# CS 3305A: Operating Systems
**Department of Computer Science**
**Western University**
**Assignment 1**
**Fall 2022**
**Due Date: September 28, 2022**

## Purpose

The goals of this assignment are the following:

- Learn about process creation and control 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)

Write a program in C that will perform the following tasks (must follow the task sequence/order below):

1. Your main program (i.e., parent process) will fork (create) a child process (e.g., child_1).
2. The parent process will wait for child_1 to complete before forking (creating) child_2.
3. child_1 will fork its own child_1.1 and wait for its completion.
4. child_1.1 will call an external program “external_program.out” (the source code of this file *external_program.c* will be provided to you) and must pass its PID concatenated with the string “for child_1.1” to the external program “external_program.out” (hint: execl()). As a result of this system call, child_1.1 will be replaced by external_program.out. The path to the external program “external_program.out” should be passed into the main program as a **command line argument** (hint: argc, argv).
5. After completion of child_1.1 process, child_1 should be completed, as no more jobs remain for child_1.
6. The parent will now fork child_2, and wait for the completion of child_2.
7. child_2 will make a call to the same external program “external_program.out”. child_2 must pass its PID concatenated with the string “for child_2” to the external program “external_program.out”. As a result of this system call, child_2 will be replaced by external_program.out (hint: execl()). The path to the external program “external_program.out” should be passed into the main program as a **command line argument** (hint: argc, argv).
8. The parent process will now terminate.

**The expected output from your program should look like the following:**

```
parent (PID 1655): process started
parent (PID 1655): forking child_1
parent (PID 1655): fork successful for child_1 (PID 1656)
parent (PID 1655): waiting for child_1 (PID 1656) to complete
child_1 (PID 1656): forking child_1.1
child_1 (PID 1656): fork success for child_1.1 (PID 1657)
child_1 (PID 1656): waiting for child_1.1 (PID 1657) to complete
child_1.1 (PID 1657): calling an external program [external_program.out]
From the external program: The PID was 1657 for child_1.1
child_1 (PID 1656): completed
```

```
parent (PID 1655): forking child_2
parent (PID 1655): fork successful for child_2 (PID 1658)
parent (PID 1655): waiting for child_2 (PID 1658) to complete
child_2 (PID 1658): calling an external program [external_program.out]
From the external program: The PID was 1658 for child_2
parent (PID 1655): completed
```

*Hints: fork(), wait(), getpid(), getppid(), execl(), strcat()*

## Mark Distribution

This section describes a tentative allocation of marks assigned for the desired features. **(100 points)**

a) A parent process will create two child processes: 20 points
b) parent will wait for child_1 to complete before creating child_2: 15 points
c) child_1 will create its own child child_1.1: 15 points
d) child_1.1 will make a system call to an external program: 15 points
e) child_2 will make a system call to an external program: 15 points
f) parent process must not terminate until all child processes are completed: 20 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 have to SSH to **compute.gaul.csd.uwo.ca** first (this server is also known as sylvia.gaul.csd.uwo.ca, in honour 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 code for the external program "external_program.out" is provided to you as "external_program.c". DO NOT make any changes to "external_program.c"
- When running the program, you must provide the path to "external_program.out" as an argument (see Assignment 1 tutorial powerpoint)
- If you have any questions, please contact the designated TAs or the Instructor

## Assignment Submission

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.

Assignment 1 FAQ will be made available on OWL. Also, consult TAs, and the Instructor for any questions you may have regarding this assignment.