



# National Textile University

## Department of Computer Science

Subject:

Operating System

---

Submitted to:

Sir Nasir Mehmood

---

Submitted by:

Ghulam Mohyuddin

---

Reg number:

23-NTU-CS-1158

---

Lab number:

6<sup>th</sup>

---

Semester:

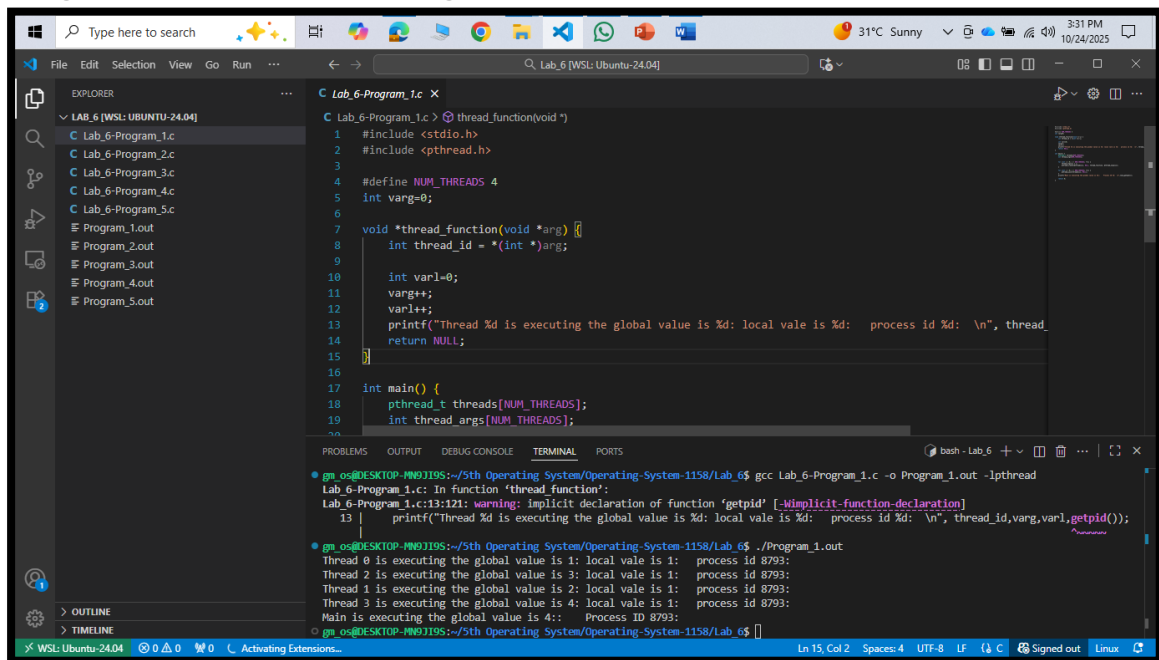
5<sup>th</sup>

---

## The lab objective is to cover Peterson and Mutex concept.

Peterson	Mutex
We use while loop of synchronization and iterations.	We use built in function <code>p_thread_mutex_t</code> object.
We must write each step in code.	We call the built in function instead of writing each line.
Its steps are:	

