UNIX LAB

CSC 2500

Department of Computer Science
Tennessee Tech University

LAB 10: Creating a Makefile

General Instructions

This assignment is *not covered sufficiently in your textbook*. There are some specific resources recommended for you to use linked below. You should utilize additional online resources to answer these questions as well.

- https://en.wikipedia.org/wiki/Makefile
- https://www.gnu.org/software/make/manual/html_node/index.html

Objective

- Writing `Makfile`
- Makefile variables
- `Targets`, `dependencies`, and rules
- Creating a shared library
- `PHONY` targets
- Auto-generating dependencies

Prerequisites

- Install unzip:
  ```
  $sudo apt-get install unzip
  ```
- Install make:
  ```
  $sudo apt-get install make
  ```
- Install g++:
  ```
  $sudo apt-get install g++
  ```
- Download craps.zip using the following commands for this purpose.

```
$wget http://100.27.26.191/craps.zip

$unzip craps.zip
```

Home Work

For this lab, you will write a make file, called `Makefile`, that compiles a craps game.

Description

Craps is a gambling game played with two dice. In Craps, the player or shooter rolls a pair of standard dice:

- If the sum is 7 or 11, the game is won
- If the sum is 2, 3 or 12, the game is lost
- If the sum is any other value, this value is called the shooter's point and he continues rolling until he rolls a 7 and loses or he rolls the point again in which case he wins

The following is a sample of the Craps game:

```
$ export LD_LIBRARY_PATH=./

$ ./craps

How many rounds of craps do you want to play? 10

Round 1 : 9*5*6*5*6*10*3*5*10*8*5*8*3*6*8*8*5*6*4*6*10*7 : LOSE

Round 2 : 6*8*9*8*6 : WIN!

Round 3 : 4*10*11*7 : LOSE

Round 4 : 8*8 : WIN!

Round 5 : 9*5*8*10*7 : LOSE

Round 6 : 3 : LOSE

Round 7 : 8*5*7 : LOSE

Round 8 : 6*6 : WIN!

Round 9 : 5*8*7 : LOSE

Round 10 : 8*10*8 : WIN!

Rounds: 10, Wins: 4
```

Your `Makefile` must compile `craps_game.cpp, craps_helper.cpp, craps_io.cpp` into object files and then combine them into a shared library. Remember that you must use the compiler flag `-fpic` when compiling and the -shared flag when linking to create a shared

library. Your `Makefile` must also create the craps object file and then link the craps object file to the shared library to create the craps executable.

Following are some features that the `Makefile` must have for you to get credit for this lab:

- The `Makefile` should construct a proper dependency graph. In other words, the `Makefile` should re-compile only those things that have changed.
- The `Makefile` must define the following variables and use them in the targets, dependencies, and rules where appropriate (every variable must be used):

Variable name	**Value**	**Description**
`PROG`	`craps`	The name of the executable
`LIBSRCS`		A list of all the .cpp file that will be included in the library
`LIBOBJS`	`$(patsubst %.cpp,%.o,$(LIBSRCS))`	The list of object files that will be included in the library. We use a pattern to generate the object files from $(LIBSRCS)
`LIBCRAPS`	`craps`	The name of the library without the beginning lib and trailing .so
`CXXFLAGS`	`-I./ -fpic`	Compiler flags needed so the compiler sees the header files in the current directory, and to generate position-independent code (PIC) suitable for the shared library
`LDFLAGS`	`-L./`	A linker flag so that the linker will look in the correct directory to find the shared library

- The `Makefile` must build the shared library as described above.
- The `Makefile` must build the executable by linking to the shared library as described above.
- The `Makefile` must include a `clean` target that delete all generated files (.o, .so, and executable files).
- The `Makefile` must include a `depend` target that auto-generates all dependencies.

- The `Makefile` must generate a correctly running program. You can take it for granted that the program's source code is correct and complete. All you must do is properly compile and link it.
- Include an `all` target that has a single dependency that is the craps executable.

The Object File List

Note that the value for the LIBOBJS variable in the table above is generated by the `Makefile` using the `LIBSRCS` variable. One of the many features of the make utility is the ability to substitute strings for other strings using patterns. In the table above, the code `$(patsubst %.cpp,%.o,$(LIBSRCS))` means to substitute any occurrence of a string in the LIBSRCS variable that ends with `.cpp` with a string that starts with the same sequence of characters, but ends with `.o`. Thus, the `Makefile` can generate a list of `.o` file name from a list of `.cpp` file names.

