Makefiles

and

Testing

Section 25.5, 25.6

Open Eclipse

- Open CharacterStaticStack

Makefiles

- Script that will build your code!
- Useful if you compile your code at least twice!
- GNU Make

```
make -h
```

https://www.gnu.org/software/make/manual/html_node/Automatic-Variables.html

Makefile Rules

```
What_To_Build: What_Is_Needed_To_Build
    How_To_Build

runMe: runMe.c
    gcc -o runMe -g -Wall runMe.c
```

Makefile

```
target: dependency1 dependency2
	command1
	command2
```

tab!

Given a set of dependencies, make will only run the necessary commands to build the project. Build a **dependency graph**.

If a target is older than any of its dependencies the commands are run to build the target

target and dependencies are files

Command line

```
zeus$> make tree
```

- looks for target named *tree* in Makefile and checks to see if it needs to be built

```
zeus$> make
```

- looks for the first target in Makefile and checks to see if it needs to be built
  - by convention, this target is named **all:**

Makefile

```makefile
CC=gcc
CFLAGS=-Wall -g

.PHONY: all clean

all: bin bin/stkdriver bin/stktester

bin:
    mkdir -p bin

bin/stkdriver: bin/stkdriver.o bin/stk.o
    ${CC} ${CFLAGS} bin/stkdriver.o bin/stk.o -o bin/stkdriver

bin/stkdriver.o: src/stkdriver.c include/stk.h
    ${CC} ${CFLAGS} -c src/stkdriver.c -o bin/stkdriver.o

bin/stktester: bin/stktester.o bin/stk.o
    ${CC} ${CFLAGS} bin/stktester.o bin/stk.o -o bin/stktester

bin/stktester.o: src/stktester.c include/stk.h
    ${CC} ${CFLAGS} -c src/stktester.c -o bin/stktester.o

bin/stk.o: src/stk.c include/stk.h
    ${CC} ${CFLAGS} -c src/stk.c -o bin/stk.o

tarball: clean
    tar czf ../cs300_1_PUNETID.tar.gz ../CharacterStaticStack

clean:
    rm -rf bin/*.o bin/stkdriver bin/stktester
```

Makefile

Dependency Graph

Makefile

- Variables
  - Makefiles can have variables such as CC and CCFLAGS
- Targets
  - By default, Makefile targets are “file targets” used to create other files
- .PHONY
  - Declares targets that do not represent physical files
  - target that is always out-of-date, thus, will always run when asked
  - e.g. make clean

Testing with Asserts

- stackTester.c

  success()

  failure()

  assert()

- You **MUST** reuse these functions to build your test drivers!

```
static void success (char *pszStr)
{
  printf ("SUCCESS: %s\n", pszStr);
}


static void failure (char *pszStr)
{
  printf ("FAILURE: %s\n", pszStr);
}


static void assert (bool bExpression, char *pTrue, char *pFalse)
{
  if (bExpression)
  {
    success (pTrue);
  }
  else
  {
    failure (pFalse);
  }
}
```

Testing the Stack

stktester.c

Practice

- Separate out success(), failure(), and assert() to testing.c/testing.h