A logo of a company

AI-generated content may be incorrect.

**National Textile University**

**Department of Computer Science**

Subject: Operating System

Submitted to: Nasir Mahmood

Submitted by: Eemaan Fatima

Reg number:23-NTU-CS-1146

Lab : 06

Semester:5th

Lab manual 06

**Task 1:**

**#include <stdio.h>**

**#include <pthread.h>**

**#include <unistd.h>**

**#define NUM\_THREADS 4**

**int varg=0;**

**void \*thread\_function(void \*arg) {**

**int thread\_id = \*(int \*)arg;**

**int varl=0;**

**varg++;**

**varl++;**

**printf("Thread %d is executing the global value is %d: local vale is %d:   process id %d:  \n", thread\_id,varg,varl,getpid());**

**return NULL;**

**}**

**int main() {**

**pthread\_t threads[NUM\_THREADS];**

**int thread\_args[NUM\_THREADS];**

**for (int i = 0; i < NUM\_THREADS; ++i) {**

**thread\_args[i] = i;**

**pthread\_create(&threads[i], NULL, thread\_function, &thread\_args[i]);**

**}**

**for (int i = 0; i < NUM\_THREADS; ++i) {**

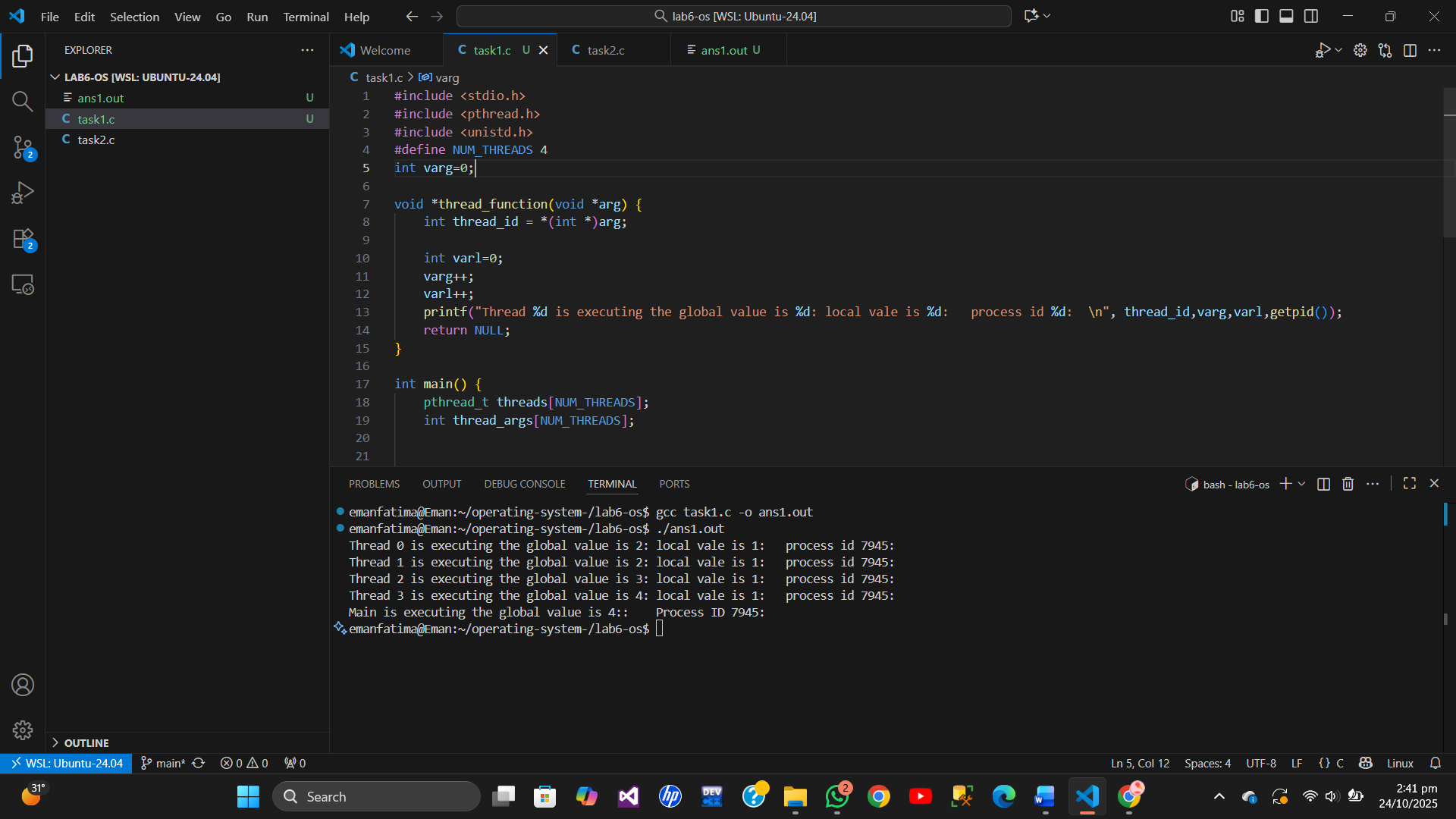
**pthread\_join(threads[i], NULL);**

**}**

**printf("Main is executing the global value is %d::    Process ID %d:  \n",varg,getpid());**

**return 0;**

**}**



**Task 2:**

**Code:**

