**Technological Institute of the Philippines**

**938 Aurora Blvd. Cubao, Quezon City**

**College of Computer Studies**

**CS 007 - Parallel and Distributed Computing**

**Prelim Period**

| **Name:** Tan, Jose Mari V. | **Date:** June 3, 2024 |
| --- | --- |
| **Program / Section**: BSCS / CS33S1 | **Instructor:** Ma’am Janice Capule |
| **Assignment 1.1 Serial Computing**  1. Create a table that will compare serial computing from parallel computing. (10)   |  | Serial Computing | Parallel Computing | | --- | --- | --- | | Advantages | * Cost is low * Less complex hardware * Lower overhead * Lower resource consumption | * All the instructions are executed parallelly. * It has high performance and the workload of the processor is low because multiple processors are working simultaneously. * It requires less time to complete the whole process. | | Disadvantages | * Slower than parallel computing * All the instructions are executed in a sequence, one at a time. * Has a higher processing overhead | * Cost is high * Can be difficult to debug * Increased resource consumption | | Algorithm | * Sorting algorithms * Searching algorithms | * Matrix computations * Dynamic Programming | | Implementation | * Uses single processors/threads * Sequential Execution * Linear Data Access | * Uses multiple processors/threads * Concurrent Execution * Divided Data Access |   2. Create a table that will compare each process scheduling algorithm. (10)   |  | Advantages | Disadvantages | | --- | --- | --- | | First Come First Serve | * It is simple and easy to understand. * Provides fairness by treating all processes equally and giving them an equal opportunity to run. * Guarantees that every process will eventually be executed, as long as the system has sufficient resources. * It is well-suited for processes that run for a longer period of time or workloads that do not have strict time constraints. | * The process with less execution time suffers. * The first process will get the CPU first (large burst time), other processes (less burst time) can get the CPU only after the current process has finished its execution that results in the convoy effect which lowers CPU and device utilization. | | Shortest Job First | * Shortest jobs are favored. * It is probably optimal, in that it gives the minimum average waiting time for a given set of processes. | * SJF may cause starvation if shorter processes keep coming. This problem is solved by aging. * It cannot be implemented at the level of short-term CPU scheduling. | | Shortest Remaining Time | * SRT is faster than SJF algorithm, given its overhead charges are not counted. * Allows for easier management of library updates or replacements without recompiling the program. * Enables efficient memory usage, as libraries can be shared among multiple instances of the program. | * Slightly slower program startup due to the additional linking process. * Requires proper handling of library dependencies to ensure correct execution. * Debugging can be slightly more complex, as libraries are separate entities loaded at runtime. | | Priority Scheduling | * This provides a good mechanism where the relative importance of each process may be precisely defined. * PB scheduling allows for the assignment of different priorities to processes based on their importance, urgency, or other criteria. | * If high-priority processes use up a lot of CPU time, lower-priority processes may starve and be postponed indefinitely. * Another problem is deciding which process gets which priority level assigned to it. | | Round Robin Scheduling | * Every process gets an equal share of the CPU. * It is cyclic in nature, so there is no starvation that can occur. | * Setting the quantum too short increases the overhead and lowers the CPU efficiency, but setting it too long may cause a poor response to short processes. * The average waiting time under the RRS is often long. * If the time quantum is very high then RR downgrades to FCFS. | | Multilevel Queue Scheduling | * Application of separate scheduling for various kinds of processes is possible. | * The lowest level process faces the starvation problem. | | Multilevel Feedback Queue Scheduling | * Low scheduling overhead. * Allows aging, thus no starvation. | * It’s not flexible. * It also requires some means of selecting values for all the parameters to define the best scheduler, thus it is also the most complex. |   **Synthesis:**  Choosing which type of computing to implement completely depends on resources available and the requirements needed to accomplish tasks. Serial computing is better in executing singular tasks which also cost less while parallel computing is better in executing multiple tasks simultaneously but offers high costs. The same can be applied to process scheduling algorithms. Choosing the most optimal process scheduling algorithm depends on the number and complexity of jobs and the maximum resources that can be allocated by the system.  **References:**  <https://www.geeksforgeeks.org/advantages-and-disadvantages-of-various-cpu-scheduling-algorithms/>  <https://www.geeksforgeeks.org/introduction-of-shortest-remaining-time-first-srtf-algorithm/>  <https://newhavendisplay.com/blog/serial-vs-parallel-communication/>  <https://www.geeksforgeeks.org/difference-between-sequential-and-parallel-computing/> | |

**Honor Pledge:**

*“I affirm that I have not given or received any unauthorized help on this assignment and that all work shall be my own.”*