CS 3305A: Operating Systems
Department of Computer Science
Western University
Assignment 2
Fall 2023
Due Date: October 11, 2023

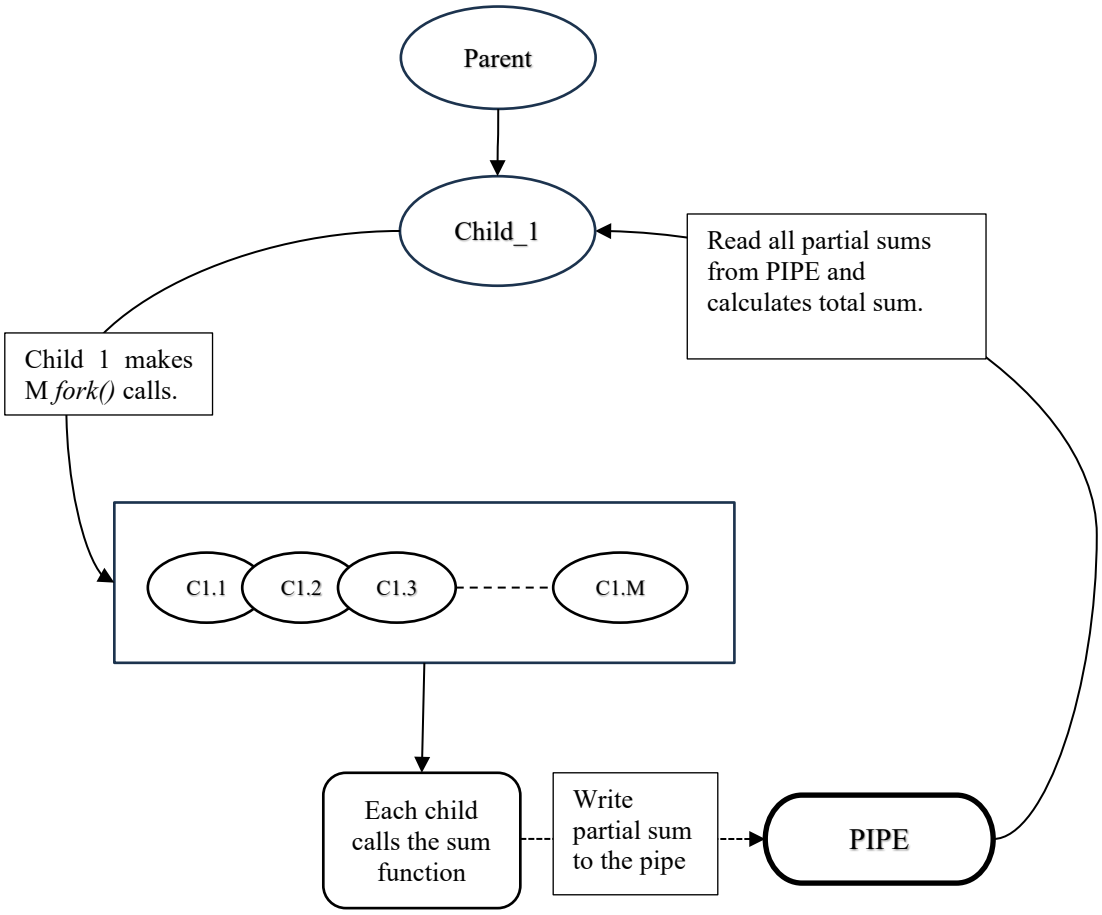
Purpose

The goals of this assignment are the following:

- Learn about process creation and control in Linux environment.
- Learn how to use pipe for bi-directional communication between parent and child process.
- Gain more experience with the C programming language from an OS perspective.

Assignment-2: Inter-Processes Communications (100 points)

Complete the given C program, **assignment2.c** to calculate the sum of N natural numbers using M processes. Ensure that your program follows the prescribed task sequence provided in the accompanying diagram below (a visual representation of the process flow).