**#include <stdio.h>**

**#include <pthread.h>**

**#include <unistd.h>**

**#define NUM\_ITERATIONS 1000000**

**int count=10;**

**// Critical section function**

**void critical\_section(int process) {**

**//printf("Process %d is in the critical section\n", process);**

**//sleep(1); // Simulate some work in the critical section**

**if(process==0){**

**for (int i = 0; i < NUM\_ITERATIONS; i++)**

**count--;**

**}**

**else**

**{**

**for (int i = 0; i < NUM\_ITERATIONS; i++)**

**count++;**

**}**

**}**

**void \*process0(void \*arg) {**

**// Critical section**

**critical\_section(0);**

**// Exit section**

**return NULL;**

**}**

**void \*process1(void \*arg) {**

**// Critical section**

**critical\_section(1);**

**// Exit section**

**return NULL;**

**}**

**int main() {**

**pthread\_t thread0, thread1, thread2, thread3;**

**// Create threads**

**pthread\_create(&thread0, NULL, process0, NULL);**

**pthread\_create(&thread1, NULL, process1, NULL);**

**pthread\_create(&thread2, NULL, process0, NULL);**

**pthread\_create(&thread3, NULL, process1, NULL);**

**// Wait for threads to finish**

**pthread\_join(thread0, NULL);**

**pthread\_join(thread1, NULL);**

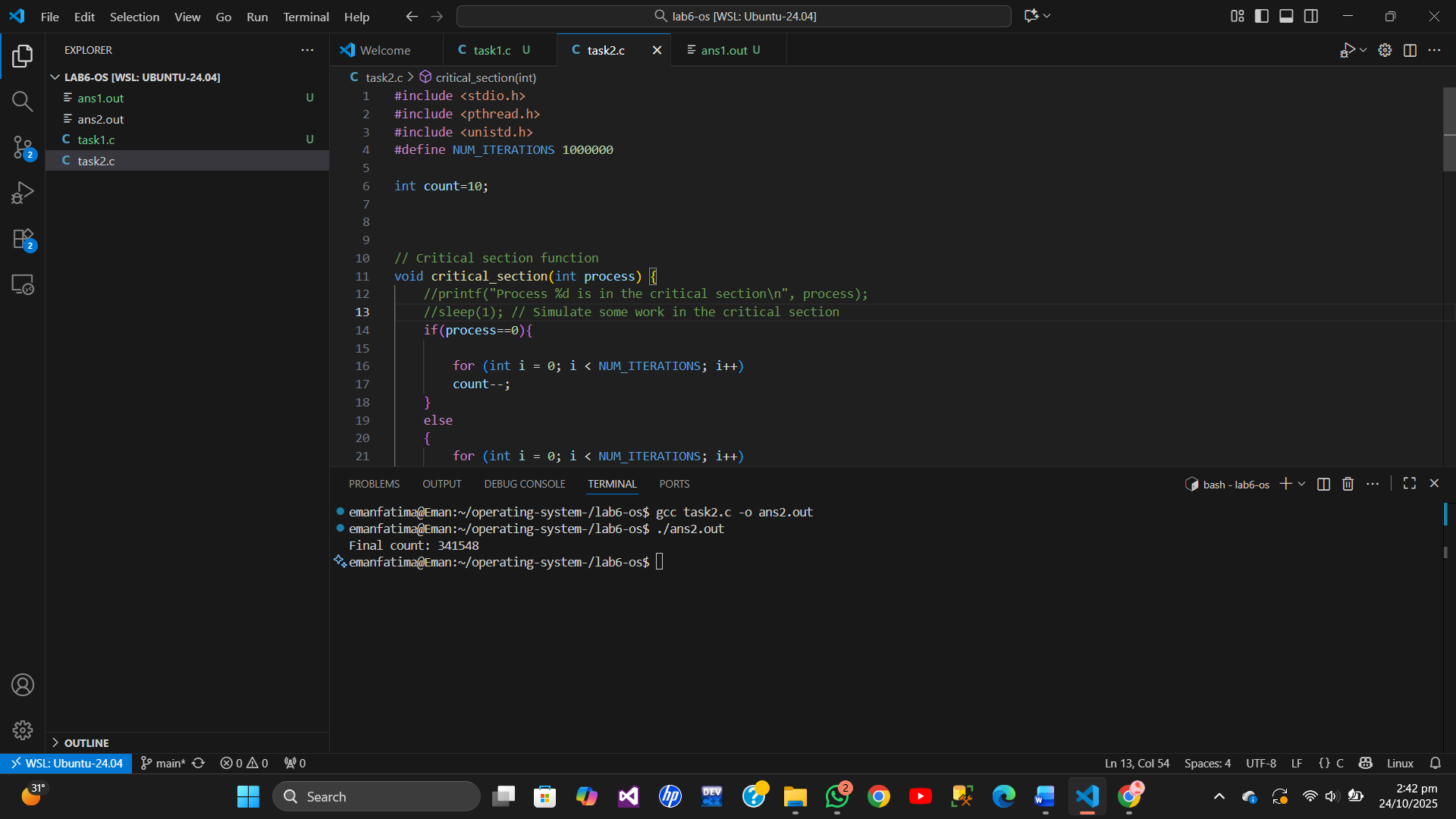