### Program 1: Multi Threading.

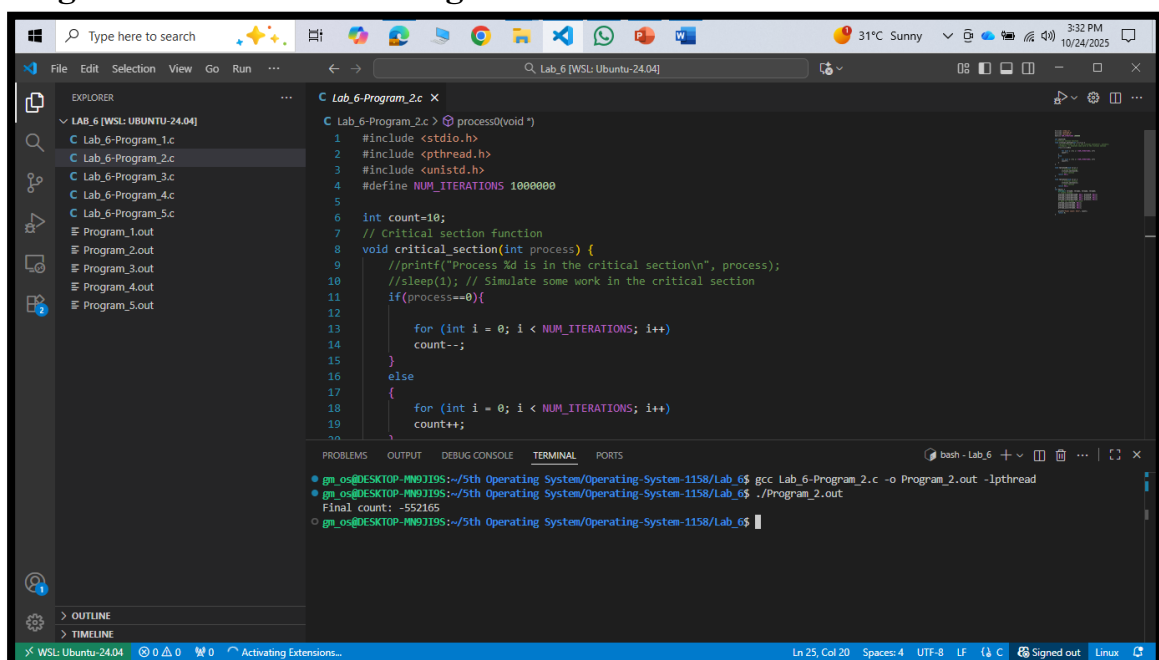


```
1 #include <stdio.h>
2 #include <pthread.h>
3
4 #define NUM_THREADS 4
5 int var=0;
6
7 void *thread_function(void *arg) {
8     int thread_id = *(int *)arg;
9
10    int var1=0;
11    var++;
12    var1++;
13    printf("Thread %d is executing the global value is %d: local vale is %d: process id %d: \n", thread_id, var, var1, getpid());
14    return NULL;
15 }
16
17 int main() {
18     pthread_t threads[NUM_THREADS];
19     int thread_args[NUM_THREADS];
20     ...
```

Lab\_6-Program\_1.c: In function 'thread\_function':  
Lab\_6-Program\_1.c:13:121: warning: implicit declaration of function 'getpid' [-Wimplicit-function-declaration]  
13 | printf("Thread %d is executing the global value is %d: local vale is %d: process id %d: \n", thread\_id, var, var1, getpid());  
|

Thread 0 is executing the global value is 1: local vale is 1: process id 8793:  
Thread 2 is executing the global value is 3: local vale is 1: process id 8793:  
Thread 1 is executing the global value is 2: local vale is 1: process id 8793:  
Thread 3 is executing the global value is 4: local vale is 1: process id 8793:  
Main is executing the global value is 4: Process ID 8793:

