Queen's School of Computing
CISC324 – Winter 2024
Instructor: Dr. Anwar Hossain
Lab 2 - Due Date: Feb. 12, 2024

Lab 2: In-group

Think about the following situation and organize the work based on it. By doing so, you'll gain practical understanding of how various scheduling algorithms can be applied in diverse system environments. You'll also see how the selection of a specific algorithm influences both the efficiency and the equitable distribution of system resources.

Scenario: Processing Requests Bound for a Web Server

In this advanced scenario, students are presented with the intricate task of constructing a simulated web server. This web server's critical role is to manage a multitude of incoming requests from a diverse array of clients spread across different locations. The inherent complexity in this task arises from the fact that each client request is unique, not only in content but also in the computational effort required for processing. Some requests might be simple and quick to handle, while others could be complex, requiring more time and resources.

The challenge lies in the development and implementation of a sophisticated scheduling algorithm. This algorithm should not be just any routine scheduler; it must be intelligently designed to prioritize and process these varied requests in a manner that is both efficient – meaning it maximizes the throughput of the server – and fair – ensuring that no single client's request is consistently favored or neglected over others.

The exercise aims to provide a holistic learning experience where students will not only understand the technical aspects of web server operations but also grapple with the complexities involved in decision-making algorithms. The ultimate goal is to minimize the waiting time for each request, thereby enhancing the overall user experience. This is a balancing act: ensuring quick response times while maintaining an equitable distribution of the server's resources among all clients.

Queen's School of Computing
CISC324 – Winter 2024
Instructor: Dr. Anwar Hossain
Lab 2 - Due Date: Feb. 12, 2024

Through this exercise, students will gain valuable insights into real-world scenarios and the importance of efficient resource allocation in high-demand, distributed computing environments. The whole process of server-based request handling is introduced here to mimic the scheduling algorithms that are an integral part of major operating systems.

Instructions, assumptions and related information:

1. Each client request is a process. The server can handle one request at a time.
2. Assume there are several client requests, arriving randomly at the server at different times (arrival time), requiring different burst time (execution time).
3. Implement the algorithm to schedule and execute processes in a round-robin fashion with a specified time quantum.
4. Run the simulation program for different set of processes, using different time quantum values, and record the results, including average waiting time and average turnaround time for the requests.
5. Implement a second scheduling algorithm (pre-emptive or non pre-emptive) and run the test for the same dataset and record the result in terms of similar parameters used above.

Data Analysis:

1. Analyze the data collected from the experiments.
2. Compare the results of round robin approach to the other algorithm you implemented. What do you notice in the results of the two approaches? Write your observation.

Prepare your environment!

The boilerplate code 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. You can use any other IDE of choice.

Install C++ on Your Machine

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

1. C/C++ for Visual Studio Code. (which works on any operating system).
2. Using Visual Studio to Build C++ apps. (Windows only).
3. Pycharm.

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 2".
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 from the previous lab.

What to submit?

1. Place all your source codes and the readme.txt file in a folder named in the following format:

   324-group**X**-Lab**2**, where X stands for the group number, e.g.: example folder name for group 1 is 324-group**1**-Lab**2**.

   Note: The **readme.pdf** should report the results and elaborate the results of the comparison mentioned in the Data Analysis section. Also put a table to add all the member's contribution in the lab.

2. Compress the above folder using Zip (the extension must be .zip or .rar).
3. Log into OnQ, locate the lab's dropbox in OnQ, and upload the compressed folder.