**Faculty of natural & Applied Sciences**
Department of Computer Science

Spring 2020
Wednesday, March 25, 2020

**CSC 414 Applied Operating Systems**

Project 1 – Implementing a Shell

**Due date: Wednesday, April, 8, 2020, 10 PM**

# 1 Overview

In this *individual* project you will have to design and implement a simple shell command interpreter, called *mysh*. The basic function of a shell is to accept lines of text as input and execute programs in response. The shell must be able to execute built-in commands in a process different from the one executing *mysh*.

# 2 Requirements

When first started, your shell should initialize any necessary data structures and then enter a loop of printing a prompt, e.g. "mysh# ", reading any commands entered by the user and executing them.

## 2.1 Simple execution of commands

*mysh* executes a program in the foreground, waits for it to terminate, and then resumes control, e.g.

```
mysh# mcedit
```

would launch the *mcedit* editor, the user can then perform text processing and after the user quits *mcedit*, control is returned back to the shell. The user shouldn't have to type in the full path to executables as long as they are within the search path specified in the environment by the *PATH* variable. In your initial implementation *mysh* should spawn a child process and block wait for the child to terminate. The relevant system calls here are *fork()*, the *wait()* and *exec()* families of calls, and *exit()*.

## 2.2 Parsing the command line

When parsing the command line, *mysh* should ignore all whitespace. You might find the *isspace* and *strtok* man pages useful. Please keep in mind that your commandline parser should work, but we do not expect it to be particularly fancy. For example, " and ' should not be treated specially by the parser. We'd much rather see you spend a lot of effort on the job control features of your shell.

## 2.3 Built-in commands

*Mysh* should support the following built-in commands.

- **ls:** List directory content
- **grep**: Print lines matching a pattern
- **pwd**: Print the current working directory.
- **cd path**: Change the current working directory to specified path.

## 3 Hints and Tips

- The man pages are your friends! Manpages of particular interest would be:
  - fork()
  - exit()
  - exec(), execl(), execv(), execle(), execve(), execlp(), execvp()
  - wait(), waitpid()
  - kill()
  - isspace(), strtok()
- This project should be developed on a UNIX OS. You are free to use any flavor of UNIX you want for development.
- Start *early*! If you wait until the last minute you will not be able to finish.

## 4 Deadline and Turnin

**This project is due at 10:00pm, April 8.** Since this is a long-term project **no late submissions will be accepted.**

Place all your files in a directory called *project1*. The directory should contain a text *README* file explaining any particular design decisions or features of your implementation. It should contain at least three parts: 1) an introduction to the basic structure of your shell; 2) a list of which features are fully implemented, partially implemented, and not implemented; and 3) a discussion of how you tested your shell for correctness.

**Submit the project files in a compressed zip archive to kkhaldi@ndu.edu.lb.**