**University of San Carlos**

**School of Arts and Sciences  
Department of Computer and Information Sciences  
Talamban Campus**

**CS 3104– Operating System**

**Assignment # 3**

**1st Semester 2023-2024**

Submitted by: Chiu, Joshua Patrick G.

Instructor: Godwin S. Monserate, MS, CCNA/CCAI

09989182612 21100768@usc.edu.ph

Contact Number, Email address

Submitted by : Chiu, Joshua Patrick G.   
Assignment Question# : 1 - 4

Topic : Processes Score :  
Due Date : October 2, 2023  
Date Submitted : September 23, 2023

Question here:

**1.** Describe the differences among short-term, medium-term, and long- term scheduling.

**2.** Describe the actions taken by a kernel to context-switch between processes.

**3.** Consider the RPC mechanism. Describe the undesirable circumstances that could arise from not enforcing either the "at most once" or "exactly once" semantics. Describe possible uses for a mechanism that had neither of these guarantees.

**4.** What are the beneﬁts and detriments of each of the following? Consider both the systems and the programmers’ levels.

a. Symmetric and asymmetric communication

b. Automatic and explicit buffering

c. Send by copy and send by reference

d. Fixed-sized and variable-sized messages

------------------------------------------------------------------------------------------------------------------------------------------

Answer here:

1. Differences Among Short-term, Medium-term, and Long-term Scheduling: Short-term scheduling revolves around immediate decisions for process execution to ensure smooth operations. It focuses on selecting the next process from the ready queue to optimize CPU utilization and minimize response time. Medium-term scheduling, conversely, tackles memory management by determining which processes should be swapped in and out of main memory, aiding in memory optimization, especially when memory resources are limited. Long-term scheduling has a broader perspective, managing the overall system load by deciding how many new processes should be admitted to the system, considering resource availability and system load to prevent overloading.
2. Actions Taken by a Kernel for Process Context-Switching: When a kernel initiates a context switch between processes, it follows a structured process. Initially, the kernel saves the state of the currently running process, which encompasses essential data such as CPU registers and program counters. Subsequently, it retrieves and loads the state of the process scheduled to run next from memory. Finally, the kernel updates the CPU's state to initiate the execution of the newly loaded process, effectively transferring control from the previous process to the new one.
3. RPC Mechanism and Semantics: The absence of "at most once" or "exactly once" semantics in Remote Procedure Calls (RPCs) can lead to undesirable outcomes. It may result in RPC call duplications, causing unintended side effects and data inconsistencies when the same operation is executed multiple times. Conversely, RPC calls may fail to be executed altogether, resulting in inconsistencies within distributed systems. However, mechanisms without these guarantees may find utility in scenarios where data consistency is not paramount, such as non-essential data synchronization or logging. They can also be preferable in highly latency-sensitive applications where minimizing overhead is critical.

a. Symmetric and Asymmetric Communication:

Benefits: Symmetric communication allows for equal participation of both parties, which can simplify coordination. Asymmetric communication can offer better control and security.

Detriments: Symmetric communication can lead to complex synchronization. Asymmetric communication may require one party to bear more responsibility.

b. Automatic and Explicit Buffering:

Benefits: Automatic buffering reduces sender wait times and can handle variable message sizes. Explicit buffering offers control over buffer size, reducing resource wastage.

Detriments: Automatic buffering might lead to resource wastage due to its dynamic nature. Explicit buffering can introduce sender delays when the buffer is full.

c. Send by Copy and Send by Reference:

Benefits: Send by Copy ensures data integrity but consumes more memory. Send by Reference conserves memory but may introduce data consistency challenges.

Detriments: Send by Copy can be memory-intensive. Send by Reference might lead to unintended data modifications.

d. Fixed-sized and Variable-sized Messages:

Benefits: Fixed-sized messages simplify buffer management and can be more efficient for certain tasks. Variable-sized messages accommodate data of varying sizes.

Detriments: Fixed-sized messages can waste space for smaller messages. Variable-sized messages can introduce complexity in buffer allocation and management.