UNIVERSITY OF TECHNOLOGY, JAMAICA
SCHOOL OF COMPUTING AND INFORMATION TECHNOLOGY

Operating Systems (CIT3002/CMP3020)

Research Assignment

Date Given: Week 6 **Research & Programming Project Date Due:** Week 13

Group Sizes: **4 students**. Additional members will have to get approval from tutor.*

Research Overview

The three objectives of Operating System design are: convenience, efficiency, and the ability to evolve. Over the years many OSes have been created, some still active, some almost gone, and others extinct. Identify two OSes of the same structure (monolithic, micro kernel, hybrid, etc.), one should be extinct and the other surviving for the past ten (10) years. Compare these two OSes based on their:

- user/computer interface
- how they manage resources
- and how it has evolve

In your conclusion you should identify why one failed and the other is still successful. Also in your conclusion you should also take a deeper look into the surviving OS and determine a prediction for its future success if any.

Report Requirements

1. Choice of operating systems 2mrks
2. For each OS
    a. Description of OS structure 5mrks
    b. OS managers: processor, device, file, memory 10mrks
    c. Reason to evolve and how it was accomplished 10mrks
3. Conclusion
    a. Why success and why failure 5mrks
    b. Prediction of surviving OS future 5mrks
4. APA Formatting and Grammar 3mrks
    (Please reference APA Style Guide to Electronic References, Sixth Edition)

.

Further Instructions

Each report should fall within the range of 7-10 pages including Title/Cover and Reference pages. Students should work in the same group for their programming project. Mark deduction will be applied for days late as follows: 25% - 1 day, 50% - 2 days, 75% - 3 days. No assignment will be accepted after the third day late, in which case the student will be awarded zero marks for the assignment.

Group Programming Assignment

Your group has been asked to simulate a small process management system which will use a priority non-preemptive scheduling scheme when executing processes on a dual-core (2 CPU's) processor system. The system manages twenty (20) processes and maintains a shared reusable resource list (list of integer pairs) that all processes will interact with; all processes perform some task on this shared resource. This prototype is aimed at demonstrating the concept of process synchronization by restricting access to the shared resource based on when processes are executing in its critical section.

At the beginning of the program the shared resource list should initialize all data values to 0. Given that multiple processes can generate the same task, and some tasks are updating the shared resource, there is a need to synchronize the execution of processes. Adding and Removing tasks modify the shared list, while retrieve and calculate tasks simply read the shared data. Any task that modifies the shared list must have mutual exclusion on the shared resource while multiple processes executing retrieve and/or calculate tasks are able to do so concurrently. Adding and removing record has priority over record retrieval and tallying. When a process that is modifying the resource list gets selected from the ready queue and is added to a CPU, it must ensure that the process on the other CPU is completed before it locks and updates the resource list based on its task – during this time that process will be blocked. The program should end when all the processes have completed.

Your team is allowed to implement the system in any of the following programming languages: Python, C, C++, C# or Java.

The ***shared resource list*** (list of integer pairs) can hold up to twenty (20) records which contains:

i. *Id* for integer (sequential list between 1 and 20)
ii. *Data* for the integer

A process can be executing any of the following **tasks** on the shared resource integer list:

i. ***Add Record*****:** randomly generates a resource record and updates that record - id (1-20) and data value (1-100)
ii. ***Remove Record*****:** randomly generates a record id (1-20) and sets that records data value to 0
iii. ***Retrieve a Record:*** randomly generates a record id then reads and outputs the data value for that id.
iv. ***Calculate Record Data Total*****:** display the sum of all data values in the share resource list

A **process** should contain:

i. *PID* (uniquely assigned between 1 and 20)
ii. ***Task*** (a randomly select option of the four above)
iii. *Priority* (integer value 1 – 5 with 1 being the highest)
iv. *Arrival time* (randomized number between 0 and 29)
v. *Blocked time* (numeric value)
vi. *Burst time* (randomized number between 1 to 5 units of time)

Expectation(s):

1. Process data structure must be implemented independent of the Shared Resource List data structure
2. Multiple processes can be generated with the same task
3. Implement the scheduling algorithm
4. Appropriate locking mechanism is used
5. Regular details of all the processes as well as the state of the shared resource list at key intervals should be displayed during program execution so verification can be done

Marking Scheme:

Documentation

a) Team Work Breakdown Structure 4mks
b) Screenshots of program execution 3mks
c) Description of how simulation works 3mks

Program

a) Run program from the terminal 3mks
b) Data structure definitions (collections or ADT's) 5mks
c) Process scheduling algorithm 10mks
d) Process synchronization algorithm 10mks
    - Simulation of concurrent process execution
    - Handling of the critical section problem
e) A running display of the state of the processes and shared resource list 3mks
f) Error Handling techniques 2mks
g) Functionality 7mks

Submission & Penalties:

Mark deduction will be applied for days late as follows: 25% - 1 day, 50% - 2 days, 75% - 3 days. No assignment will be accepted after the third day late, in which case the student will be awarded zero marks for the assignment.