Fitchburg State University

CSC 7132 Operating Systems and Networking

Instructor: Nguyen Thai

Due: 3/24/2017 at 11 PM

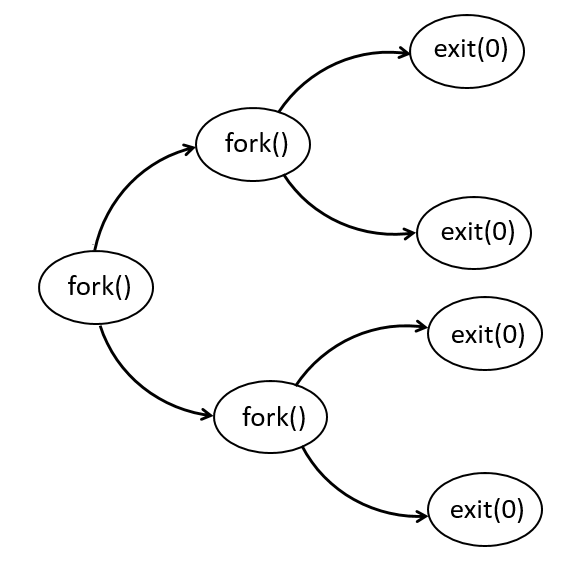
Student: Piyusha Jahagirdar

**CSC 7132 Assignment 2: Create Processes**

The purpose of this lab is to learn how to create processes in Linux.

As you work through the lab be sure to answer all questions (type your answers into this document) and take all screenshots as requested (copy them into the document). For the screenshots, you can use the Snipping Tool that is built-in to Windows to capture the important parts of the lab as highlighted in the document below. Do not delete the contents of this file. When finished, you will submit the document, source code file and associated data files to the instructor via Blackboard. DO NOT SUBMIT ZIP FILES OR INDIVIDUAL IMAGES. If you have any questions or need any clarification, see the instructor *before* the deadline.

1. Write a C program called *createProcesses.c* to implement the following diagram:



1. Document your program logic here.

**Solution:**

pid==0

pid1>0

pid2>0

pid>0

Child

Child

Parent

Child

Parent

Parent

Main

fork()

fork()

fork()

pid2==0

pid1==0

exit(0)

exit(0)

exit(0)

exit(0)

* In the main() we will create variables pid, pid1, pid2 of type pid\_t
* Then call the fork() and store the value in pid, this will result in a child process(pid==0) and a parent process(pid>0). Here we will also check if pid<0 in case forking failed.
* In the child process where pid==0, we will print that it’s a CHILD and the pid of the process.
  + Again, call fork() in the CHILD and store the value in pid1, this will result in a child process(pid1==0) and a parent process(pid1>0). Here we will also check if pid1<0 in case forking failed.
  + The newly created child process will have pid1==0, we will print that its CHILD->CHILD and the pid of the process. We will then call exit(0) for the process to exit.
  + The parent process here will have pid1>0, we will print that its CHILD->PARENT and the pid of the process. We will call wait() on the parent process to let the child complete its execution and then call exit(0) for the process to exit.
* Similarly, in the parent process where pid>0, we will print it’s a PARENT and the pid of the process. At the end it will wait for its CHILD to complete.
  + Again, call fork() in the PARENT and store the value in pid2, this will result in a child process(pid2==0) and a parent process(pid2>0). Here we will also check if pid2<0 in case forking failed.
  + The newly created child process will have pid2==0, we will print that its PARENT->CHILD and the pid of the process. We will then call exit(0) for the process to exit.
  + The parent process here will have pid2>0, we will print that its PARENT->PARENT and the pid of the process. We will call wait() on the parent process to let the child complete its execution and then call exit(0) for the process to exit.
* The program will terminate.

1. Submit this document and your source code to Blackboard for grading.