**pthread\_join(thread2, NULL);**

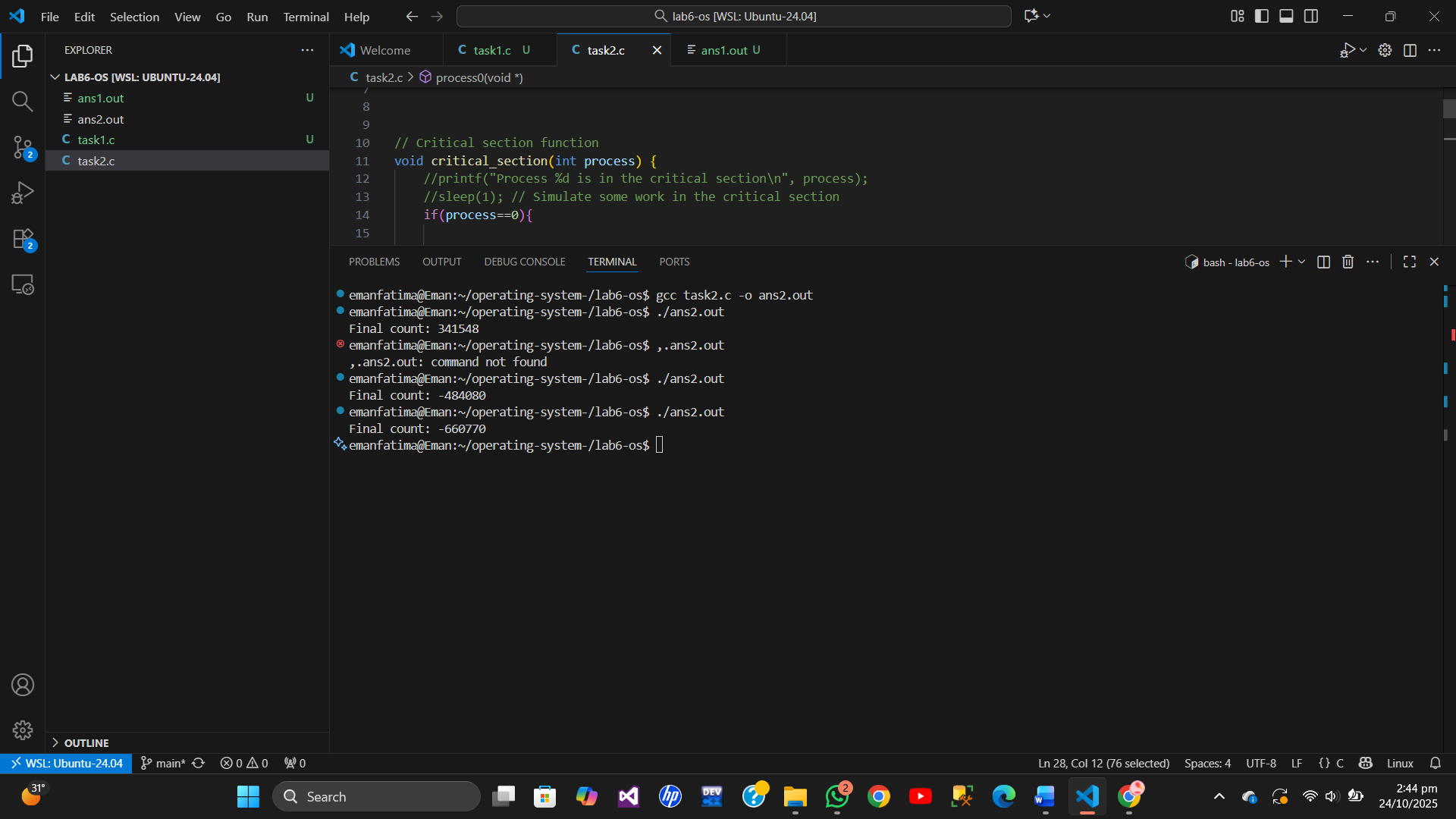
**pthread\_join(thread3, NULL);**

**printf("Final count: %d\n", count);**

**return 0;**

**}**

****

****

**Task 3:**

**#include <stdio.h>**

**#include <pthread.h>**

**#include <unistd.h>**

**#define NUM\_ITERATIONS 100000**

**// Shared variables**

**int turn;**

**int flag[2];**

**int count=0;**

**// Critical section function**

**void critical\_section(int process) {**

**//printf("Process %d is in the critical section\n", process);**

**//sleep(1); // Simulate some work in the critical section**

**if(process==0){**

**for (int i = 0; i < NUM\_ITERATIONS; i++)**

**count--;**

**}**

**else**

**{**

**for (int i = 0; i < NUM\_ITERATIONS; i++)**

**count++;**

**}**

**// printf("Process %d has updated count to %d\n", process, count);**

**//printf("Process %d is leaving the critical section\n", process);**

**}**

**// Peterson's Algorithm function for process 0**

**void \*process0(void \*arg) {**

**flag[0] = 1;**

**turn = 1;**

**while (flag[1]==1 && turn == 1) {**

**// Busy wait**

**}**

**// Critical section**

**critical\_section(0);**

**// Exit section**

**flag[0] = 0;**

