CSCI 3614, Systems Programming
Assignment 2 (08.04.2019)

This assignment contains 5 tasks, giving you 100 points in total. In this assignment each task will be an extension of the previous one. Basically, you are going to work on a single application step by step. The name of the executable for this application should be **shell**.

Make sure that your code is neat, clean and thoroughly commented. Try to use meaningful variable names.
Codes will be thoroughly checked for plagiarism and cheating. It is not recommended to work on the implementation of the assignment as a group work. Students producing similar implementations will be penalized, in severe cases an honor code case will be reported.

Deadline: 15.04.2019 23:59

1 Simple Exec (25 points)

Your task is to create a program that executes given user command. *Simple Exec* should wait for user input and execute the command given by the user. The first token in the user input is the **command** or the **executable**, all the next tokens are going to be the **command line arguments**. Tokens consist of alphanumerical characters and symbols. After executing a command, simple should exit with the same *exit status* as the **command** itself. In case if the execution of the command was not successful, your application should exit with an status -1.

2 Simple Shell (20 points)

The next task is to extend the first one by making it possible to execute commands, not just once, but repeatedly, until user prints **exit**. The shell should work like **/bin/sh**, except you don't need to implement pipes, forwarding, etc.

3 Simple Shell Extended (15 points)

Extend the *Simple Shell* to execute not just one command, but several commands separated by the **&&** characters. They should be executed one after another, whenever the previous one terminates. In case of an error, the shell should terminate the execution of command sequence and just ignore them.
Another extension of the shell is that by putting a **backslash (\)** character at the end of a line, one should be able to extend the input to the next line. The \ character itself is not part of a command, acts as a delimiter between the lines and should be ignored. It will allow the user to separate a long command into multiple lines.

4 Simple Shell with Statistics (20 points)

In this task, each time when a user executes a command, after the command has been successfully terminated, you need to print some statistical information about it. Which information to display should depend on command line arguments/options with which the *Simple Shell* has been executed:

- **-c** user and system CPU time used by the process
- **-m** size of the operating residential memory used by the process
- **-b** number of block input and output operations performed by the process
- **-a** all options at once

For example, when executed like **./shell -mc**, after each user input, your *Simple Shell* should be able to display memory and CPU usage information about the command that has been successfully executed.

5 Simple Shell Restricted (20 points)

Requirement for this task is implementation of an additional command line argument **-r**, which is going to restrict resources for all commands executed in the shell. Option **-r** should always come as the last option and it expects three integer values after itself, which stand for following values measures: **CPU time limit**, **virtual memory limit** and **stack size limit**. If **-r** is specified without any values, default values for the resources are going to be used:

- **CPU time: 2 seconds**,
- **Virtual Memory: 16 MB**,
- **Stack Size: 1 MB**.

Processes exceeding their limits should be terminated and a message about resource limit exceeding process should be displayed. Note, that all error messages should be printed to the standard error.