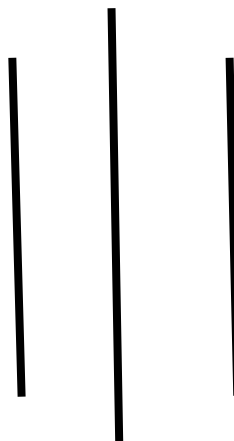


**INSTITUTE OF ENGINEERING**  
**ADVANCED COLLEGE OF ENGINEERING AND MANAGEMENT**  
Kupondole, Lalitpur  
**(AFFILIATED TO TRIBHUVAN UNIVERSITY)**



Lab No:5  
Subject: Distributed System

**Submitted By:**

Name: Sameep Dhakal  
Roll no: ACE074BCT063  
Date: 1/08/2021

**Submitted To:**

Department of Computer  
and  
Electronics Engineering

## LAB 5: Mutual Exclusion

**OBJECTIVE:** To implement mutual exclusion in a process.

### Theory

Mutual exclusion is a concurrency control property which is introduced to prevent race conditions. It is the requirement that a process cannot enter its critical section while another concurrent process is currently present or executing in its critical section i.e. only one process is allowed to execute the critical section at any given instance of time.

Mutual exclusion in single computer system vs. distributed system: In single computer system, memory and other resources are shared between different processes. The status of shared resources and the status of users is easily available in the shared memory so with the help of shared variable (For example: Semaphores) mutual exclusion problem can be easily solved.

In Distributed systems, we neither have shared memory nor a common physical clock and there for we cannot solve mutual exclusion problem using shared variables. To eliminate the mutual exclusion problem in distributed system approach based on message passing is used.

A site in distributed system do not have complete information of state of the system due to lack of shared memory and a common physical clock.

### Code:

```
#include<stdio.h>
#include<conio.h>
#include<time.h>
void main()
{
int cs=0,pro=0;
int run=5;
char key='a';
time_t t1,t2;

printf("Press a key(except q) to enter a process into critical section.");
printf(" \nPress q at any time to exit.");
//t1 = time(NULL) - 5;
while(key!='q')
{
while(!kbhit())
{
if(cs!=0)//1
{
t2 = time(NULL);//123334445+3
if(t2-t1 > run)//
{
printf("Process%d ",pro-1);
printf(" exits critical section.\n");
cs=0;//
```

```

}
}
}
key = getch();
if(key!='q')
{
if(cs!=0)//1

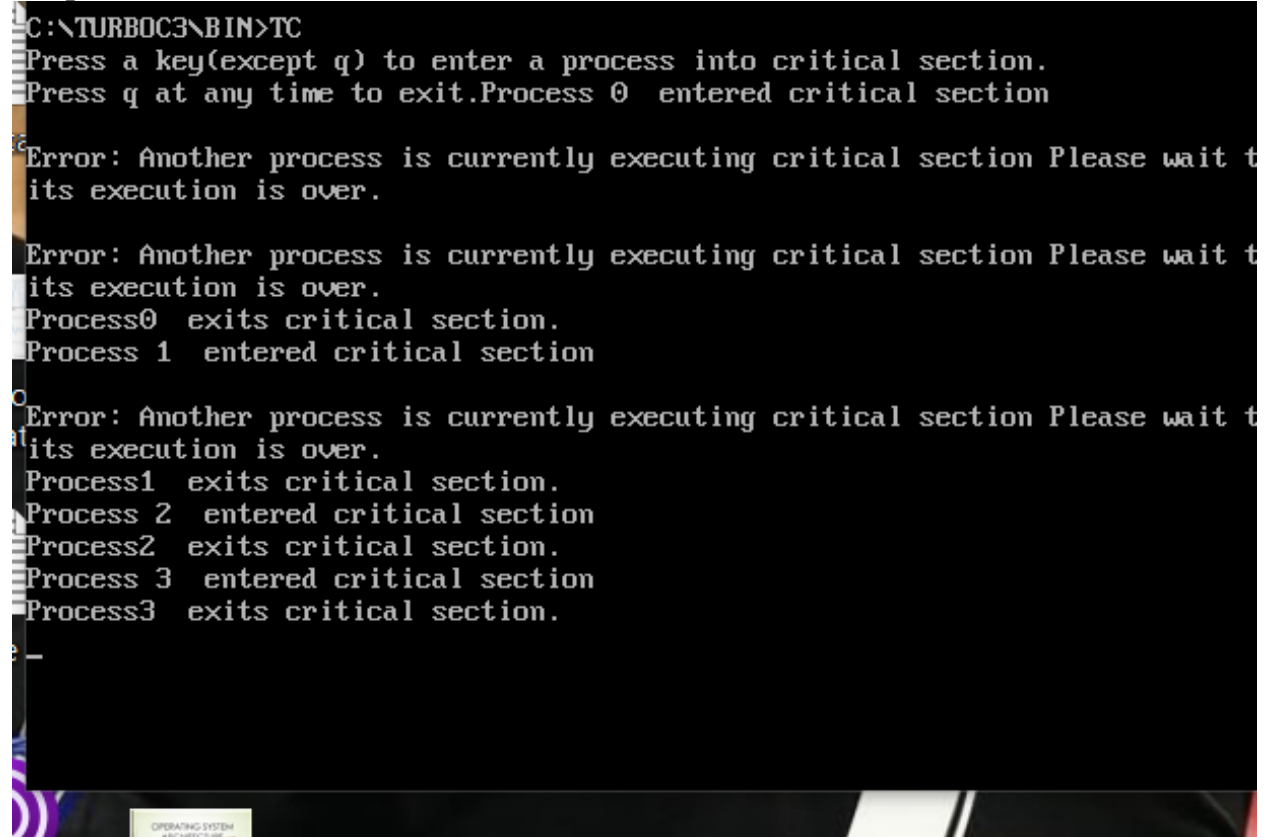
```

```

printf("\nError: Another process is currently executing critical section Please wait till its
execution is over.\n");
else
{
printf("Process %d ",pro);//0
printf(" entered critical section\n");
cs=1;//
pro++;
t1 = time(NULL);//1234455788
}
}
}
}
}

```

### Output:



```

C:\TURBOC3\BIN>TC
Press a key(except q) to enter a process into critical section.
Press q at any time to exit.Process 0 entered critical section
Error: Another process is currently executing critical section Please wait t
its execution is over.
Error: Another process is currently executing critical section Please wait t
its execution is over.
Process0 exits critical section.
Process 1 entered critical section
Error: Another process is currently executing critical section Please wait t
its execution is over.
Process1 exits critical section.
Process 2 entered critical section
Process2 exits critical section.
Process 3 entered critical section
Process3 exits critical section.
-

```

## **Discussion and Conclusion**

In this lab, we implemented mutual exclusion in a process. When a process is currently executing critical section, another process cannot enter it until the first one exits critical section .Error message shows up when we try to fetch another process into critical section.