Assignment 1: Process API in Unix

Part 1: Live Coding Exam (In-Class, Date: 01/24/2023)

On the exam date, you need to bring a WIFI-enabled laptop to the classroom, this exam will have to be accessed on your laptop through WIFI connections. Everyone is required to be presented in the classroom to take the exam in person. TA and I will give you the access code in the classroom at the beginning of the class. If you didn't come, you will not get that access code and will not be able to see the questions. So, please try to come to class earlier to put in the access code so as to start the exam on time. This exam is open books and open notes but no solution search and communication with others is allowed. At the same time, no shared documents (such as Google Docs, Google Sheets, etc.) are allowed during the exam. I trust you take the academic honor code seriously and thus will not look for help outside your own capacity of solving problems and answering questions related to operating systems. FYI, during the exam, TAs will help monitor plagiarism, so, please be careful. After you get the access code, you can not share this code with anyone else, otherwise, it's counted as plagiarism too.

Besides, you are strongly encouraged to compile and run your own code on the C4 Lab machine during the exam in order to test your code. Please paste your tested code through the text box for each question before the submission. Late submission is NOT acceptable.

This exam contains **2 questions** of **10 points** each. The maximum points you can get from solving these questions: is **20**. You have 60 minutes to finish the exam. You may get partial credits depending on your answer's logic flow and complete degree, but 50% will be taken off if the code is not compilable.

Part 2: CPU Virtualization (Deadline: 02/16/2023, 11:59 PM on Canvas)

Learning Objectives

The objective of this assignment is to experiment with process API in Unix and to better understand how the Unix shell works by trying to mimic its behavior.

Problem 1

Write a C program that does the following:

- Takes an input from the command line. This input can be a sequence of characters without separators, such as ”date” or ”ls”, or a sequence that contains separators (e.g., space or ”-”), such as ”ls -l”. Let's refer to this input as *cmd* if only one word, or *cmd* and *params* if more than one word. If the input has multiple separators, *cmd* is the sequence of characters before the first separator, and *params* is the rest of the input. For example, if the input is: ls -a -l then *cmd* is ’ls’ and *params* is ’-a -l’.

- Creates a new process (using fork());

- Makes the new process execute *cmd* with *params* as parameters, if given.

- Waits for the new process to finish executing, and then prints ++++ on a new line.

Problem 2

This problem builds significantly on the previous problem. Specifically, it asks you to again use fork() to create processes and exec() (or one of the many variants) to assign the newly created processes what to do. In addition, however, it asks you to mimic the behavior of a shell command such as:

```
ls | wc
```

What happens in the case above is the output of the first command ('ls') becomes the input to the second command ('wc'). (Try it in a terminal on a Unix machine). Thus, you are required to write another program:

- Expects an input of the form: cmd1 | cmd2
- Creates two processes
- Makes the first process run cmd1
- Makes the second process run cmd2
- Makes sure that the output of the first process becomes the input of the second process (using the function pipe()).
- Waits for the two processes to finish before it ends by printing ++++ on a new line.

What to Submit

Name your C files as problem1.c and problem2.c and place them in a folder called assignment1. Compress the folder into a .tar file:

```
tar -cvf assignment1.tar assignment1
```

You will then need to download the tar file into your local machine. You may use FileZilla, PuTTY or a similar tool. If doing it from the CLI, issue the following command: scp <usf_id>@cselxXX.csee.usf.edu:<abs path to the tar file> .

This will download the tar file to the location this command was run from. Submit the tar file on Canvas.

Other Instructions and Suggestions

- This is an independent assignment. You are expected to work by yourself. If you use external sources (such as code from the web), please cite them as comments in your code. We will use MOSS to identify plagiarism in code.
- The C functions (that, in fact, use system calls) you'll need include: fork(), wait(), pipe(), exec() (or variants). You might want to also experiment with and use for debugging the functions getpid() and getppid().

- For on-line reference manuals use the command man as below:

```
man -s 2 pipe
man man man
getppid
man -s 2 exec
```