**//sleep(1);**

**pthread\_exit(NULL);**

**}**

**// Peterson's Algorithm function for process 1**

**void \*process1(void \*arg) {**

**flag[1] = 1;**

**turn = 0;**

**while (flag[0] ==1 && turn == 0) {**

**// Busy wait**

**}**

**// Critical section**

**critical\_section(1);**

**// Exit section**

**flag[1] = 0;**

**//sleep(1);**

**pthread\_exit(NULL);**

**}**

**int main() {**

**pthread\_t thread0, thread1;**

**// Initialize shared variables**

**flag[0] = 0;**

**flag[1] = 0;**

**turn = 0;**

**// Create threads**

**pthread\_create(&thread0, NULL, process0, NULL);**

**pthread\_create(&thread1, NULL, process1, NULL);**

**// Wait for threads to finish**

**pthread\_join(thread0, NULL);**

**pthread\_join(thread1, NULL);**

**printf("Final count: %d\n", count);**

**return 0;**

**}**

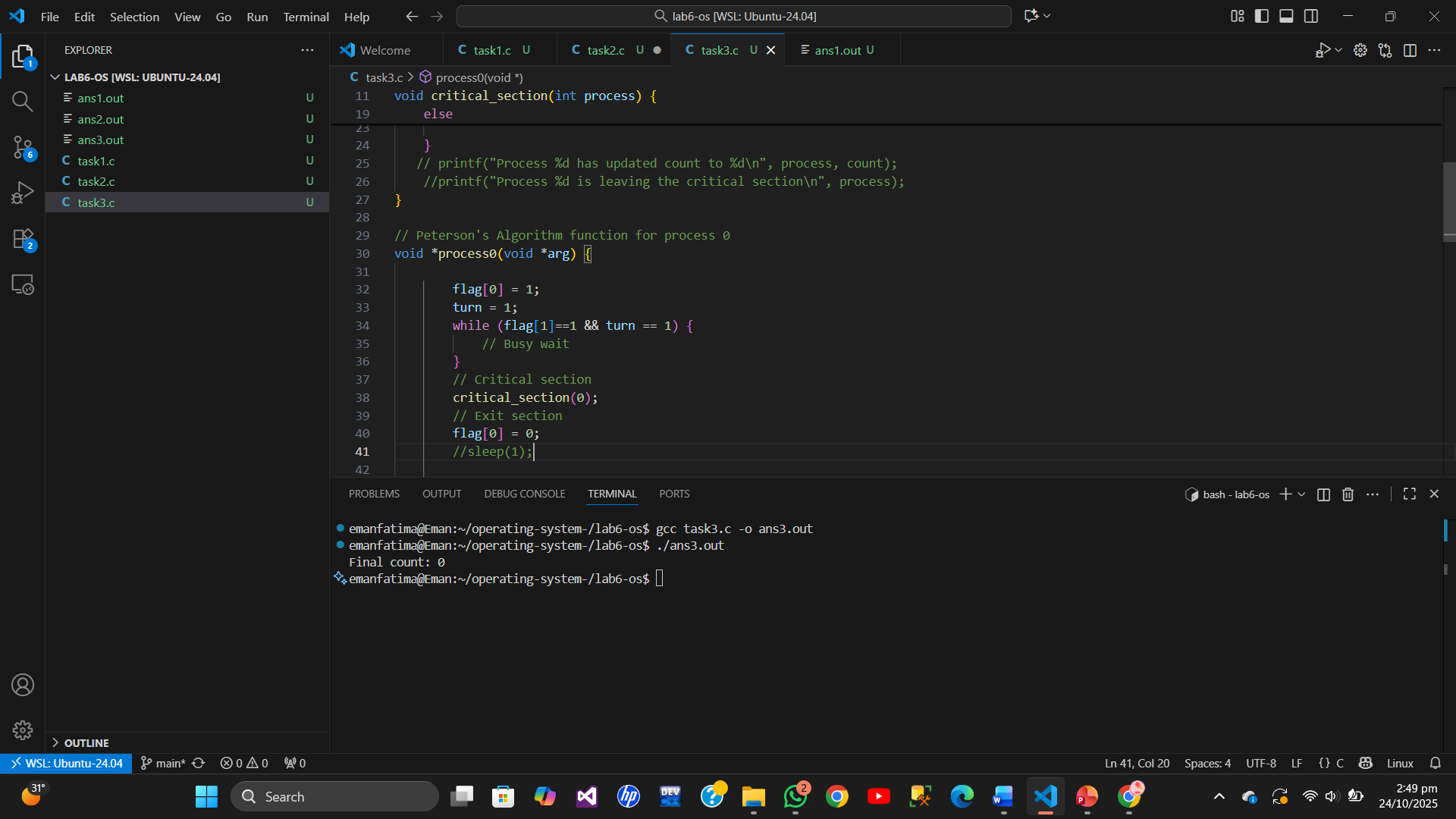
**Remarks:**

**In this program we use peterson algorithm .**

**For the synchronization of proccess that the proccess not clash with each other.**

**Flags and while loop which is entry point for the condition are used .**

**When one proccess flag is true it will run the turn condition of other procces then while of 1st proccess run and it will execute .**

