Lab Assignment #2

CS 3080 Inter-Process Communications Spring 2021

Dr. Rajaei

Due: Pseudocode: Feb 4, at 11:00 on uploaded to Canvas
Program: Feb 11, at 11:00 pm on BGLinux

Total points: 30

Objective:

- Practice writing Pseudocode first [program design]
- Understand process creation and task assign
- Understand write/read communications between processes
- Practice with fork & pipe system calls

Pseudocode: 10 points [Due: Feb 4, at 11:00 pm online, upload Canvas]:

Write first the pseudo code of your program to show its key steps whose details given below.
This ***Pseudocode*** has three major parts: the ***main*** body + ***parent*** part + ***child*** part.
Following items should be parts of the tasks your pseudocode must show:

- Initialization RNG using seed
- Create ***pipe*** & ***fork***
- Identify and assign parent task
- Identify and assign child task
- Initialize pipe for read or write
- Communication via the pipe
- For each process outline and complete its own tasks
- Print results and exit

Program: Pipe & Fork, and IPC: 20 points [Feb 11, 11:00 pm, on BGLinux]

Overview: Write a C++ wrapper [embeds C] program called ***pipeLab.cpp*** that creates a pipe channel and then a child process to communicate via the pipe with the parent. The parent ***write****s* to a pipe with multiple random numbers while the child ***read****s* numbers from the pipe and do some statistical calculations. When done, both processes print their messages and exit the program.

Details: The parent calls first the ***pipe*** system-call and creates the ***pipe***, then calls the ***fork*** to create the child. After that, both parent and child prepare a one-way pipe channel, where the parent fills the pipe ***one-at-a-time*** with random numbers [i.e. items] until certain limit [***numItems***] is reached.

The child reads these numbers ***one-at-a-time***, update the received data, and at the end finds the ***min***, ***max*** and ***avg*** (average) of all number received, and prints the results on the screen with its own real ID [i.e. using ***getpid()*** system call]:

`Child ID: number items received: n, min: xx, max: yy, avg: zz.z`

When parent is done, it waits first the child to be terminated by using ***wait***() system call. Then the parent prints:

`Parent ID: number items written into the pipes are: nn`

The program uses ***argc*** & **argv** to pass ***three (3)*** parameters to the program:

1. The 1st item is ***numItems*** [e.g. 200], to generate

2. The 2nd item refers to ***range*** of the numbers [e.g. 1000]
3. The 3rd item refers to ***seed*** of the random number generator [e.g. 3127]

where the ***argv[0]*** has the name of the executable program [i.e. pipeEx], thus ***argc*** must be 4.

The parent generates random numbers in the range 0 to 999 (modulo division by 1000).

In this program, there is ***only <u>one</u> pipe***, to pass the random numbers from parent process to child. Parent & child use ***getpid(void)*** system call to get and print their real IDs.

<u>Your program should run as</u>:

```
$ g++ pipeLab.cpp -o pipeEx
$ ./pipeEx 200 1000 3127   [i.e. pipeEx numItems range seed]
```

<u>Notes:</u>

- Make a sub-directory named ***IPC*** in your ***cs3080*** class directory and do all your work regarding lab2 assignment in this directory.
- Name your program as ***pipeLab.cpp***
- ***<u>Write your Pseudocode first and submit (upload) a file copy at its due date</u>***
- Implement your C++ (embedded with C) program code based on your pseudocode
- Develop and test your program with small number of items first (e.g. 10)
- Test your program with larger number of inputs and make sure it works as well
- Make a log-file named ***pipeLab.log*** as described in Turn-in below
- The help session for this program will be announced

<u>To Turn-in:</u>

- Your Pseudocode must be submitted as a **printed** file [i.e. no hand-written] uploaded (at the due date) to course Canvas under Lab2 Assignment. No email submission is accepted.
- All program assignments should be submitted through BGLinux and through cs3080/IPC.
- Make ***IPC*** subdirectory in your ***cs3080*** class directory and do all your lab2 works there.
- To compile: ***g++ pipeLab.cpp -o pipeEx***
- Run: ***./pipeEx 200 1000 3127***
- Create ***pipeLab.log*** by using the following CMDs and leave the log by due date to be collected for grading:

```
$ script [or photo] pipeLab.log
$ pwd
$ ls –o
$ cat pipeLab.cpp
$ g++ pipeLab.cpp -o pipeEx
$ ./pipeEx 200 1000 3127
. . . .
```

- Our auto-collecting program will collect your ***<u>pipeLab.cpp</u>*** <u>and</u> ***<u>pipeLab.log</u>*** files at the due date for grading. This program is case sensitive and looks for exact words/names in your cs3080/IPC director. Needless to say, if any error, the program cannot find it and you are risking to miss your scores. <u>As a result, no points will be received for this lab if any part is missing or named differently in IPC folder</u>.