Execution Flow of your program:

1. Your main program (referred to as the parent process) will initiate the creation of a child process, denoted as child_1.
2. The parent process will pause its execution, awaiting the completion of child_1's tasks before it concludes its own operation.
3. Within child_1, the fork() function will be called M times, utilizing a loop to generate M child processes.
4. Each child_1.i (ranging from child_1.1 to child_1.M) will invoke the given summation function to calculate a partial sum. The implementation of the summation function is already provided in the assignment2.c file. It is necessary to provide the starting and ending numbers of the range to be summed to the summation function. To determine these values, utilize the given *ith_part_start()* and *ith_part_end()* functions.
5. Each child_1.i (child_1.1 to child_1.M) will write the partial sum to a pipe.
6. Upon the completion of all M child processes, child_1 will gather the partial sums by reading from the pipe and compute the total sum.
7. Subsequently, child_1 will display the total sum and conclude its own execution.
8. Finally, the parent process will conclude its own execution as well.

Variables *N, M* must be passed to the main program as a command line argument (hint: utilize argc and argv). Implementation of *summation(), ith_part_start()* and *ith_part_end()* is already provided in the assignment2.c

Constraints:
Please follow the constraints below while developing the source code.

$$\mathbf{1 \leq M \leq 10}$$
$$\mathbf{1 \leq N \leq 100}$$

Example:

sample Input:

N = 12
M = 5

The output of your program must be in the following format and sequence:

parent(PID 7056): process started

parent(PID 7056): forking child_1
parent(PID 7056): fork successful for child_1(PID 7061)
parent(PID 7056): waiting for child_1(PID 7061) to complete

child_1(PID 7061): process started from parent(PID 7056)
child_1(PID 7061): forking child_1.1....child_1.5

child_1.1(PID 7062): fork() successful
child_1.1(PID 7062): partial sum: [0 - 1] = 1
child_1.2(PID 7063): fork() successful

child_1.2(PID 7063): partial sum: [2 - 3] = 5
child_1.3(PID 7064): fork() successful
child_1.3(PID 7064): partial sum: [4 - 5] = 9
child_1.4(PID 7065): fork() successful
child_1.4(PID 7065): partial sum: [6 - 7] = 13
child_1.5(PID 7066): fork() successful
child_1.5(PID 7066): partial sum: [8 - 12] = 50

child_1(PID 7061): total sum = 78
child_1(PID 7061): child_1 completed

parent(PID 7056): parent completed

Note that the output will change depending on the parameter passed (e.g., N, M) during the execution of the program.
Hints: fork(), wait(), getpid(), getppid(), pipe()

Mark Distribution: 100 points

a) A parent process will create a child processes: 10 points
b) parent will wait for child_1 to complete terminating itself: 10 points
c) child_1 will create its own child child_1.1 to child_1.M: 10 points
d) child_1.1 to child_1.m processes each calculate and write partial sum to the pipe: 40 points
e) child_1 waits for the child_1.1 to child_1.M to complete and read partial sum from the pipe: 20 points.
f) Parent process terminates after completing the execution of all the child processes: 10 points.

Computing Platform for Assignments

You are responsible for ensuring that your program compiles and runs without error on the computing platform mentioned below. **Marks will be deducted** if your program fails to compile, or your program runs into errors on the specified computing platform (see below).

- Students have virtual access to the MC 244 lab, which contains 30 Fedora 28 systems. Linux machines available to you are: **linux01.gaul.csd.uwo.ca** through **linux30.gaul.csd.uwo.ca**.
- It is your responsibility to ensure that your code compiles and runs on the above systems. You can SSH into MC 244 machines.
- If you are off campus, you must SSH to **compute.gaul.csd.uwo.ca** first (this server is also knownas sylvia.gaul.csd.uwo.ca, in honor of Dr. Sylvia Osborn), and then to one of the MC 244 systems **(linux01.gaul.csd.uwo.ca** to **linux30.gaul.csd.uwo.ca**).
- https://wiki.sci.uwo.ca/sts/computer-science/gaul

Provided Files

Skeleton code is provided to you as "assingment2.c" file. You need to complete the "assignment2.c". Implementation of *summation(), ith_part_start()* and *ith_part_end()* are already provided in the assignment2.c. DO NOT make any changes to the implementation of *summation(), ith_part_start()* and *ith_part_end().*

You need to submit only one C file. The name of your submitted C file must be **"assignment2.c"**. Marks will be deducted if your submitted C file name is different. You must submit your assignment through OWL. Be sure to test your code on one of MC 244 systems (see "Computing Platform for Assignments" section above). **Marks will be deducted** if your program fails to compile, or your program runs into errors on the computing platform mentioned above.

If you have a question and you would like to email, please email your designated TA.

Good luck!!