****

**Task 4:**

**#include <stdio.h>**

**#include <pthread.h>**

**#include <unistd.h>**

**#define NUM\_ITERATIONS 1000000**

**int count=10;**

**pthread\_mutex\_t mutex; // mutex object**

**// Critical section function**

**void critical\_section(int process) {**

**//printf("Process %d is in the critical section\n", process);**

**//sleep(1); // Simulate some work in the critical section**

**if(process==0){**

**for (int i = 0; i < NUM\_ITERATIONS; i++)**

**count--;**

**}**

**else**

**{**

**for (int i = 0; i < NUM\_ITERATIONS; i++)**

**count++;**

**}**

**//printf("Process %d has updated count to %d\n", process, count);**

**//printf("Process %d is leaving the critical section\n", process);**

**}**

**// Peterson's Algorithm function for process 0**

**void \*process0(void \*arg) {**

**pthread\_mutex\_lock(&mutex); // lock**

**// mutex lock find kr ky next line ma chla jy ga .nhi mily ga tu mangta rhy ga lock**

**//lock p1 or p0 ma sy kisi ik ko mily ga .**

**// jis ko lock mily ga wo run hujay ga proocess or dosra wait ma phas jay ga**

**// jab kam hujay ga process ka tu wo unlock hujay ga or dosra process ko lock mily ga or wo run hujay ga**

**// Critical section**

**critical\_section(0);**

**// Exit section**

**pthread\_mutex\_unlock(&mutex); // unlock**

**return NULL;**

**}**

**// Peterson's Algorithm function for process 1**

**void \*process1(void \*arg) {**

**pthread\_mutex\_lock(&mutex); // lock**

**// Critical section**

**critical\_section(1);**

**// Exit section**

**pthread\_mutex\_unlock(&mutex); // unlock**

**return NULL;**

**}**

**int main() {**

**pthread\_t thread0, thread1, thread2, thread3;**

**pthread\_mutex\_init(&mutex,NULL); // initialize mutex**

**// Create threads**

**pthread\_create(&thread0, NULL, process0, NULL);**

