CSC 412 Assignment 1: C Intro

Out: Tuesday, September 19, 2023 03:15 PM

Due: Tuesday, September 26, 2023 11:59 AM

Contents

1 Introduction **3**

1.1 Objectives . . . 3

1.2 Preliminaries . . . 3

1.3 Resources . . . 3

2 Part I: A Simple C Program **4**

2.1 Program Description . . . 4

2.2 Input Specification . . . 4

2.3 Output Specification . . . 4

2.4 Examples - Valid Input . . . 4

2.5 Data validation (Invalid Input) . . . 5

2.5.1 Too few arguments . . . 5

2.5.2 Invalid arguments . . . 5

3 Part II: A Simple bash Program **6**

3.1 Program Description . . . 6

3.2 Input Specification . . . 6

3.3 Output Specification . . . 6

3.4 Examples - Valid Input . . . 6

3.5 Data validation (Invalid Input) . . . 7

3.5.1 Too few arguments . . . 7

3.5.2 Source file does not exist . . . 7

3.5.3 Invalid arguments . . . 7

4 Extra Credit **8**

4.1 Example . . . 8

5 Grading **9**

5.1 Grade allocation . . . 9

5.2 Design Document . . . 9

5.3 Post-Implementation Review . . . 9

5.4 Structure and Organization . . . 9

1 Introduction

1.1 Objectives

The objectives of this assignment are for you to

1. Learn and/or refresh your knowledge of executing commands in a Unix-like environment (GNU/Linux)
2. Refresh your knowledge of C programming
3. Get started with bash scripting

This is a solo assignment.

1.2 Preliminaries

1. A computer with Docker installed to run the CSC 412's development environment.
2. Working knowledge of the bash scripting language and C programming language

1.3 Resources

C:

1. https://www.tutorialspoint.com/cprogramming
2. https://subscription.packtpub.com/book/programming/9781789951288/2/ch02lvl1sec15/writing-to-stdout-and-stderr

Bash:

1. Bash cheat sheet: https://devhints.io/bash
2. https://linuxconfig.org/bash-scripting-tutorial
3. https://linuxize.com/post/bash-redirect-stderr-stdout/
4. https://www.tutorialspoint.com/execute_bash_online.php

2 Part I: A Simple C Program

2.1 Program Description

Your program will take an arbitrary number of integers between 0 and 20. It will calculate the factorial for that number.

2.2 Input Specification

Your program should take one or more space-delimited integers as input.

2.3 Output Specification

For each integer n given in the input, your program should print the factorial for that number in the format "n! = <n!>", where <n!> is the factorial value of n.

2.4 Examples - Valid Input

For this section, we will assume the program is compiled as prog01. **Do not hardcode this in**, as it is bad practice. For instance, if somebody compiles your program as foo, the usage statement "Usage: ./prog01 <num> [NUMS]" would be erroneous.

Provided correct input, the output should be as follows:

```
$ ./prog01 3
3! = 6

$ ./prog01 3 19 3 4 1 2
3! = 6
19! = 121645100408832000
3! = 6
4! = 24
1! = 1
2! = 2
```

Note: The blue color is only to differentiate program output from the shell command. Do not print it in blue.

2.5 Data validation (Invalid Input)

An important part of writing secure programs is validating, sanitizing, or rejecting invalid inputs.

Note: The red color is only to differentiate program output from the shell commands. Do not print it in red.

2.5.1 Too few arguments

If the user provides no arguments and compiles/runs the program as ./prog01, the program should print its usage message to stdout as follows:

```
Usage: ./prog01 <num> [NUMS]
```

2.5.2 Invalid arguments

There may be instances where the user provides invalid arguments such as a string or a float. In that case, print the following to stderr:

```
$ ./prog01 x
$ ./prog01 -5
$ ./prog01 5.4
[C] Error: The first argument is not an integer between 0 and 20.

$ ./prog01 12 x
$ ./prog01 12 -5
$ ./prog01 12 5.4
[C] Error: The second argument is not an integer between 0 and 20.

$ ./prog01 12 8 x
$ ./prog01 12 8 -5
$ ./prog01 12 8 5.4
[C] Error: The third argument is not an integer between 0 and 20.

$ ./prog01 12 8 1 x
$ ./prog01 12 8 1 -5
$ ./prog01 12 8 1 5.4
[C] Error: Argument #4 is not an integer between 0 and 20.
```

