COSC 3360/6310 - Operating Systems Fall 2017

Programming Assignment 1 - Pipes and Process Synchronization

Due Date: Monday, October 9, 2017, 11:59pm CDT

Objectives

This assignment should help you to understand several basic UNIX/Linux primitives, implemented at the process management layer of an OS, such as fork(), execv() and pipe() as well as simple process synchronization and mutual exclusion. You will also learn how to manage a producer-consumer relationship between X processes, where X is an input value.

Specifications

You are to write a C or C++ program that will spawn X processes cooperating with each other to execute two steps of a given task. You will have to set up a pipe to allow the output of the first X-1 processes (the producers) to become the input of the other one (the consumer). The producer processes access and modify a shared resource (a file) and thus need to be synchronized.

Your program should:

1. use a pipe() system call to create a communication channel having one read file descriptor and one write file descriptor (the pipe),
2. divide itself into X processes using a fork() system call, creating X-1 child processes (the producers).
3. redirect the standard output of the child processes to the write field descriptor of the pipe and the standard input of the parent process to the read field descriptor of the pipe,
4. have each child process execute the producer program while its parent executes the consumer program.

Your producer processes should read data from an arbitrary text file passed as first argument of your main program and echo it after having done a minimum amount of text formatting detailed below. The second argument to your main program is the value of X (indicating the total number of processes). Whenever a number (integer) is found in the text, each of the X-1 child (producer) processes should increment/decrement that number by the number specified as the third argument to your main program. Thus the number 50 becomes 80 if the third argument is 5. Similarly, the number 50 becomes 30 if the third argument is -5. However, the non-numeric text should be left intact. The content should be output by the consumer process with the modification done to each number. This, of course, requires synchronization between the producer processes. Note that the producers can be thought of as X-1 ATM processes making deposits/withdrawal to the same bank account.

1. input lines should be filled and joined to produce output lines of up to 72 characters,
2. blank lines should be preserved, as well as the spacing between words,
3. indentation should be preserved and input lines with differing indentation should not to be joined, and

4. substrings that do not contain blanks or tabs should not be divided or hyphenated at the end of a line.

Your consumer program should display its input twenty lines by twenty lines asking the user to hit the return key to let the process continue.

Hints

Before you start your assignment, familiarize yourself with the way C/C++ programs handle arguments that are passed to them by execv() and with the UNIX functions fork(), pipe() and dup().

A Last Word

The programs you turn in should be well documented. Each C/C++ func- tion should start by a brief description of its purpose and its parameters. Each variable declaration should contain a brief description of the variable(s) being declared.