**pthread\_create(&thread1, NULL, process1, NULL);**

**pthread\_create(&thread2, NULL, process0, NULL);**

**pthread\_create(&thread3, NULL, process1, NULL);**

**// Wait for threads to finish**

**pthread\_join(thread0, NULL);**

**pthread\_join(thread1, NULL);**

**pthread\_join(thread2, NULL);**

**pthread\_join(thread3, NULL);**

**pthread\_mutex\_destroy(&mutex); // destroy mutex**

**printf("Final count: %d\n", count);**

**return 0;**

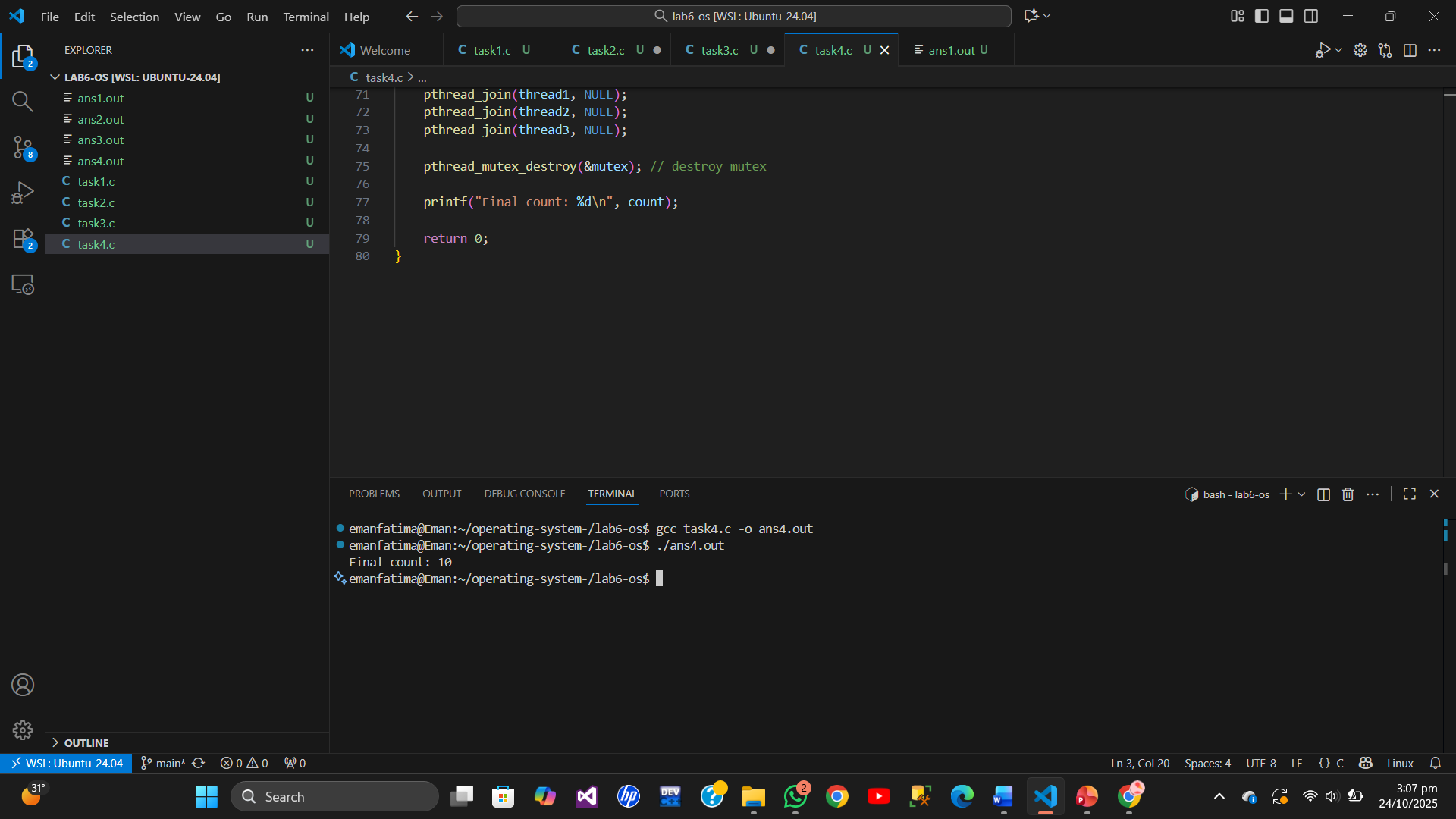
**}**

**Remarks :**

**This code basically used an algorithm (Mutelock )**

**This involves a lock condition whoever finds locks first it will run and the other proccess will wait .**

**And when the proccess finish its work it will unlock .and give lock to other.**

****

**Task 5:**

**#include <stdio.h>**

**#include <pthread.h>**

**#include <unistd.h>**

**#define NUM\_ITERATIONS 1000000**

**int count=10;**

**pthread\_mutex\_t mutex; // mutex object**

**// Critical section function**

**void critical\_section(int process) {**

**if (process == 0) {**

**for (int i = 0; i < NUM\_ITERATIONS; i++)**

**count--; // Process 0 decreases count**

**}**

**else if (process == 1) {**

**for (int i = 0; i < NUM\_ITERATIONS; i++)**

**count++; // Process 1 increases count**

**}**

**else if (process == 2) {**

**for (int i = 0; i < NUM\_ITERATIONS; i++)**

**count += 2; // Process 2 adds 2**

**}**

**}**

**// Peterson's Algorithm function for process 0**

**void \*process0(void \*arg) {**

**pthread\_mutex\_lock(&mutex); // lock**

**// mutex lock find kr ky next line ma chla jy ga .nhi mily ga tu mangta rhy ga lock**

**//lock p1 or p0 ma sy kisi ik ko mily ga .