Lab Report

ECPE 170 – Computer Systems and Networks – Spring 2016

Name: Drew Overgaard

Lab Topic: C Programming (Language, Toolchain, and Makefiles) (Lab #: 3)

Question #1:
Copy and paste in your functional Makefile-1.
Answer:

```
all:
	gcc main.c output.c factorial.c -o factorial_program
```

Question #2:
Copy and paste in your functional Makefile-2.
Answer:

```
all: factorial_program

factorial_program: main.o factorial.o output.o
	gcc main.o factorial.o output.o -o factorial_program

main.o: main.c
	gcc -c main.c

factorial.o: factorial.c
	gcc -c factorial.c

output.o: output.c
	gcc -c output.c

clean:
	rm -rf *.o factorial_program
```

Question #3:
Describe – in detail – what happens when the command "make -f Makefile-2" is entered. How does make step through your Makefile to eventually produce the final result?
Answer:
This command executes a makefile named Makefile-2. It will run through the commands in the makefile in the specified directory. The -f specifies the name of the Makefile that we want to execute. Using a makefile is a simpler way to compile a large program.

Question #4:
Copy and paste in your functional Makefile-2.
Answer:

```
# The variable CC specifies which compiler will be used.
# (because different unix systems may use different compilers)
CC=gcc

# The variable CFLAGS specifies compiler options
#   -c :    Only compile (don't link)
#   -Wall:  Enable all warnings about lazy / dangerous C programming
CFLAGS=-c -Wall

# The final program to build
EXECUTABLE=factorial_program

# -------------------------------------------

all: $(EXECUTABLE)

$(EXECUTABLE): main.o factorial.o output.o
	$(CC) main.o factorial.o output.o -o $(EXECUTABLE)

main.o: main.c
	$(CC) $(CFLAGS) main.c

factorial.o: factorial.c
	$(CC) $(CFLAGS) factorial.c

output.o: output.c
	$(CC) $(CFLAGS) output.c

clean:
	rm -rf *.o $(EXECUTABLE)
```

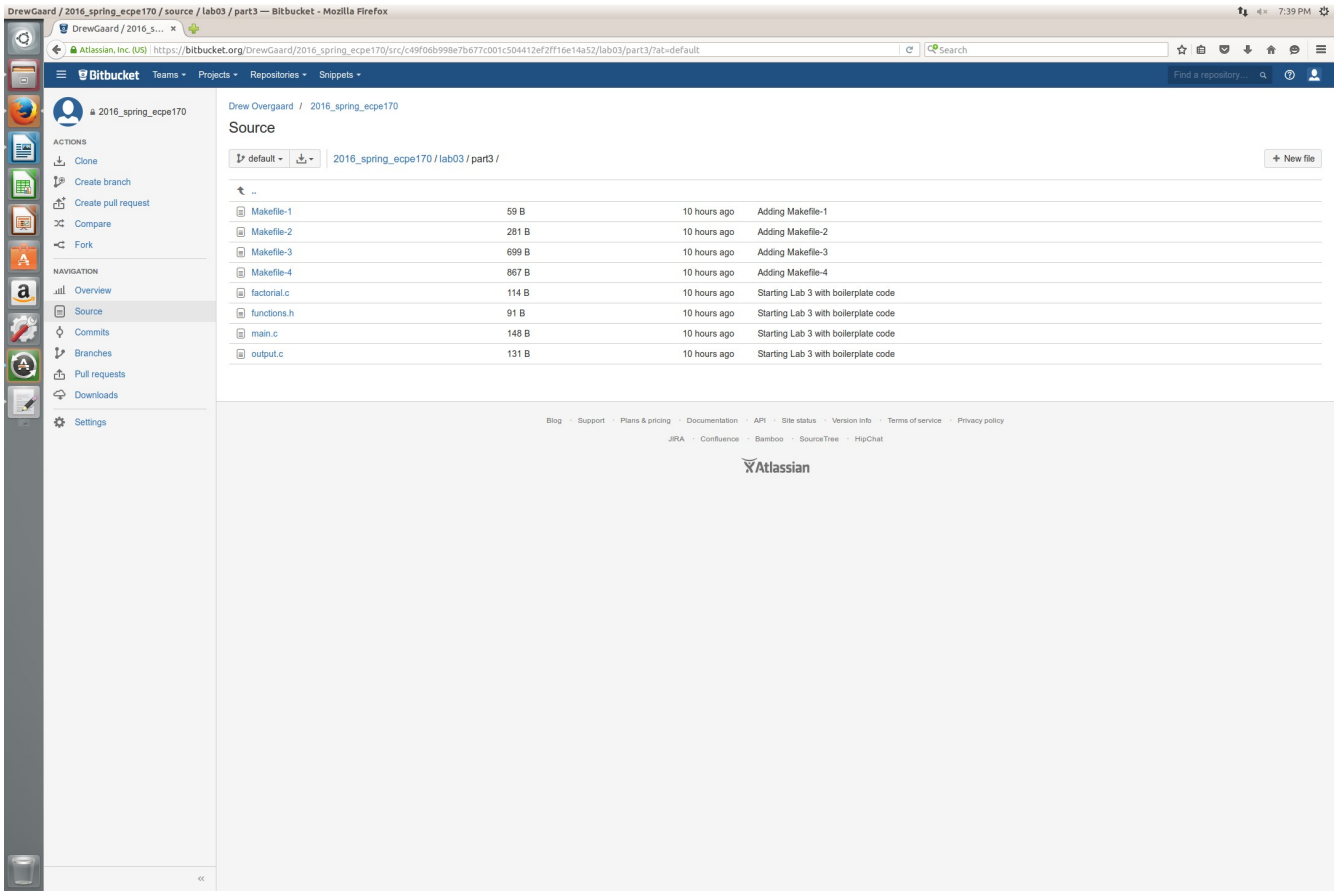
Question #5:
Copy and paste in your functional Makefile-4.
Answer:

```
# The variable CC specifies which compiler will be used.
# (because different unix systems may use different compilers)
CC=gcc

# The variable CFLAGS specifies compiler options
#   -c :    Only compile (don't link)
#   -Wall:  Enable all warnings about lazy / dangerous C programming
#  You can add additional options on this same line..
#  WARNING: NEVER REMOVE THE -c FLAG, it is essential to proper operation
CFLAGS=-c -Wall

# All of the .h header files to use as dependencies
HEADERS=functions.h

# All of the object files to produce as intermediary work
OBJECTS=main.o factorial.o output.o

# The final program to build
EXECUTABLE=factorial_program

# -------------------------------------------

all: $(EXECUTABLE)

$(EXECUTABLE): $(OBJECTS)
        $(CC) $(OBJECTS) -o $(EXECUTABLE)

%.o: %.c $(HEADERS)
        $(CC) $(CFLAGS) -o $@ $<

clean:
        rm -rf *.o $(EXECUTABLE)
```

