. If the Requirements are met, we'll run the Instructions under this Target.

Format

Target: Requirements

\tInstruction

\tInstruction

…

Code

```
test.o: test.c
        gcc -c test.c
```

Make Clean

Clean is often used to remove all .o .exe files (targets).

Format

clean:

\trm -f Targets

Code

```
clean:
	rm -f main test1.o test2.o
```

Architecture

Compile *Link*

.c → .o → .exe

1. Default start from the first target

```
main: f1.o f2.o f3.o
        gcc -o main f1.o f2.o f3.o
f1.o: f1.c
        gcc -c f1.c
f2.o: f2.c
        gcc -c f2.c
f3.o: f3.c
        gcc -c f3.c
clean:
        rm -f main f1.o f2.o f3.o
```

3. After requirements met, link all files.

2. Requirements not met (generate f1-3.o in order)

Clean is **not the first target** or **in any requirements**.
Clean won't run unless you call make clean.

Advanced (variables)

```
CC = gcc                Compiler
CFLAGS = -g             If you want to use GDB, add this flag while compiling
OBJS = f1.o f2.o f3.o
EXE = main              Output main file name

all: ${OBJS}
        ${CC} -o ${EXE} ${CFLAGS} ${OBJS}
%.o: %.c
        ${CC} -c $^ -o $@       $^ and $@ are Automatic Variables. $^ = requirements, $@ = target.
clean:
        rm -f ${EXE} ${OBJS}
```

References

- A Brief Introduction to Makefiles (usask.ca)
- 簡單學 makefile：makefile 介紹與範例程式 | Mr. Opengate (mropengate.blogspot.com)
- Makefile Tutorial By Example