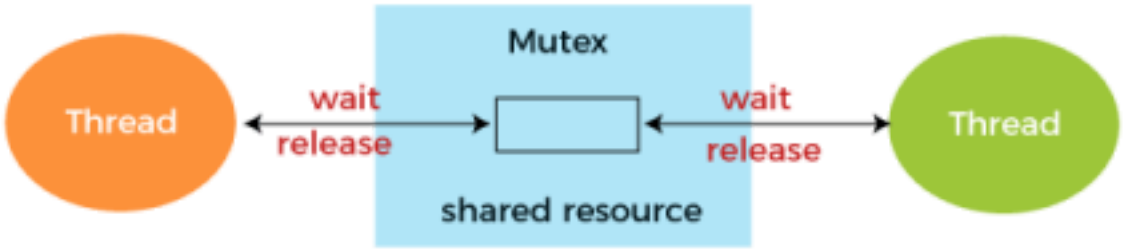
Operating Systems (CS F372)
Tutorial Sheet 7
Mutex

In this tutorial sheet, we will explore the concept of Binary Locks using Mutex.



In multitasking programming, mutex locks, or mutual exclusion locks, are synchronization functions that prevent simultaneous possession of resources shared by numerous threads or procedures. The word "mutex" means "mutual exclusion."

A mutex lock makes it possible to implement mutual exclusion by limiting the number of threads or processes that can simultaneously acquire the lock. A single thread or process has to first try to obtain the mutex lock for something that is shared before it can access it. The thread or process can use the resource that has been shared after acquiring the lock. When finished, it releases the lock so that different threads or processes can take possession of it. It is basically a program object that prevents multiple threads from accessing the same shared resource simultaneously.

Questions

1. This problem is to understand the importance of mutex and simultaneous access to a shared variable. Create a global array `arr` and a variable `ind` Initialized to 0. In the main process, create 5 threads. In each thread, increment the value of `ind`, sleep the thread for a small duration, and set the value at index `ind-1` to `ind`. Finally, print the contents of the array. Now implement mutex locks in the critical section of the code and observe the difference. Explain why this happens**.**

2. This program is to understand how to allow a Mutex to be shared between processes. Write a program that creates a shared memory containing a mutex lock and an index variable, then creates five child processes that each acquire the lock, read and print the index number, and increment it.

3. What is the difference between Mutex, Semaphore, and Spin-lock?

4. Write a C program to solve the reader-writer problem, with 5 writers and 5 readers. Create 2 global variables `count` and `numberOfReader`. The reader process should read the current value of the `count` variable and print it along with the reader number (Reader 1, 2, etc.). The writer process should double the value of the `count` variable and print the value of the variable as well as the writer number (Writer 1, 2, etc.). The remaining requirements are the same as the reader-writer problem.

Take Home Exercise: Analyze your solution to Q4 and check whether it satisfies all the conditions for solutions to the Critical Section Problem.