Question #6:
Describe – in detail – what happens when the command “make -f Makefile-4” is entered. How does make step through your Makefile to eventually produce the final result?
Answer:
This command executes a makefile named Makefile-2. It will run through the commands in the makefile in the specified directory. The -f specifies the name of the Makefile that we want to execute. Using a makefile is a simpler way to compile a large program. Make files let us compile programs with one single command.

Question #7:
To use this Makefile in a future programming project (such as Lab 4...), what specific lines would you need to change?
Answer:
You would need to change the 'OBJECTS' variables in the Makefile in order for it to compile future projects. You would also have to change the 'EXECUTABLE' variable to the name you want. The 'HEADERS' variable would also need to be changed.

Question #8:
Take one screen Capture of the Bitbucket.org website, clearly showing the “Part 3” source folder that contains all of your Makefiles added to version control, along with the original boilerplate code.
Answer:


DrewGaard / 2016_spring_ecpe170 / source / lab03 / part3 — Bitbucket - Mozilla Firefox
7:39 PM
DrewGaard / 2016_s...
Atlassian, Inc. (US) https://bitbucket.org/DrewGaard/2016_spring_ecpe170/src/c49f06b998e7b677c001c504412ef2ff16e14a52/lab03/part3/?at=default
Search
Bitbucket
Teams
Projects
Repositories
Snippets
Find a repository...
2016_spring_ecpe170
ACTIONS
Clone
Create branch
Create pull request
Compare
Fork
NAVIGATION
Overview
Source
Commits
Branches
Pull requests
Downloads
Settings
Drew Overgaard / 2016_spring_ecpe170
Source
default
2016_spring_ecpe170 / lab03 / part3 /
+ New file
..
Makefile-1 59 B 10 hours ago Adding Makefile-1
Makefile-2 281 B 10 hours ago Adding Makefile-2
Makefile-3 699 B 10 hours ago Adding Makefile-3
Makefile-4 867 B 10 hours ago Adding Makefile-4
factorial.c 114 B 10 hours ago Starting Lab 3 with boilerplate code
functions.h 91 B 10 hours ago Starting Lab 3 with boilerplate code
main.c 148 B 10 hours ago Starting Lab 3 with boilerplate code
output.c 131 B 10 hours ago Starting Lab 3 with boilerplate code
Blog · Support · Plans & pricing · Documentation · API · Site status · Version info · Terms of service · Privacy policy
JIRA · Confluence · Bamboo · SourceTree · HipChat
Atlassian