EE6253 - Operating Systems & Network Programming

Yugani Gamlath
Department of Electrical and Information Engineering
Faculty of Engineering
University of Ruhuna

Lecture 04 – Process Synchronization

Intended Learning Outcomes

- What is Synchronization in OS?
- Explanation using Producer Consumer Problem
- Race Condition
- The Critical-Section Problem
- Requirements of Synchronization
- Synchronization Mechanisms
- Classical Problems of Synchronization

What is Synchronization in OS?

- Processes Synchronization is the way by which processes that share the same memory space are managed in an OS.

- It helps maintain the consistency of data by using variables or hardware so that only one process can make changes to the shared memory at a time.

What is Synchronization in OS?

Processes are categorized as one of the following two types:

- *Independent Process:* The execution of one process does not affect the execution of other processes.
- *Cooperative Process:* A process that can affect or be affected by other processes executing in the system.

Process synchronization problem arises in the case of Cooperative processes also because resources are shared in Cooperative processes.

What is Synchronization in OS?

Concurrent access to shared data may result in data inconsistency.

The orderly execution of cooperating processes that share a logical address space is a best way to ensure that problem. Then data consistency is maintained.

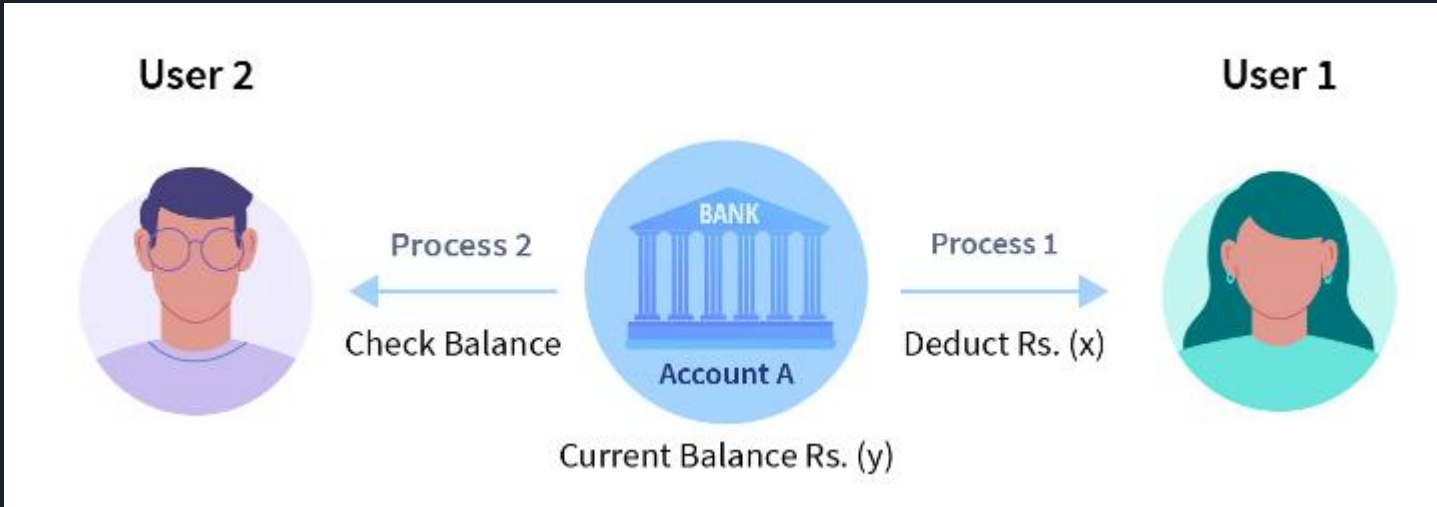
- The main objective of process synchronization is to ensure that multiple processes access shared resources without interfering with each other and to prevent the possibility of inconsistent data due to concurrent access.

- Process synchronization is an important aspect of modern OS, and it plays a significant role in ensuring the efficient functioning of multi-process systems.

What is Process Synchronization in OS?