The clean target

The clean target removes all files generate by the `Makefile` during compilation. What is difference about the clean target when compared to the other targets is that the clean target is not supposed to generate a file. It is just supposed to run a command. If you define a target that does not generate a file with the same name, you should declare the target as `.PHONY` so the `Makefile` does not check timestamps to determine if it should generate the file. To do so in make, put a declaration on the line before you define the target that looks like the following:

```
.PHONY: clean
```

Then, declare the target as you would normally declare a target, but with no dependencies, like so:

```
clean:
    rm -f  use your variables here as parameters to rm
```

Note that the `-f` option to `rm` forces the removal of the file, and suppresses complaints if the file does not exist.

The depend target

For your `Makefile`, you only have to list dependencies for the executable target and for the shared library target. Let your `Makefile` automatically generate the rest of the dependencies so that everything gets compiled properly. To do so, you will use a feature of the g++ compiler that prints dependency information. Therefore, your Makefile's depend target will execute the following command:

```
g++ -MM $(PROG).cpp $(LIBSRCS) > depends.mak
```

The `-MM` option makes `g++` output dependency information, but only for non-system files. The command redirects output to create a file called `depends.mak`.

Finally, on the last line of your makefile (*not* part of the depends target), put the following line of code in your Makefile to include the dependencies generated by `g++`:

```
include depends.mak
```

The depend target does not generate a file called depend, and has no dependencies so make sure you declare it as `.PHONY`

Running the Makefile

Before executing the make file, execute following command at the Bash prompt to create an initial depends.mak so that the make utility does not complain that it is missing:

```
touch depends.mak
```

Then, just run your `Makefile` normally, like so:

```
$ make
```

When I run my `Makefile`, the output looks like the following:

```
$ # call make before creating the depends.mak file - will cause an error
$ make
Makefile:24: depends.mak: No such file or directory
make: *** No rule to make target `depends.mak'.  Stop.
$ # create the depends.mak file and run make – everything gets built
$ touch depends.mak
$ make
c++ -I./ -fpic   -c -o craps.o craps.cpp
c++ -I./ -fpic   -c -o craps_game.o craps_game.cpp
c++ -I./ -fpic   -c -o craps_helper.o craps_helper.cpp
c++ -I./ -fpic   -c -o craps_io.o craps_io.cpp
g++ -shared -o libcraps.so craps_game.o craps_helper.o craps_io.o
g++ -L./ -lcraps -o craps craps.o
```

```
$ # update the timestamp on craps_helper so that it looks like it has been
modified

$ touch craps_helper.h

$ # Whoops, make will not re-compile because I did not run the depend target
to generate the dependencies

$ make

make: `craps' is up to date.

$ # run make depends to generate the dependencies

$ make depends

g++ -MM craps.cpp craps_game.cpp craps_helper.cpp craps_io.cpp > depends.mak

$ # this time, make will work, and will only re-compile craps_game.cpp
because it was the only file changed

$ make

c++ -I./ -fpic   -c -o craps_game.o craps_game.cpp

g++ -shared -o libcraps.so craps_game.o craps_helper.o craps_io.o

g++ -L./ -lcraps -o craps craps.o

$ # test the clean target

$ make clean

rm -f craps craps.o craps_game.o craps_helper.o craps_io.o libcraps.so

$ # now make will compile everything

$ make

c++ -I./ -fpic   -c -o craps.o craps.cpp

c++ -I./ -fpic   -c -o craps_game.o craps_game.cpp

c++ -I./ -fpic   -c -o craps_helper.o craps_helper.cpp

c++ -I./ -fpic   -c -o craps_io.o craps_io.cpp

g++ -shared -o libcraps.so craps_game.o craps_helper.o craps_io.o

g++ -L./ -lcraps -o craps craps.o

$
```

After compiling the program, you will have to execute the following command before you run the craps executable so that it can find the shared library:

```
export LD_LIBRARY_PATH=./
```

Submission:

Submit your `Makefile`

Submission Site: iLearn (a Dropbox folder named “Lab 10”)