Process Synchronization: Critical Section Problem in OS

What is Process Synchronization?

Process Synchronization is the task of coordinating the execution of processes in a way that no two processes can have access to the same shared data and resources.

It is specially needed in a multi-process system when multiple processes are running together, and more than one processes try to gain access to the same shared resource or data at the same time.

This can lead to the inconsistency of shared data. So the change made by one process not necessarily reflected when other processes accessed the same shared data. To avoid this type of inconsistency of data, the processes need to be synchronized with each other.

In this operating system tutorial, you will learn:

How Process Synchronization Works?

For Example, process A changing the data in a memory location while another process B is trying to read the data from the **same** memory location. There is a high probability that data read by the second process will be erroneous.

(//cdn.guru99.com/images/1/122319_0848_ProcessSync1.png)

Sections of a Program

Here, are four essential elements of the critical section:

- **Entry Section:** It is part of the process which decides the entry of a particular process.
- **Critical Section:** This part allows one process to enter and modify the shared variable.
- **Exit Section:** Exit section allows the other process that are waiting in the Entry Section, to enter into the Critical Sections. It also checks that a process that finished its execution should be removed through this Section.
- **Remainder Section:** All other parts of the Code, which is not in Critical, Entry, and Exit Section, are known as the Remainder Section.

What is Critical Section Problem?

A critical section is a segment of code which can be accessed by a signal process at a specific point of time. The section consists of shared data resources that required to be accessed by other processes.

- The entry to the critical section is handled by the wait() function, and it is represented as P().
- The exit from a critical section is controlled by the signal() function, represented as V().

In the critical section, only a single process can be executed. Other processes, waiting to execute their critical section, need to wait until the current process completes its execution.

Rules for Critical Section

The critical section need to must enforce all three rules:

- **Mutual Exclusion:** Mutual Exclusion is a special type of binary semaphore which is used for controlling access to the shared resource. It includes a priority inheritance mechanism to avoid extended priority inversion problems. Not more than one process can execute in its critical section at one time.
- **Progress:** This solution is used when no one is in the critical section, and someone wants in. Then those processes not in their reminder section should decide who should go in, in a finite time.
- **Bound Waiting:** When a process makes a request for getting into critical section, there is a specific limit about number of processes can get into their critical section. So, when the limit is reached, the system must allow request to the process to get into its critical section.

Solutions To The Critical Section

In Process Synchronization, critical section plays the main role so that the problem must be solved.

Here are some widely used methods to solve the critical section problem.

Peterson Solution

Peterson's solution is widely used solution to critical section problems. This algorithm was developed by a computer scientist Peterson that's why it is named as a Peterson's solution.

In this solution, when a process is executing in a critical state, then the other process only executes the rest of the code, and the opposite can happen. This method also helps to make sure that only a single process runs in the critical section at a specific time.

Example

(//cdn.guru99.com/images/1/122319_0848_ProcessSync2.png)

```
PROCESS Pi
FLAG[i] = true
while( (turn != i) AND (CS is !free) ){ wait;
}
CRITICAL SECTION FLAG[i] = false
turn = j; //choose another process to go to CS
```

- Assume there are N processes (P1, P2, ... PN) and every process at some point of time requires to enter the Critical Section

- A FLAG[] array of size N is maintained which is by default false. So, whenever a process requires to enter the critical section, it has to set its flag as true. For example, If Pi wants to enter it will set FLAG[i]=TRUE.
- Another variable called TURN indicates the process number which is currently wating to enter into the CS.
- The process which enters into the critical section while exiting would change the TURN to another number from the list of ready processes.
- Example: turn is 2 then P2 enters the Critical section and while exiting turn=3 and therefore P3 breaks out of wait loop.

Synchronization Hardware

Some times the problems of the Critical Section are also resolved by hardware. Some operating system offers a lock functionality where a Process acquires a lock when entering the Critical section and releases the lock after leaving it.

So when another process is trying to enter the critical section, it will not be able to enter as it is locked. It can only do so if it is free by acquiring the lock itself.

Mutex Locks

Synchronization hardware not simple method to implement for everyone, so strict software method known as Mutex Locks was also introduced.

In this approach, in the entry section of code, a LOCK is obtained over the critical resources used inside the critical section. In the exit section that lock is released.

Semaphore Solution

Semaphore is simply a variable that is non-negative and shared between threads. It is another algorithm or solution to the critical section problem. It is a signaling mechanism and a thread that is waiting on a semaphore, which can be signaled by another thread.

It uses two atomic operations, 1)wait, and 2) signal for the process synchronization.

Example

```
WAIT ( S ):
while ( S <= 0 );
S = S - 1;
SIGNAL ( S ):
S = S + 1;
```

Summary:

- Process synchronization is the task of coordinating the execution of processes in a way that no two processes can have access to the same shared data and resources.
- Four elements of critical section are 1) Entry section 2) Critical section 3) Exit section 4) Reminder section
- A critical section is a segment of code which can be accessed by a signal process at a specific point of time.
- Three must rules which must enforce by critical section are : 1) Mutual Exclusion 2) Process solution 3)Bound waiting
- Mutual Exclusion is a special type of binary semaphore which is used for controlling access to the shared resource.
- Process solution is used when no one is in the critical section, and someone wants in.
- In bound waiting solution, after a process makes a request for getting into its critical section, there is a limit for how many other processes can get into their critical section.

- Peterson's solution is widely used solution to critical section problems.
- Problems of the Critical Section are also resolved by synchronization of hardware
- Synchronization hardware is not a simple method to implement for everyone, so the strict software method known as Mutex Locks was also introduced.
- Semaphore is another algorithm or solution to the critical section problem.

Prev

Report a Bug

Next

You Might Like

Top Tutorials

(https://www.facebook.com/guru99com/)

(https://twitter.com/guru99com)

(https://www.linkedin.com/company/guru99/)

(https://www.youtube.com/channel/UC19i1XDCh88IzgHlET8atqFQ)

(https://forms.aweber.com/form/46/724807646.htm)

(http://guru99.link/recommends-kobiton-mobile)

Kobiton (http://guru99.link/recommends-kobiton-mobile)

(/software-testing.html)

Testing (/software-testing.html)

(/ethical-hacking-tutorials.html)

Hacking (/ethical-hacking-tutorials.html)

About

About Us (/about-us.html)
Advertise with Us (/advertise-us.html)
Write For Us (/become-an-instructor.html)
Contact Us (/contact-us.html)

Career Suggestion

SAP Career Suggestion Tool (/best-sap-module.html)
Software Testing as a Career (/software-testing-career-complete-guide.html)

Interesting

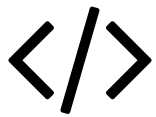
eBook (/ebook-pdf.html)
Blog (/blog/)
Quiz (/tests.html)
SAP eBook (/sap-ebook-pdf.html)

(/sap-training-hub.html)

SAP (/sap-training-hub.html)

(/java-tutorial.html)

Java (/java-tutorial.html)



(/python-tutorials.html)

Python (/python-tutorials.html)

Execute online

Execute Java Online (/try-java-editor.html)
Execute Javascript (/execute-javascript-online.html)
Execute HTML (/execute-html-online.html)
Execute Python (/execute-python-online.html)

(/selenium-tutorial.html)

Selenium (/selenium-tutorial.html)

(/informatica-tutorials.html)

Informatica (/informatica-tutorials.html)

(/jira-tutorial-a-complete-guide-for-beginners.html)

JIRA (/jira-tutorial-a-complete-guide-for-beginners.html)