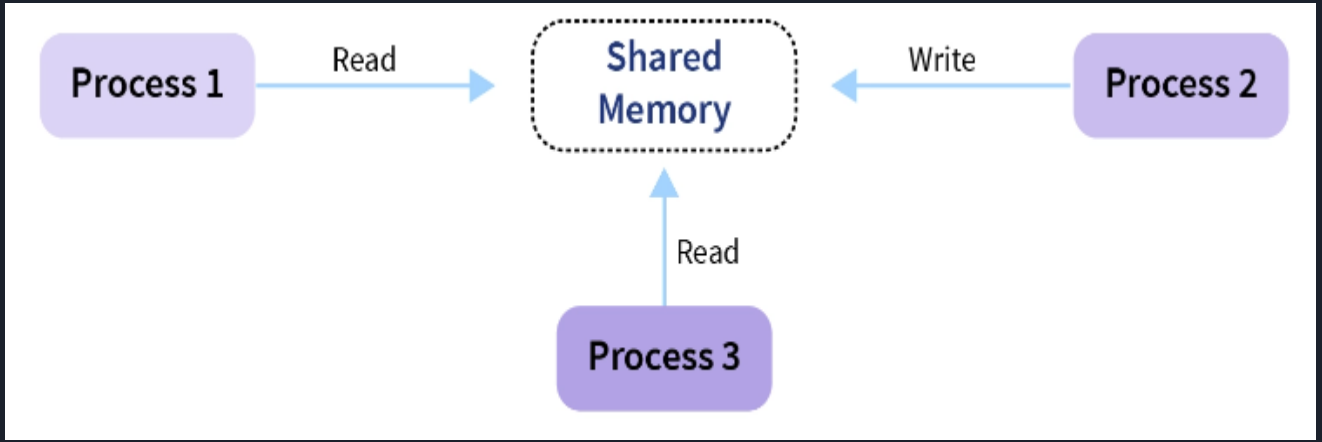
- Consider a bank that stores the account balance of each customer in the same database.
- Now suppose you initially have x rupees in your account.
- Now, you take out some amount of money from your bank account, and at the same time, someone tries to look at the amount of money stored in your account.
- As you are taking out some money from your account, after the transaction, the total balance left will be lower than x.
- But the transaction takes time, and hence the person reads x as your account balance which leads to inconsistent data.
- If in some way, we could make sure that only one process occurs at a time, we could ensure consistent data.

What is Process Synchronization in OS?



Inconsistency of data can occur when various processes share a common resource in a system which is why there is a need for process synchronization in OS.

What is Process Synchronization in OS?



Explanation using Producer Consumer Problem

- Producer produces information that is consumed by a consumer.
- Producer-consumer problem uses shared memory.
- To allow producer and consumer processes to run concurrently, we must have available a buffer of items that can be filled by the producer and emptied by the consumer.
- This buffer will reside in a region of memory that is shared by the producer and consumer processes.
- A producer can produce one item while the consumer is consuming another item.
- The producer & consumer must be synchronized, so that the consumer does not try to consume an item that has not yet been produced.

Explanation using Producer Consumer Problem

2 Types of Buffers:

Unbounded Buffer:

- Place no practical limit on the size of the buffer.
- The consumer may have to wait for new items, but the producer can always produce new items.

Bounded buffer:

- Assumes a fixed buffer size.
- In this case, the consumer must wait if the buffer is empty, and the producer must wait if the buffer is full.

Explanation using Producer Consumer Problem

- Counter variable = 0
- Counter is incremented every time we add a new item to the buffer Counter++
- Counter is decremented every time we remove one item from the buffer counter--

----------Example----------

- Suppose that the value of the variable counter is currently 5.
- The producer and consumer processes execute the statements "counter++" and "counter--" concurrently.
- Following the execution of these two statements, the value of the variable counter may be 4, 5, or 6!
- The only correct result, though, is counter == 5, which is generated correctly if the producer and consumer execute separately.

Explanation using Producer Consumer Problem

"counter++" may be implemented in machine language (on a typical machine) as:

```
register1 = counter
register1 = register1 + 1
counter   = register1
```

"counter--" may be implemented in machine language (on a typical machine) as:

```
register2 = counter
register2 = register2 - 1
counter   = register2
```

T_0:	producer	execute	$register_1$	=	counter	{ $register_1$ = 5 }
T_1:	producer	execute	$register_1$	=	$register_1 + 1$	{ $register_1$ = 6 }
T_2:	consumer	execute	$register_2$	=	counter	{ $register_2$ = 5 }
T_3:	consumer	execute	$register_2$	=	$register_2 - 1$	{ $register_2$ = 4 }
T_4:	producer	execute	counter	=	$register_1$	{ counter = 6 }
T_5:	consumer	execute	counter	=	$register_2$	{ counter = 4 }

Race Condition

- A situation like previous, where several processes access and manipulate the same data concurrently and the outcome of the execution depends on the particular order in which the access takes place, is called race condition.

- Since many processes use the same data, the results of the processes may depend on the order of their execution.

