

CSC209H Worksheet: Makefiles

In this worksheet we will examine the **Makefile** for Assignment 2. Remember that the purpose of **make** is to automate the build process so that

- we don't have to type a long compile command every time we want to compile our code, and
- dependencies between files are tracked and source files are only recompiled when necessary.

1. Before we look at the full **Makefile**, consider the following **Makefile** rule:

```
test_print: test_print.o ptree.o
    gcc -Wall -g -std=gnu99 -o test_print test_print.o ptree.o
```

- (a) Circle the target.
- (b) Underline the prerequisites. What is another term for prerequisites?
- (c) How many actions does this rule have?
- (d) What does a file that ends in `.o` contain? How is it generated?

2. The **Makefile** for A2 is on the other side of the page. The remaining questions are about the **Makefile**.

Suppose that the only files in the current working directory are the source files, the header files, and the **Makefile**. In other words, this is the first time any compilation happens.

- (a) If we were to run `make print_ptree` which rule is evaluated first?
- (b) What new files would be created?
- (c) What is the *last* action that is executed in the **make** command above?
- (d) Which files will the pattern rule `(%.o : %.c)` match on?
- (e) If we the modify `ptree.c` and run `make print_ptree` again, which rules are evaluated? Which actions are executed?

CSC209H Worksheet: Makefiles

```
FLAGS = -Wall -g -std=gnu99
# FLAGS = -Wall -g -std=gnu99 -DTEST
DEPENDENCIES = ptree.h

all: test_print print_ptree

test_print: test_print.o ptree.o
    gcc ${FLAGS} -o $@ $^

print_ptree: print_ptree.o ptree.o
    gcc ${FLAGS} -o $@ $^

%.o: %.c ${DEPENDENCIES}
    gcc ${FLAGS} -c $<

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

Makefile syntax

Variable	Meaning
<code>\$@</code>	Target
<code>\$<</code>	First prerequisite
<code>\$?</code>	All out of date prerequisites
<code>\$^</code>	All prerequisites