CS 3305A: Operating Systems
Department of Computer Science
Western University
Assignment 1
Fall 2023
Due Date: September 27, 2023

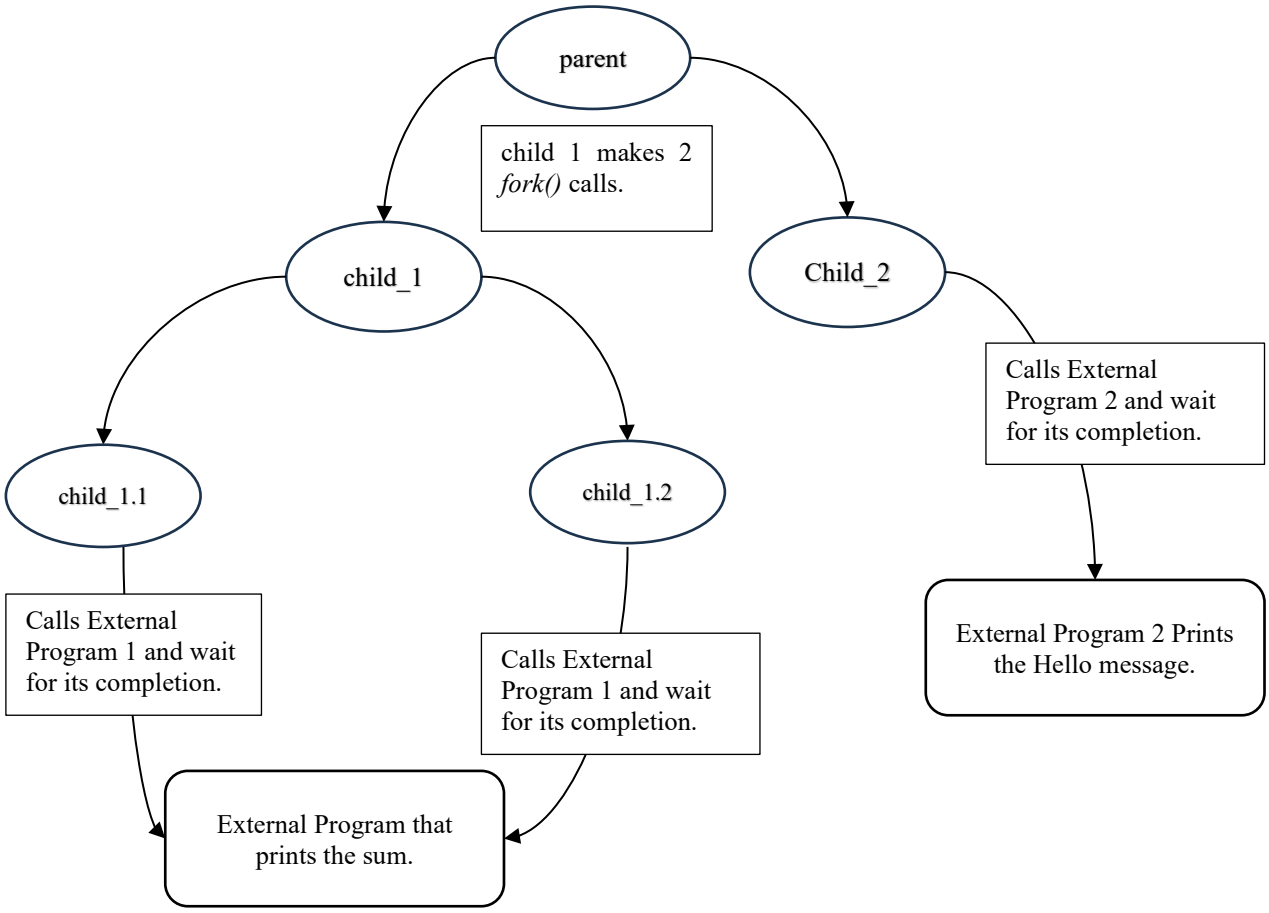
Purpose

The goals of this assignment are the following:

- Learn about process creation and their control using C in Linux environment.
- Get experience with the *fork(), wait()* and *execl()* system functions.
- Gain more experience with the C programming language from an OS perspective.

