

Task1:

Code:

```
#include <stdio.h>
#include <pthread.h>
#include <unistd.h> //  getpid() ke liye include karna zaroori hai

#define NUM_THREADS 4
int varg = 0;

void *thread_function(void *arg) {
    int thread_id = *(int *)arg;

    int varl = 0;
    varg++; // global variable increment
    varl++; // local variable increment
    printf("Thread %d is executing. Global value = %d | Local value = %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. Final Global value = %d | Process ID = %d\n", varg, getpid());
```

```
return 0;  
}
```

The screenshot shows the Visual Studio Code interface running in WSL. The Explorer sidebar shows files in the 'LAB6 [WSL: UBUNTU-24.04]' folder, including 'a.out', 'task1', 'Task1-out', 'Task2', 'Task2.c', 'Task3', 'Task3.c', 'Task4', 'Task4.c', and 'Task1.c'. The 'Task1.c' file is open in the editor, displaying C code for a multi-threaded program. The terminal tab shows the following session:

```
fizza1157@DESKTOP-P8HNL3S:~/Lab6$ ./Task3  
bash: ./Task3: No such file or directory  
fizza1157@DESKTOP-P8HNL3S:~/Lab6$ gcc Task3.c -o Task3-out  
fizza1157@DESKTOP-P8HNL3S:~/Lab6$ ./Task3-out  
Final count: 10  
fizza1157@DESKTOP-P8HNL3S:~/Lab6$ gcc Task4.c -o Task4-out  
fizza1157@DESKTOP-P8HNL3S:~/Lab6$ ./Task4-out  
Final count: 4000010  
fizza1157@DESKTOP-P8HNL3S:~/Lab6$ ^C  
fizza1157@DESKTOP-P8HNL3S:~/Lab6$ gcc Task1.c -o Task1-out  
fizza1157@DESKTOP-P8HNL3S:~/Lab6$ ./Task1-out  
Thread 0 is executing. Global value = 1 | Local value = 1 | Process ID = 9564  
Thread 1 is executing. Global value = 2 | Local value = 1 | Process ID = 9564  
Thread 3 is executing. Global value = 4 | Local value = 1 | Process ID = 9564  
Thread 2 is executing. Global value = 3 | Local value = 1 | Process ID = 9564  
Main is executing. Final Global value = 4 | Process ID = 9564  
fizza1157@DESKTOP-P8HNL3S:~/Lab6$
```

Task2:

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;
}

```

The screenshot shows a Windows desktop environment with the Visual Studio Code (VS Code) application open. The title bar indicates the workspace is "Lab6 [WSL: Ubuntu-24.04]". The left sidebar shows a file tree for "LAB6 [WSL: UBUNTU-24.04]" containing files like a.out, task1, Task1-out, Task1.c, Task2, Task2-out, Task2.c, Task3, Task3-out, Task3.c, Task4, Task4-out, and Task4.c. The main editor area displays the content of Task2.c:

```

void *process0(void *arg) {
    // Critical section
    critical_section();
    // Exit section

    return NULL;
}

```

The bottom terminal window shows the following command-line session:

```

./Task3-out
Final count: 10
./Task4-out
Final count: 4000010
./Task1-out
Main is executing. Final Global value = 4 | Process ID = 9564
./Task2-out
Final count: -374511

```

The system tray at the bottom left shows the date and time as "10/24/2025 3:46 PM".

Task3:

Code:

```
#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
```

// 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;
```

```
}
```

Task2:

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) {
```

```
[REDACTED]
```

```
    // Critical section
```

```
    critical_section(0);
```

```
    // Exit section
```

```
[REDACTED]
```

```
    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);
```

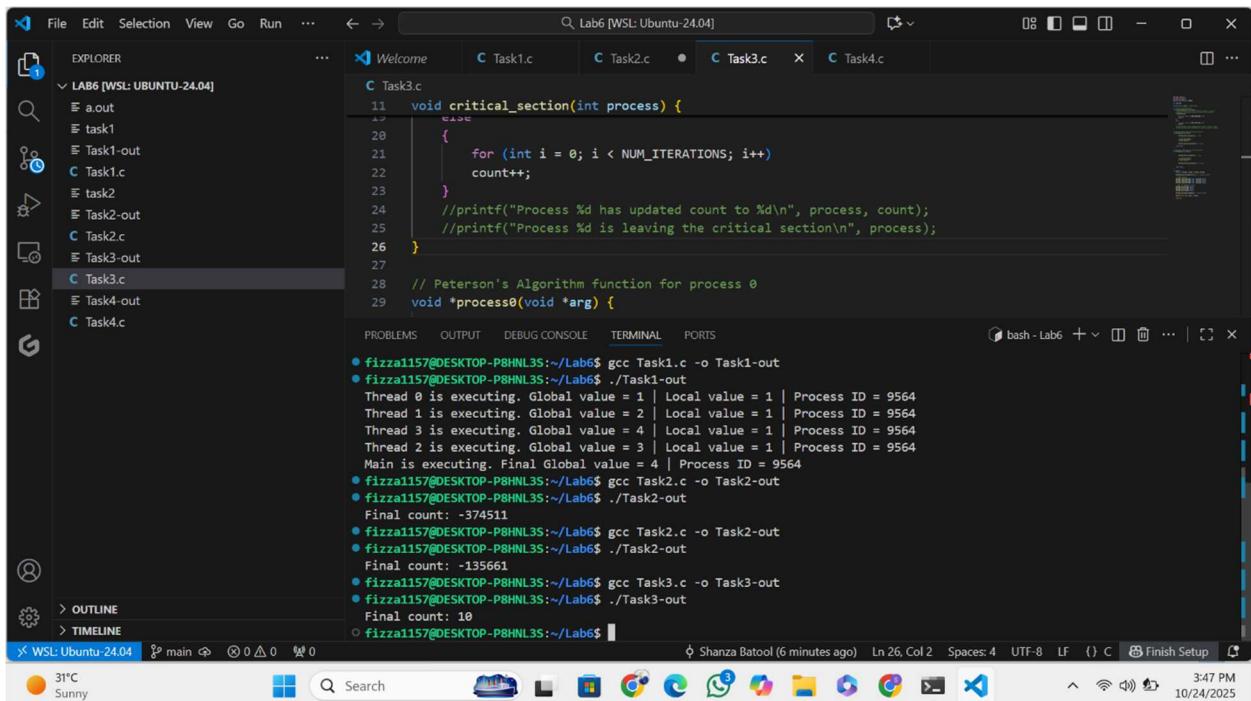
```
    // Wait for threads to finish  
    pthread_join(thread0, NULL);
```