The Critical-Section Problem

https://youtu.be/6x_XMDCMyAk?feature=shared

The Critical-Section Problem

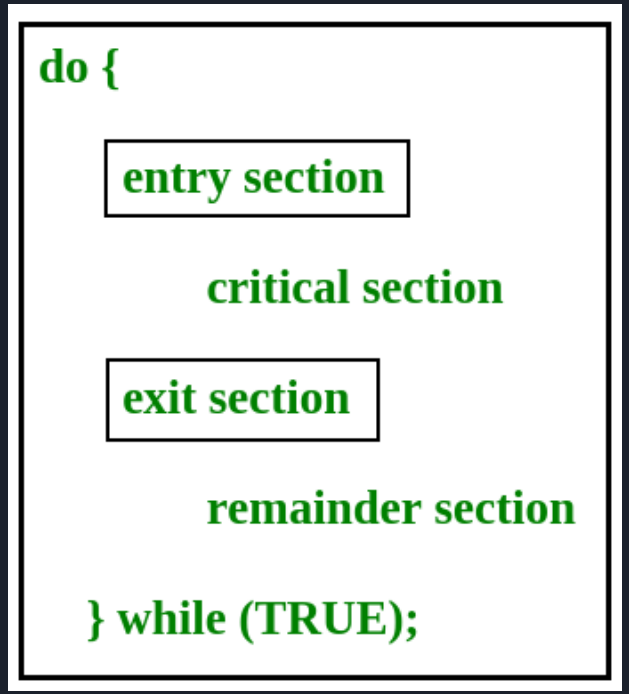
- Consider a system consisting of a n number of processes.
- Each process has a segment of code, called a critical-section in which the process may be changing common variables, updating a file, writing a file, and so on.
- When one process is executing in its critical section, no other process is to be allowed to execute in its critical section.
- Then no 2 processes are executing in their critical sections at the same time.
- The critical section problem is to design a protocol that the corporate processes.

The Critical-Section Problem

Different Sections

- Entry Section: This part of code checks whether a process is allowed to enter the critical section. The process requests permission and waits if another process is already using the shared resource.
- Critical Section: The Critical section allows and makes sure that only one process is modifying the shared data.
- Exit Section: After finishing work in the critical section, the process leaves and signals that it is done. This allows other waiting processes to enter their critical sections.
- Remainder Section: This contains all other code that does not access shared data. The process can execute this part freely without restrictions.

The Critical-Section Problem



Requirements of Synchronization (Requirements of Critical-Section Solution)

- Mutual exclusion: If a process is running in the critical section, no other process should be allowed to run in that section at that time.

- Progress: If no process is still in the critical section and other processes are waiting outside the critical section to execute, then any one of the threads must be permitted to enter the critical section. The decision of which process will enter the critical section will be taken by only those processes that are not executing in the remaining section.

- Bounded waiting: There exists a limit as to how many other processes can get into their critical sections after a process requests entry into their critical section and before that request is granted.

Synchronization Mechanisms

These mechanisms can assist in the solving of critical section issues.

Mutex Locks

- Mutex is a locking mechanism used to synchronize access to a resource in the critical section.
- In this method, we use a LOCK over the critical section.
- The LOCK is set when a process enters from the entry section, and it gets unset when the process exits from the exit section.

Synchronization Mechanisms

How a Mutex Works?

- **Before entering the critical section (Entry Section)**
  - A process tries to **acquire the lock**.
  - If the lock is already held by another process, it must **wait**.
- **Inside the critical section**
  - The process that holds the lock can **safely access or modify shared data**.
  - No other process can enter until the lock is released.
- **After leaving the critical section (Exit Section)**
  - The process **releases (unsets) the lock**.
  - This allows another waiting process to acquire the lock and enter.

Synchronization Mechanisms

Semaphores

- A semaphore is a signaling mechanism, and a process can signal a process that is waiting on a semaphore.
- This differs from a mutex in that the mutex can only be notified by the process that sets the shared lock.
- Semaphores make use of the wait() and signal() functions for synchronization among the processes.

Reference: https://youtu.be/LIzTbA3cAWY?si=kwTR44Dh3IxM1wks

Classical Problems of Synchronization

- Bunded Buffer Problem / Producer-Consumer Problem
- Dining Philosophers Problem
- Readers Writers Problem

Classical Problems of Synchronization
Bounded Buffer Problem

https://youtu.be/kG_vsZZR-xY?feature=shared

Classical Problems of Synchronization
Bounded Buffer Problem

- The Bounded Buffer Problem is one of the classic problems of synchronization.
- There is a buffer of n slots and each slot is capable of storing one unit of data.
- There are 2 processes running namely producer and consumer, which are operating on the buffer.

Classical Problems of Synchronization
Bounded Buffer Problem

- Producer tries to insert data into an empty slot of the buffer.
- Consumer tries to remove data from a filled slot in the buffer.
- Producer must not insert data when the buffer is full.
- Consumer must not remove data when the buffer is empty.
- Producer and consumer should not insert and remove data simultaneously.
- Semaphores can be used to solve this problem.

Classical Problems of Synchronization References

https://youtu.be/VSkvwzqo-Pk?feature=shared

https://youtu.be/Uonka8vLCJ8?feature=shared

Self-Learning Task

Search about the following 2 problems of synchronization with suitable solutions and brief the idea into single page document.

- Dining Philosophers Problem
- Readers Writers Problem

Thank You...