Queen's School of Computing
CISC324 – Winter 2024
Instructor: Dr. Anwar Hossain
Lab 1 - Due Date:

Lab 1

This lab has been prepared to be conducted on any operating system (Linux, MacOS, and Windows). Please make sure to have a well-set-up environment. You will need a computer, any operating system, and any Python interpreter of your choice.

Background and Outcomes

During this lab, you will learn how to use the most important process management system calls. You will be writing programs, using Python, to create new processes, terminate running processes, wait for processes (synchronization), change the code segment of processes, and get the attributes of processes. In the following, we will learn some of the system calls that are used in this regard. We will be using the OS module of Python. OS module provides functions for interacting with the operating system. OS module comes under Python's standard utility modules. This module provides a portable way of using operating system-dependent functionality:

1- os.fork(): method in Python is used to create a child process. This method works by calling the underlying OS function fork(). This method returns 0 in the child process and the child's process ID in the parent process. **IN ORDER TO USE THIS FUNCTION AND YOU HAVE A WINDOWS OPERATING SYSTEM, YOU NEED TO INSTALL CYGWIN AND INSTALL PYTHON FOR CYGWIN FROM CYGWIN INSTALLATION PROMPT.**

2- os.getpid(): The os.getpid() method returns the process id of the current process. This method returns a integer value denoting process ID of current process. The return type of this method is of class 'int'.

3- exit(): sys.exit() is a built-in function in the Python sys module that allows us to end the execution of the program.

4- os.wait(): In Python, you can find the wait() function in two different modules: the os module and the threading module. Whereas the os module also has the same working, but in the os module, it works with the parent process to wait until the child completes its execution. In the below section, let us both these methods in detail with examples. This syntax returns the child process's id in the form of a tuple and a 16-bit integer which is also present in the tuple to denote the exit status. This method returns a 16-bit integer which in turn includes higher and lower bytes where the lower byte has the signal number as zero, which will kill the process, and the higher byte will have the exit status notification. This os.wait() function does not take any parameters or arguments

Queen's School of Computing
CISC324 – Winter 2024
Instructor: Dr. Anwar Hossain
Lab 1 - Due Date:

Prepare your environment!

The boilerplate code, provided to you, can work on any environment (Linux, Windows, MacOS, etc.) so please feel free to search for how to install Python on your machine.

IDEs that you can use (feel free to use any other IDE)

1. Visual Studio Community Edition for Windows-based machines only (it's something different from VS Code).
2. Visual Studio Code (Can work on any operating system).
3. PyCharm (Can work on any operating system).

Install Python on Your Machine

There are too many methods to install Python (feel free to use any other method):

1. Download and install Anaconda (which works on any operating system).
2. Download and install Python from the official website (works on any operating system).

Download the Boilerplate Code

There are two methods to download the boilerplate code:

1. Download the whole repository of the course (Zip file) and open the folder named "Lab 1".
2. If you are familiar with Git and Git commands, you can clone the repository or you can update your local repository if you already have it cloned previously.

Process Communication Issue

Assume that we possess a multiprocessing computer and that we would like to compute, using a computer program the sum of a sequence from 0 to n (see equation 1), where $n > 0$. To speed up the computations we would like to implement our program in such a way so that we make use of multiprocessing. A simple intuition consists of dividing the sum into two parts that will be run by two different processes. Let us say process P_1 executes the sum from 0 to $[n/2]$, and process P_2 executes the sum from $[n/2] + 1$ to n. Process P_1 is set the task to display the final result. The file "process_com.py" is a typical implementation of this scenario using the Python language.

$$\sum_{i=0}^{n} i = 0 + 1 + 2 + \cdots + n \qquad (1)$$

1- Compile and execute the program for different values. The program appears to be returning incorrect computations (e.g., for $n = 1$ it returns 0). Why is that?

Queen's School of Computing
CISC324 – Winter 2024
Instructor: Dr. Anwar Hossain
Lab 1 - Due Date:

2- Modify the program code to fix the issue using "wait()" and exit() system calls. Explain how you fixed the issue.
3- After fixing the issue, you may notice that starting from a certain value n, the returned sum becomes incorrect. What is the value of n? Explain the reason behind this limitation.
4- If we switch the function of the parent and child process (i.e., A(.) by B(.) and B(.) by A(.) in the source code), you may notice that starting from a certain value n, the returned sum becomes incorrect. What would be the value of n?

What to submit?

1. Place all your source codes in a folder named in the following format:

   324-1234-Lab1, where 1234 stands for the last 4 digits of your student ID, e.g.: For student with ID 20196072, the folder should be named 324-6072-Lab1.

2. Place a ReadMe.txt file into the same folder above.
3. Compress the above folder using Zip (the extension must be .zip or .rar).
4. Log into OnQ, locate the lab's dropbox, and upload the compressed folder.