```
pthread_join(thread1, NULL);  
pthread_join(thread2, NULL);  
pthread_join(thread3, NULL);
```

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

```
return 0;
```

}



Task4:

Code:

```
#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--;

    }

    else if (process == 1) {

        for (int i = 0; i < NUM_ITERATIONS; i++)

            count++;

    }

    else if (process == 2) {

        for (int i = 0; i < NUM_ITERATIONS; i++)

            count += 2; // third process modifies differently

    }

}

// Process 0

void *process0(void *arg) {

    pthread_mutex_lock(&mutex); // lock

    critical_section(0);

    pthread_mutex_unlock(&mutex); // unlock

    return NULL;

}

// Process 1

void *process1(void *arg) {
```

```
pthread_mutex_lock(&mutex);

critical_section(1);

pthread_mutex_unlock(&mutex);

return NULL;

}
```

// Process 2 (newly added)

```
void *process2(void *arg) {

pthread_mutex_lock(&mutex);

critical_section(2);

pthread_mutex_unlock(&mutex);

return NULL;

}
```

```
int main() {
```

```
pthread_t thread0, thread1, thread2, thread3, thread4, thread5;
```

```
pthread_mutex_init(&mutex, NULL); // initialize mutex
```

```
// Create threads for all processes
```

```
pthread_create(&thread0, NULL, process0, NULL);

pthread_create(&thread1, NULL, process1, NULL);

pthread_create(&thread2, NULL, process2, NULL);

pthread_create(&thread3, NULL, process0, NULL);

pthread_create(&thread4, NULL, process1, NULL);

pthread_create(&thread5, NULL, process2, NULL);
```

```
// Wait for all threads to complete
```

```
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;  
}
```

The screenshot shows a Windows desktop environment with the Visual Studio Code (VS Code) application open. The title bar indicates the workspace is "Lab6 [WSL: UBUNTU-24.04]".

The Explorer sidebar on the left lists files in the "LAB6 [WSL: UBUNTU-24.04]" folder, including "task1", "Task1.c", "Task2", "Task2.c", "Task3", "Task3.c", "Task4", and "Task4.c".

The code editor on the right displays the content of "Task4.c". The code defines two functions: "process0" and "process1".

```
void *process0(void *arg) {  
    ...  
}  
  
// Process 1  
void *process1(void *arg) {  
    pthread_mutex_lock(&mutex);  
    critical_section(1);  
    pthread_mutex_unlock(&mutex);  
    return NULL;  
}  
  
// Process 2 (newly added)
```

The terminal tab at the bottom shows the following command-line session:

```
fizza1157@DESKTOP-P8HNL3S:~/Lab6$ ./Task1-out  
fizza1157@DESKTOP-P8HNL3S:~/Lab6$ gcc Task2.c -o Task2-out  
fizza1157@DESKTOP-P8HNL3S:~/Lab6$ ./Task2-out  
Final count: -374511  
fizza1157@DESKTOP-P8HNL3S:~/Lab6$ gcc Task2.c -o Task2-out  
fizza1157@DESKTOP-P8HNL3S:~/Lab6$ ./Task2-out  
Final count: -135661  
fizza1157@DESKTOP-P8HNL3S:~/Lab6$ gcc Task3.c -o Task3-out  
fizza1157@DESKTOP-P8HNL3S:~/Lab6$ ./Task3-out  
Final count: 10  
fizza1157@DESKTOP-P8HNL3S:~/Lab6$ gcc t  
/usr/bin/ld: cannot find t: No such file or directory  
collect2: error: ld returned 1 exit status  
fizza1157@DESKTOP-P8HNL3S:~/Lab6$ gcc Task4.c -o Task4-out  
fizza1157@DESKTOP-P8HNL3S:~/Lab6$ ./Task4-out  
Final count: 4000010  
fizza1157@DESKTOP-P8HNL3S:~/Lab6$
```

The status bar at the bottom shows the user's name "Shanza Batool", the time "3:48 PM", and the date "10/24/2025".