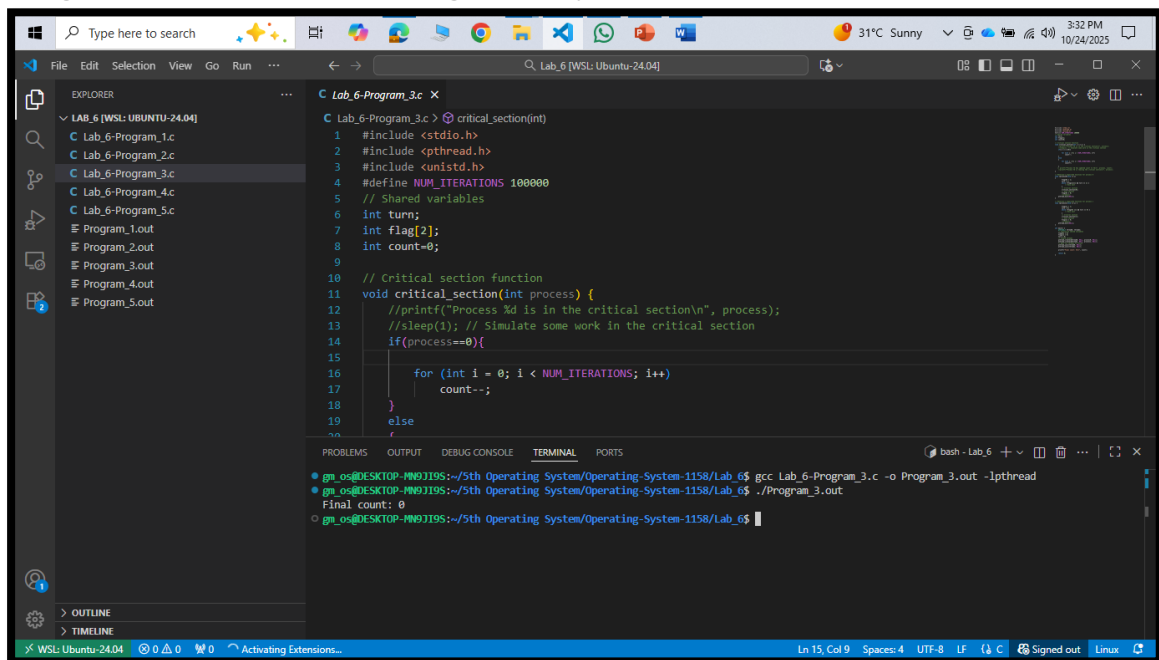
### Program 2: Multi-Processing.



```
1 #include <stdio.h>
2 #include <pthread.h>
3 #include <unistd.h>
4 #define NUM_ITERATIONS 1000000
5
6 int count=10;
7 // Critical section function
8 void critical_section(int process) {
9     //printf("Process %d is in the critical section\n", process);
10    //sleep(1); // Simulate some work in the critical section
11    if(process==0){
12        for (int i = 0; i < NUM_ITERATIONS; i++)
13            count--;
14    }
15    else {
16        for (int i = 0; i < NUM_ITERATIONS; i++)
17            count++;
18    }
19    ...
```

