Assignment 3

Linux and C Programming (62558)

Mads Richardt (s224948)

December 4, 2022

Contents

Feedback **1**

Original Submission **1**
Introduction 1
Source Code, Makefile and Project Folder Layout 2
SCP Project Folder to Student Server 3

Updates **3**
Makefile 3

Feedback

Good explanations on what is where and how you used scp to upload it to the server. It would have been good if you had included the makefile, but looking at the output from make it seems reasonable that you have a good makefile.

Original Submission

Introduction

The aim of this assignment is to introduce the student to Make[1]. Make is a build automation tool that enables automatic compilation of executable programs from source code. This is done by reading so called Makefiles, which specify how a target program should be compiled. Makefiles are just standard text files read by the Make program and their main component are so called build rules. These rules specify the dependencies of a particular file, the *target*, and provide instructions to the compiler on how build this target from these dependencies. Build rules ar constructed using the following syntax.

```
target: [dependencies ...]
Tab [command 1]
        .
        .
        .
Tab [command n]
```

[1] https://www.gnu.org/software/make/

In this assignment, the student must generate a Makefile which automates the build process of an executable program from source code. In addition, the Makefile and the source code must be uploaded to the student server using SCP.

Source Code, Makefile and Project Folder Layout

As source code, I have the chosen assignment 6 from the course thought by Ole Schults. I this assignment, we wrote a small program that counts the number of times elements from on string appears in another string. The code was written using Visual Studio Code[2] and the project folder layout and MakeFile was generated using the C/C++ Project Generator[3] extension. The project folder, a3, has the following layout.

```
a3
|_docs/
|_include/
|_src/
|_output/
|_Makefile
```

The Makefile is located in the a3 directory. The a3/docs directory contains the present document in pdf and Markdown format. The a3/include and a3/src directories contain .h and .c files, respectively. The a3/output directory will contain the executable program, hnp_a6, when successfully compiled. When we run the ***make*** command we get the following printout on the terminal, indicating that the compilation completed successfully.

```
$ make
gcc -Wall -Wextra -g -Iinclude -c src/countCharactersInSet.c
-o src/countCharactersInSet.o
gcc -Wall -Wextra -g -Iinclude -c src/get_double.c -o src/get_double.o
gcc -Wall -Wextra -g -Iinclude -c src/main.c -o src/main.o
gcc -Wall -Wextra -g -Iinclude -o output/hnp_a6 src/countCharactersInSet.o
src/get_double.o src/main.o
Executing all complete!
```

In the last call to gcc made by Make, third line from the bottom, we see that the executable target, hnp_a6, was placed in the a3/output folder. If we now run hnp_a6, we get the expected result, confirming that the build was a success.

```
$ ./output/hnp_a6
**********************
Welcome to Assignment 6
**********************

Enter text (Max length 100): linux

Enter search set (Max length 100): li

Number of hits: 2

1: Try again.
2: Exit program.
Please choose option: 2
```

[2] https://code.visualstudio.com/
[3] https://marketplace.visualstudio.com/items?itemName=danielpinto8zz6.c-cpp-project-generator

```
Closing program...
```

SCP Project Folder to Student Server

The a3 directory was converted to a tarball using the command

```
$ tar -caf a3.tar.gz ./a3
```

the tarball was then send to the student server using SCP with the command

```
$ scp -P 22222 ./a3.tar.gz student@130.225.170.80:/home/student/assignments
Enter passphrase for key '/home/madsrichardt/.ssh/id_rsa':
a3.tar.gz                                     100%  204KB 940.2KB/s   00:00
```

Updates

The Makefile was not included in the original submission - it can be found in the following section.

Makefile

```
# Define the C compiler to use
CC := gcc

# Define compiler flags
CFLAGS := -Wall -Wextra -g

# Define output directory
OUTPUT := output

# Define source directory
SRC := src

# Define include directory
INCLUDE := include

# Define output file name
MAIN := main

# Find source dirs in current directory
SOURCEDIRS := $(shell find $(SRC) -type d)

# Find include dirs in current directory
INCLUDEDIRS := $(shell find $(INCLUDE) -type d)

# Define remove command
RM := rm -f

# Define make dir command
MD := mkdir -p

# Generate include files
INCLUDES := $(patsubst %,-I%, $(INCLUDEDIRS:%/=%))
```

```
# Define the C source files
SOURCES := $(wildcard $(patsubst %,%/*.c, $(SOURCEDIRS)))

# Define the C object files
OBJECTS := $(SOURCES:.c=.o)

# Define main output file
OUTPUTMAIN := $(OUTPUT)/$(MAIN)

# Make all
all: $(OUTPUT) $(MAIN)
    @echo Executing 'all' complete!

# Make the output dir
$(OUTPUT):
    $(MD) $(OUTPUT)

# Make main from object files
$(MAIN): $(OBJECTS)
    $(CC) $(CFLAGS) $(INCLUDES) -o $(OUTPUTMAIN) $(OBJECTS) $(LFLAGS)

# Rule for making object files.
%.o : %.c
    $(CC) $(CFLAGS) $(INCLUDES) -c $< -o $@

# Declare "clean" as phony target
.PHONY: clean

# Clean
clean:
    $(RM) $(OUTPUTMAIN)
    $(RM) $(call FIXPATH,$(OBJECTS))
    @echo Cleanup complete!

# Run main
run: all
    ./$(OUTPUTMAIN)
    @echo Executing 'run: all' complete!
```