Process API in Unix

Part 1: CPU Virtualization

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.

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

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