3 Part II: A Simple bash Program

3.1 Program Description

This is a simple bash program which has two classes of input:

1. Filename for the source code from Part I
2. One or more space-delimited integers

The program will then do the following:

1. Build the C program
2. Run C program with the space-delimited integers provided as input to the bash script

3.2 Input Specification

The program should take as input a path to the C source file followed by one or more space-delimited integers.

3.3 Output Specification

The bash script will compile the C program as "prog01", without the quotes. Given the correct input, the bash script should have no further output. The compiled program will instead provide any further output.

3.4 Examples - Valid Input

Note: The blue color is only to differentiate program output from the shell commands. Do not print it in blue.

Also note that the output is being provided by the compiled prog01, NOT the bash script itself.

```
$ ./script01.sh prog01.c 3
3! = 6

$ ./script01.sh prog01.c 3 19 3 4 1 2
3! = 6
19! = 121645100408832000
3! = 6
4! = 24
1! = 1
2! = 2
```

3.5 Data validation (Invalid Input)

Note: The red color is only to differentiate program output from the shell commands. Do not print it in red. Again, do not hardcode the script filename in.

3.5.1 Too few arguments

If the user provides one or zero arguments, the program should print its usage message to stdout as follows:

```
Usage: ./script01.sh <source> <num> [NUMS]
```

3.5.2 Source file does not exist

If the first argument is a file that does not exist, the program should print an error message to stderr.

```
$ ./script01.sh 5 10
Error: File "5" does not exist.
$ ./script01.sh doesnotexist.c 5 10
Error: File "doesnotexist.c" does not exist.
```

3.5.3 Invalid arguments

There may be instances where the user provides invalid arguments such as a string or a float. In that case, print the following to stderr:

```
$ ./script01.sh prog01.c x
$ ./script01.sh prog01.c -5
$ ./script01.sh prog01.c 5.4
[bash] Error: The second argument is not an integer between 0 and 20.

$ ./script01.sh prog01.c 12 x
$ ./script01.sh prog01.c 12 -5
$ ./script01.sh prog01.c 12 5.4
[bash] Error: The third argument is not an integer between 0 and 20.

$ ./script01.sh prog01.c 12 8 x
$ ./script01.sh prog01.c 12 8 -5
$ ./script01.sh prog01.c 12 8 5.4
[bash] Error: Argument #4 is not an integer between 0 and 20.
```

4 Extra Credit

5 points of extra credit are available if you return multiple factorials sorted by their base integer.

4.1 Example

```
$ ./prog01 3 19 3 4 1 2
1! = 1
2! = 2
3! = 6
3! = 6
4! = 24
19! = 121645100408832000
```

5 Grading

5.1 Grade allocation

Assessment Component	Percentage
Design Document	15%
Post-implementation Review	15%
Autograder Unit Tests	50%
Structure and Organization	20%

5.2 Design Document

This document describes your initial plan of attack for finishing this assignment. As such, it should be completed BEFORE you start programming anything.

This should be a PDF document. We advise you to write this in LaTeX, as it is an important skill to have if you plan to do further research. A free tool to do this is at www.overleaf.com

It should contain, at a minimum, the following:

1. How you will approach the two parts of the assignment
2. A rough outline of the functions you think you will need
3. Things you are unsure of and will need to research to complete the assignment. For example, is this your first time writing a bash script? Do you know how to redirect output in bash and C?

5.3 Post-Implementation Review

This document describes what changed between your design document and the final implementation. What worked? What didn't work? What changed? If you had topics to research, what difficulties did you have?

This should be a PDF document. We advise you to write this in LaTeX also.

5.4 Structure and Organization

Your final submission should contain the following files in a zip file:

1. prog01.c
2. script01.sh
3. design.pdf
4. review.pdf

If your files, especially the C file and bash script, are improperly named, you will fail the autograder.