**

**// jis ko lock mily ga wo run hujay ga proocess or dosra wait ma phas jay ga**

**// jab kam hujay ga process ka tu wo unlock hujay ga or dosra process ko lock mily ga or wo run hujay ga**

**// Critical section**

**critical\_section(0);**

**// Exit section**

**pthread\_mutex\_unlock(&mutex); // unlock**

**return NULL;**

**}**

**// Peterson's Algorithm function for process 1**

**void \*process1(void \*arg) {**

**pthread\_mutex\_lock(&mutex); // lock**

**// Critical section**

**critical\_section(1);**

**// Exit section**

**pthread\_mutex\_unlock(&mutex); // unlock**

**return NULL;**

**}**

**void \*process2(void \*arg) {**

**pthread\_mutex\_lock(&mutex); // lock**

**// Critical section**

**critical\_section(2);**

**// Exit section**

**pthread\_mutex\_unlock(&mutex); // unlock**

**return NULL;**

**}**

**int main() {**

**pthread\_t thread0, thread1, thread2, thread3, thread4 , thread5;**

**pthread\_mutex\_init(&mutex,NULL); // initialize mutex**

**// Create threads**

**pthread\_create(&thread0, NULL, process0, NULL);**

**pthread\_create(&thread1, NULL, process1, NULL);**

**pthread\_create(&thread2, NULL, process0, NULL);**

**pthread\_create(&thread3, NULL, process1, NULL);**

**pthread\_create(&thread4, NULL, process2, NULL);**

**pthread\_create(&thread5, NULL, process2, NULL);**

**// Wait for threads to finish**

**pthread\_join(thread0, NULL);**

**pthread\_join(thread1, NULL);**

**pthread\_join(thread2, NULL);**

**pthread\_join(thread3, NULL);**

**pthread\_join(thread4, NULL);**

**pthread\_join(thread5, NULL);**

**pthread\_mutex\_destroy(&mutex); // destroy mutex**

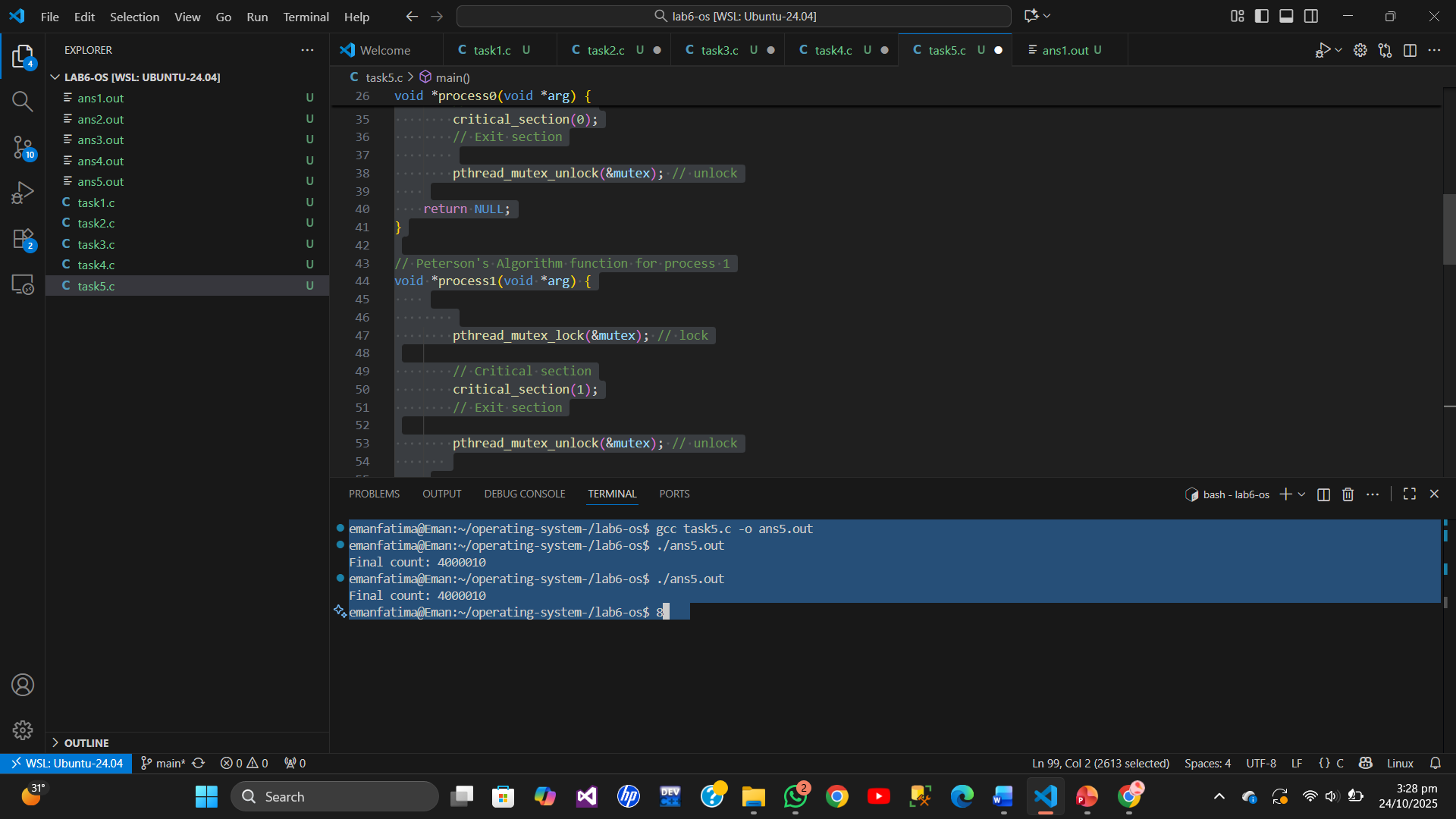
**printf("Final count: %d\n", count);**

**return 0;**

**}**

**Remarks :**

**In this task i simply add a new 3rd condition and 3rd proccess and make thread object for it .this is a program using mutelock algorithm .**

