

COMP 201 - Spring 2021

Assignment 0 - Getting Started with Unix and C

Assigned: 19 February 2021 00:59, Due: 25 February 2021 23:59

Samet Demir (sdemir20@ku.edu.tr) is the lead person for this assignment.

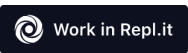
1 Introduction

The purpose of this assignment is to become more familiar with basic C fundamentals and building simple C programs as well as basic Git commands and working through Github classroom. You are asked to accept an assignment on Github Classroom using a link provided in this document and completing a simple C program that involves the very basics of C programming.

2 Logistics

This is an individual project. All handins are electronic. Clarifications and corrections will be announced on Blackboard.

3 Handout Instructions

- I Create a GitHub (github.com) account if you do not have any.
- II Fill the following Google Form so that we can match your GitHub username with you:
<https://forms.gle/eE4RMNf9tWjoY2V79>
- III Accept the GitHub Classroom assignment using the following link:
<https://classroom.github.com/a/QNH-gVdB>
- IV Clone the GitHub repository created for you to a Linux machine in which you plan to do your work. Here, you have two options:
 - Option 1: Work in [Repl.it](https://repl.it) by clicking  on your Assignment 0 - GitHub repository. **(We strongly advice you to do your work in [Repl.it](https://repl.it) since it is easier)**
 - Option 2: Work in locally-installed Linux/Unix distribution or linuxpool.ku.edu.tr servers by cloning your Assignment 0 - GitHub repository:

```
$ git clone https://github.com/COMP201-Spring2021/assignment-0-USER.git
```


(Replace USER with your GitHub username that you use to accept the assignment)

4 Task

4.1 Files

In this assignment include several files:

- `main.c`: the entry point to the C program
- `mylibrary.c`: a large open source mock library
- `mylibrary.h`: the corresponding header file of `mylibrary.c`
- **`myprogram.c`**: a simple library with 5 functions
- `myprogram.h`: the corresponding header file of `myprogram.c`
- `Makefile`: the make script
- `README.md`: the repository readme file
- `test`: a directory containing automated tests

You only need to modify the `myprogram.c` file. In particular, any section that needs modification and writing of code is marked with a `//TODO` comment. For each `//TODO`, you need to implement the corresponding function by writing a short block of code that satisfies the functionality that `//TODO` comment asks for.

4.2 Functions To Be Implemented

- `factorial(n)`: calculates $n!$ which equals to $1 \times 2 \times \dots \times n$
- `sum_of_odd_numbers(array, count)`: calculates the sum of odd numbers in the given array with length of `count`.
- `min_of_numbers(array, count)`: returns the minimum number in the given array with length of `count`.
- `reversed_number(number)`: reverses the digits of the given number. For example, if the number is 12345, return 54321.
- `alphabet_index(index)`: return the alphabet (as a lowercase letter) of given `index` assuming. For index 0, return 'a'; for index 1, return 'b'; ... for index 25, return 'z'. If index is out of range, return space ' '.

5 Testing your work

5.1 How to run

To run the program, you must first build it. The provided Makefile streamlines this process, all you need to do is type `make` on command-line and hit enter, and the program will be compiled and the binary file `assignment0` will be created (assuming there are no errors).

You can run this binary file by typing `./assignment0` which will start the program. The program start point is the function `main()` inside `main.c`. Feel free to read that file but please do not modify anything except `myprogram.c`.

5.2 Autograding

You can manually test your code by running the program and providing it with manual inputs, or type `make test` on command-line and hit enter to run the automated tests that are included in the repository.

If any of the automated tests included in the package fail, you will receive an error in the form below:

```
$ make test
1c1
< 16
---
> 28
Test test/2-1.test failed
make: *** [test/2-1.test] Error 1
```

The error specifies which test has failed (which you can look inside `test/` directory to see expected input/output) and also tells you what output was expected and what was generated by the program. For example, in the figure above, the expected output was 12 but the generated output was 28.

It goes without saying that most of the tests should fail when you begin the assignment, as the corresponding code is not written yet.

6 Git Instructions (Handin and Version-Control)

You will submit your work using Git and GitHub. You are expected to use Git effectively in terms of committing your work frequently. See Section 6 (Evaluation/Effective use of version control points).

Here are the steps to submit your work (you are expected to follow these steps for every meaning change as described in Section 6):

- I Commit the changes you make: `$ git commit -a -m "commit message"`
Note: Please do not forget to replace `commit message` with an appropriate commit message describing the commit.
- II Push your work to GitHub servers: `$ git push origin main`

7 Evaluation

Your score will be computed out of a maximum of 100 points based on the following distribution:

60 Correctness points.

30 Effective use of version control points.

10 Style points.

Correctness points. We will evaluate your functions using the `make test` to autograde. You will get full credit (12 points) for a function if it passes all of the tests performed by `make test`, and no credit otherwise.

Effective use of version control points. You are required to push your changes to the repository frequently. If you only push the final version, even if it is implemented 100% correctly, you will lose a fraction of the grade because you are expected to learn to use Version Control Systems effectively. You do not have to push every small piece of change to Github but every meaningful change should be pushed. For example, each of the functions coded and tested can be one commit. **For each function, there should be at least one commit (with proper commit message) that includes just modifications on that function.** Here, 30 points divided between functions. For each function with at least one proper commit, you will get 6 points.

Style points. Finally, we've reserved 10 points for a subjective evaluation of the style of your solutions and your commenting. Your solutions should be as clean and straightforward as possible. Your comments should be informative, but they need not be extensive.

8 Academic Integrity

All work on assignments must be done individually unless stated otherwise. You are encouraged to discuss with your classmates about the given assignments, but these discussions should be carried out in an abstract way. That is, discussions related to a particular solution to a specific problem (either in actual code or in the pseudocode) will not be tolerated. In short, turning in someone else's work, in whole or in part, as your own will be considered as a violation of academic integrity. Please note that the former condition also holds for the material found on the web as everything on the web has been written by someone else. See [Koç University - Student Code of Conduct](#).

9 Late Submission Policy

You may use up to 7 grace days (in total) over the course of the semester for the assignments. That is you can submit your solutions without any penalty if you have free grace days left. Any additional unapproved late submission will be punished (1 day late: 20% off, 2 days late: 40% off) and **no submission after 2 days (48 hours) will be accepted.**

10 Useful Links

- If you are a Windows user, you can study this short tutorial on how to use Git on Windows:
<https://www.computerhope.com/issues/ch001927.htm>
- 35 basic Linux commands that every user should know:
<https://www.hostinger.com/tutorials/linux-commands>
- You can also take a look at Unix shell commands on page 15 and 16:
<http://cslibrary.stanford.edu/107/UnixProgrammingTools.pdf>