Assignment-1: Parent and Child Processes (100 points)

Develop a C program that calculates the sums of natural numbers from 1 to N and from 1 to M using two separate 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 (i.e., the parent process) will initiate the creation of a child process (e.g., child_1).
2. The parent process will wait for child_1 to complete before forking (creating) child_2.
3. The parent process will wait for child_1 and child_2 to complete before completing its own execution.
4. child_1 will invoke the fork() function to create child_1.1 and then once the child_1.1 is completed, child_1 will create child_1.2.
5. child_1.1 and child_1.2 both will execute an external program named *"*external_program1.*out"* (the source code for this file, *external_program1.c,* will be provided to you).
6. For child_1.1 you must pass N as an argument to the external program (hint: use *execl()*). As a result of this system call, child_1.1 will be replaced by *"external_program1.out."* which will print the sum from 1 to N.
7. Similarly, for child_1.2 you must pass M as an argument to the external program (hint: use *execl()*). As a result of this system call, child_1.2 will also be replaced by *"external_program1.out"* which will print the sum from 1 to M.
8. Upon the completion of child_1.1 and child_1.2 processes, child_1 will complete its own execution.
9. At this point, child_1 should be completed. The parent will now fork child_2 and wait for the completion of child_2.
10. child_2 will make a call to the "external_program2.out" (the source code for this file, *external_program2.c,* will be provided to you). child_2 must pass a string namely (S) [where S is your First name] to the external program. As a result of this system call, child_2 will be replaced by external_program2.out (hint: *execl()*).
11. External program called by Child_2 will display "Hello S, Thanks for your effort!" message and terminate.
12. At this point, child_2 will also complete its execution and hence no remaining jobs to complete for the parent process at this point.
13. The parent process will now terminate.

The path to the external programs, *"external_program1.out", "external_program2.out" and* variables *M, N* and message *S* must be passed to the main() program as a command line argument (hint: utilize argc and argv) where:

$$\mathbf{1} \leq \boldsymbol{M} \leq \mathbf{100}$$
$$\mathbf{1} \leq \boldsymbol{N} \leq \mathbf{100}$$
$$\mathbf{1} \leq \boldsymbol{L} \leq \mathbf{20} \ \boldsymbol{Where\ L\ is\ the\ length\ of\ message\ S}$$

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

Example/sample Input:

N = 12
M = 5
S = "Sudipto"

Output:

parent (PID 5769): process started

parent (PID 5769): forking child_1

parent (PID 5769): fork successful for child_1 (PID 5770)
parent (PID 5769): waiting for child_1 (PID 5770) to complete
child_1 (PID 5770): process started from parent (PID 5769)

child_1 (PID 5770): forking child_1.1
child_1.1 (PID 5771): process started from child_1 (PID 5770)
child_1.1 (PID 5771): calling an external program [./external_program1.out]
child (PID 5771): external_program1: sum: [1 - 12] = 78
child_1 (PID 5770): completed child_1.1

child_1 (PID 5770): forking child_1.2
child_1.2 (PID 5772): process started from child_1 (PID 5770)
child_1.2 (PID 5772): calling an external program [./external_program1.out]
child (PID 5772): external_program1: sum: [1 - 5] = 15
child_1 (PID 5770): completed child_1.2

parent (PID 5769): forking child_2
parent (PID 5769): fork successful for child_2 (PID 5773)
child_2 (PID 5773): process started from parent (PID 5769)
child_2 (PID 5773): calling an external program [./external_program2.out]

child (PID 5773): external_program2: Hello Sudipto, Thanks for your effort!
parent (PID 5769): completed parent

Note that the output above is an example to illustrate how the output sequence must look like. The output will change depending on the parameter passed (e.g., N, M, S) to the main() program.
Hints: fork(), wait(), getpid(), getppid(), execl()

Mark Distribution (100 points)

1. A parent process will create two child processes: 10 points
2. Parent will wait for child_1 to complete before creating child_2: 15 points
3. child_1 will create two child child_1.1 and child_1.2: 15 points
4. child_1.1 make a system all to external_program1: 20 points
5. child_1.2 make a system all to external_program1: 20 points
6. child_2 will make a system call to external_program2: 10 points
7. Parent process must not terminate until all child processes are completed: 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

The source codes for the external programs, "external_program1.out" and "external_program2.out" are provided to you as "external_program1.c" and "external_program2.c", respectively. DO NOT make any changes to these external programs.

You need to submit only one C file. The name of your submitted C file must be "assignment1.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!!