****

**Task 6:**

**#include <stdio.h>**

**#include <pthread.h>**

**#include <unistd.h>**

**#define NUM\_ITERATIONS 1000000**

**int count=10;**

**pthread\_mutex\_t mutex; // mutex object**

**// Critical section function**

**void critical\_section(int process) {**

**if (process == 0) {**

**for (int i = 0; i < NUM\_ITERATIONS; i++)**

**count--; // Process 0 decreases count**

**}**

**else if (process == 1) {**

**for (int i = 0; i < NUM\_ITERATIONS; i++)**

**count++; // Process 1 increases count**

**}**

**else if (process == 2) {**

**for (int i = 0; i < NUM\_ITERATIONS; i++)**

**count += 3; // Process 2 adds 2**

**}**

**}**

**// Peterson's Algorithm function for process 0**

**void \*process0(void \*arg) {**

**pthread\_mutex\_lock(&mutex); // lock**

**// mutex lock find kr ky next line ma chla jy ga .nhi mily ga tu mangta rhy ga lock**

**//lock p1 or p0 ma sy kisi ik ko mily ga .**

**// jis ko lock mily ga wo run hujay ga proocess or dosra wait ma phas jay ga**

**// jab kam hujay ga process ka tu wo unlock hujay ga or dosra process ko lock mily ga or wo run hujay ga**

**// Critical section**

**critical\_section(0);**

**// Exit section**

**pthread\_mutex\_unlock(&mutex); // unlock**

**return NULL;**

**}**

**// Peterson's Algorithm function for process 1**

**void \*process1(void \*arg) {**

**pthread\_mutex\_lock(&mutex); // lock**

**// Critical section**

**critical\_section(1);**

**// Exit section**

**pthread\_mutex\_unlock(&mutex); // unlock**

**return NULL;**

**}**

**void \*process2(void \*arg) {**

**//pthread\_mutex\_lock(&mutex); // lock**

**// Critical section**

**critical\_section(2);**

**// Exit section**

**//pthread\_mutex\_unlock(&mutex); // unlock**

**return NULL;**

**}**

**int main() {**

**pthread\_t thread0, thread1, thread2, thread3, thread4 , thread5;**

**pthread\_mutex\_init(&mutex,NULL); // initialize mutex**

**// Create threads**

**pthread\_create(&thread0, NULL, process0, NULL);**

**pthread\_create(&thread1, NULL, process1, NULL);**

**pthread\_create(&thread2, NULL, process0, NULL);**

**pthread\_create(&thread3, NULL, process1, NULL);**

**pthread\_create(&thread4, NULL, process2, NULL);**

**pthread\_create(&thread5, NULL, process2, NULL);**

**// Wait for threads to finish**

**pthread\_join(thread0, NULL);**

**pthread\_join(thread1, NULL);**

**pthread\_join(thread2, NULL);**

**pthread\_join(thread3, NULL);**

**pthread\_join(thread4, NULL);**

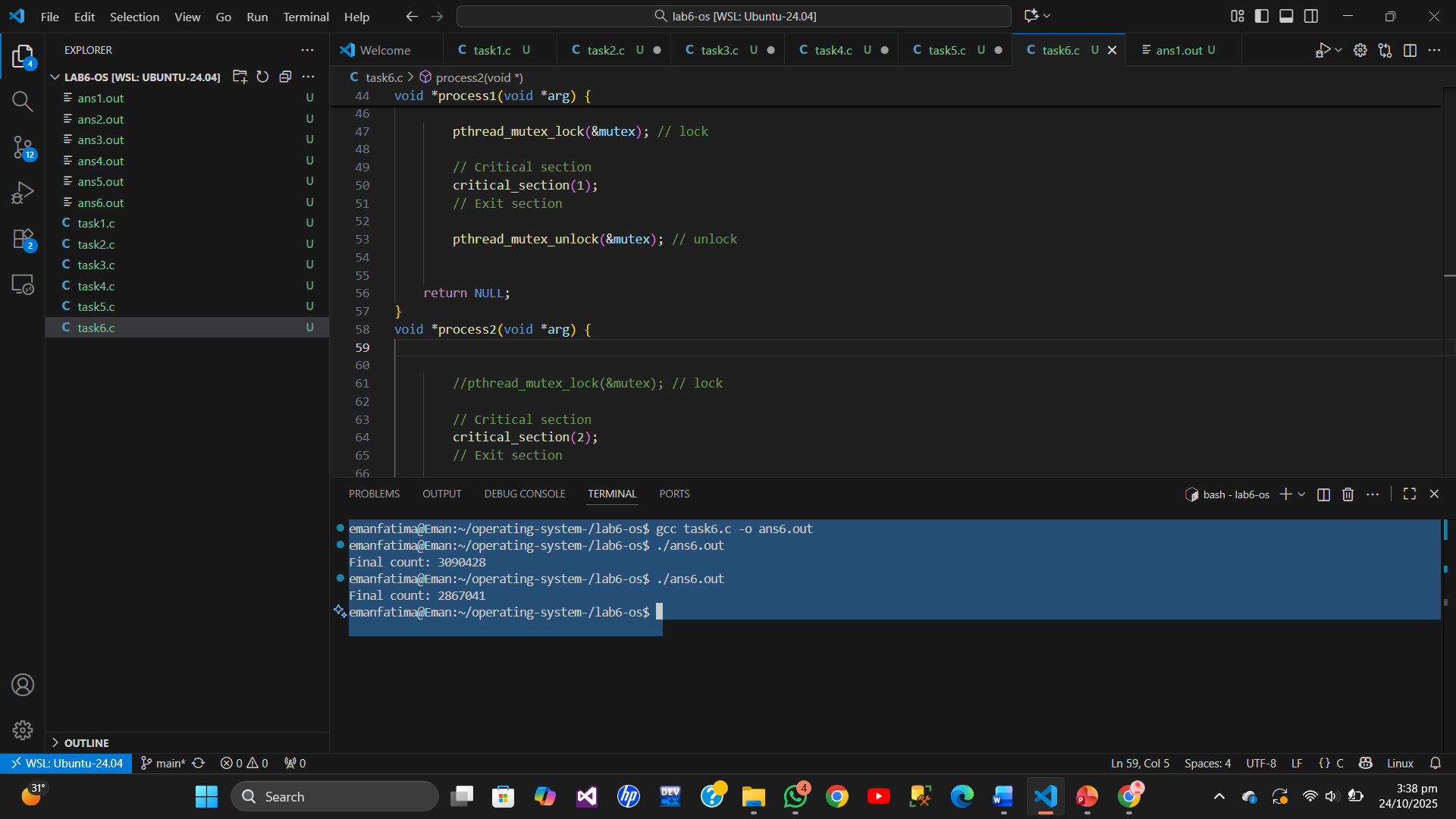
**pthread\_join(thread5, NULL);**

**pthread\_mutex\_destroy(&mutex); // destroy mutex**

**printf("Final count: %d\n", count);**

**return 0;**

**}**

****

**Remarks :**

**We just comment lock and unlock then it will not synchronizen the proccess and give different answer every time in final count .**

**Differnce :**

| **Aspect** | **Peterson’s Algorithm** | **Mutex Lock** |
| --- | --- | --- |
| **Implementation** | **Software-based using flag[2] and turn** | **Using pthread\_mutex\_t** |
| **How It Works** | **Each process sets flag, turn, and waits in loop for the other to complete.** | **Thread locks with pthread\_mutex\_lock() and unlocks after the work finish .** |
| **Number of Threads** | **Only 2 threads can be used** | **Multiple threads can be used** |
| **Complexity** | **More complex** | **Simple (lock/unlock )** |
| **Limitations** | **Works only for 2 processes busy waiting** | **Works for many threads, as it is efficient** |