Final count: -552165

## Program 3: Multi-Processing with Synchronization solutions.



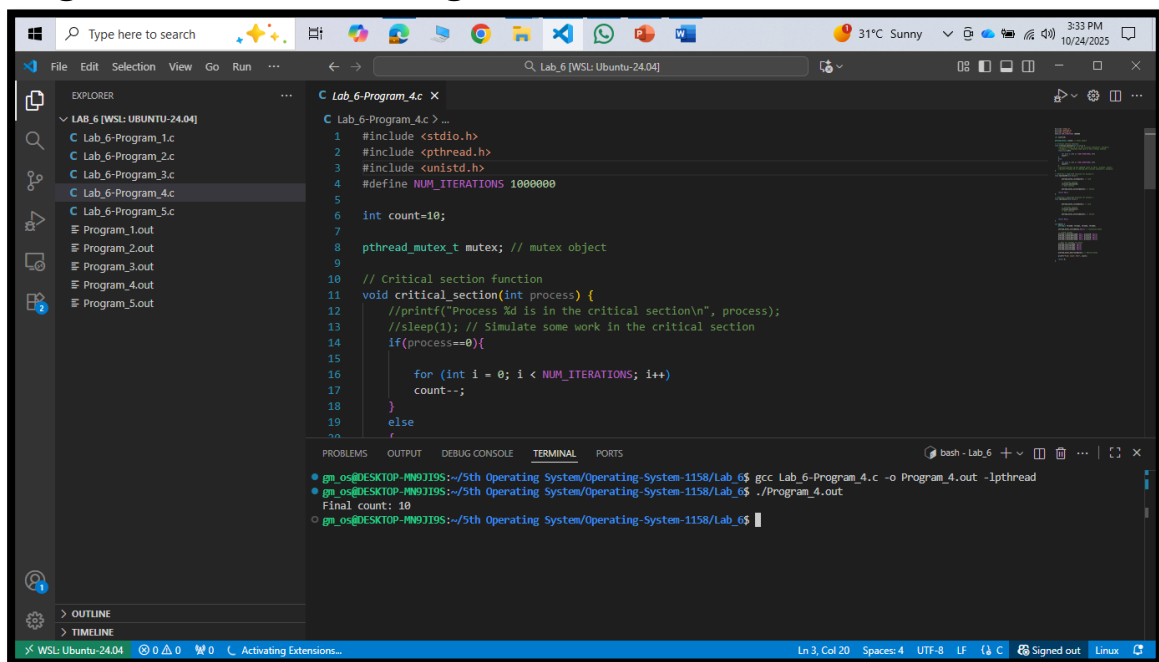
The screenshot shows the Visual Studio Code editor with a file explorer on the left displaying a directory structure for 'LAB\_6 [WSL: UBUNTU-24.04]'. The main editor window is open to 'Lab\_6-Program\_3.c'. The code implements a multi-threaded program using critical sections for synchronization. It includes headers for `<stdio.h>`, `<pthread.h>`, and `<unistd.h>`, and defines `NUM_ITERATIONS` as 100000. A shared variable `count` is initialized to 0, and a `flag` array is declared. The `critical_section` function is defined to take a process ID and contains a loop that increments `count` while holding the critical section. The `main` function creates two threads, each calling `critical_section` with its own process ID. The terminal output shows the compilation and execution of the program, resulting in a final count of 0.

```
1 #include <stdio.h>
2 #include <pthread.h>
3 #include <unistd.h>
4 #define NUM_ITERATIONS 100000
5 // Shared variables
6 int turn;
7 int flag[2];
8 int count=0;
9
10 // Critical section function
11 void critical_section(int process) {
12     //printf("Process %d is in the critical section\n", process);
13     //sleep(1); // Simulate some work in the critical section
14     if(process==0){
15         for (int i = 0; i < NUM_ITERATIONS; i++)
16             count++;
17     }
18     else
19         ;
20 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
gn_os@DESKTOP-HN9J1T9S:~/5th Operating System/Operating-System-1158/Lab_6$ gcc Lab_6-Program_3.c -o Program_3.out -lpthread
gn_os@DESKTOP-HN9J1T9S:~/5th Operating System/Operating-System-1158/Lab_6$ ./Program_3.out
Final count: 0
gn_os@DESKTOP-HN9J1T9S:~/5th Operating System/Operating-System-1158/Lab_6$
```

## Program 4: Multi-Processing with Mutex.



The screenshot shows the Visual Studio Code editor with a file explorer on the left displaying a directory structure for 'LAB\_6 [WSL: UBUNTU-24.04]'. The main editor window is open to 'Lab\_6-Program\_4.c'. The code implements a multi-threaded program using a mutex for synchronization. It includes headers for `<stdio.h>`, `<pthread.h>`, and `<unistd.h>`, and defines `NUM_ITERATIONS` as 100000. A shared variable `count` is initialized to 10, and a `pthread_mutex_t` object is declared. The `critical_section` function is defined to take a process ID and contains a loop that increments `count` while holding the mutex. The `main` function creates two threads, each calling `critical_section` with its own process ID. The terminal output shows the compilation and execution of the program, resulting in a final count of 10.

```
1 #include <stdio.h>
2 #include <pthread.h>
3 #include <unistd.h>
4 #define NUM_ITERATIONS 100000
5
6 int count=10;
7
8 pthread_mutex_t mutex; // mutex object
9
10 // Critical section function
11 void critical_section(int process) {
12     //printf("Process %d is in the critical section\n", process);
13     //sleep(1); // Simulate some work in the critical section
14     if(process==0){
15         for (int i = 0; i < NUM_ITERATIONS; i++)
16             count++;
17     }
18     else
19         ;
20 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
gn_os@DESKTOP-HN9J1T9S:~/5th Operating System/Operating-System-1158/Lab_6$ gcc Lab_6-Program_4.c -o Program_4.out -lpthread
gn_os@DESKTOP-HN9J1T9S:~/5th Operating System/Operating-System-1158/Lab_6$ ./Program_4.out
Final count: 10
gn_os@DESKTOP-HN9J1T9S:~/5th Operating System/Operating-System-1158/Lab_6$
```