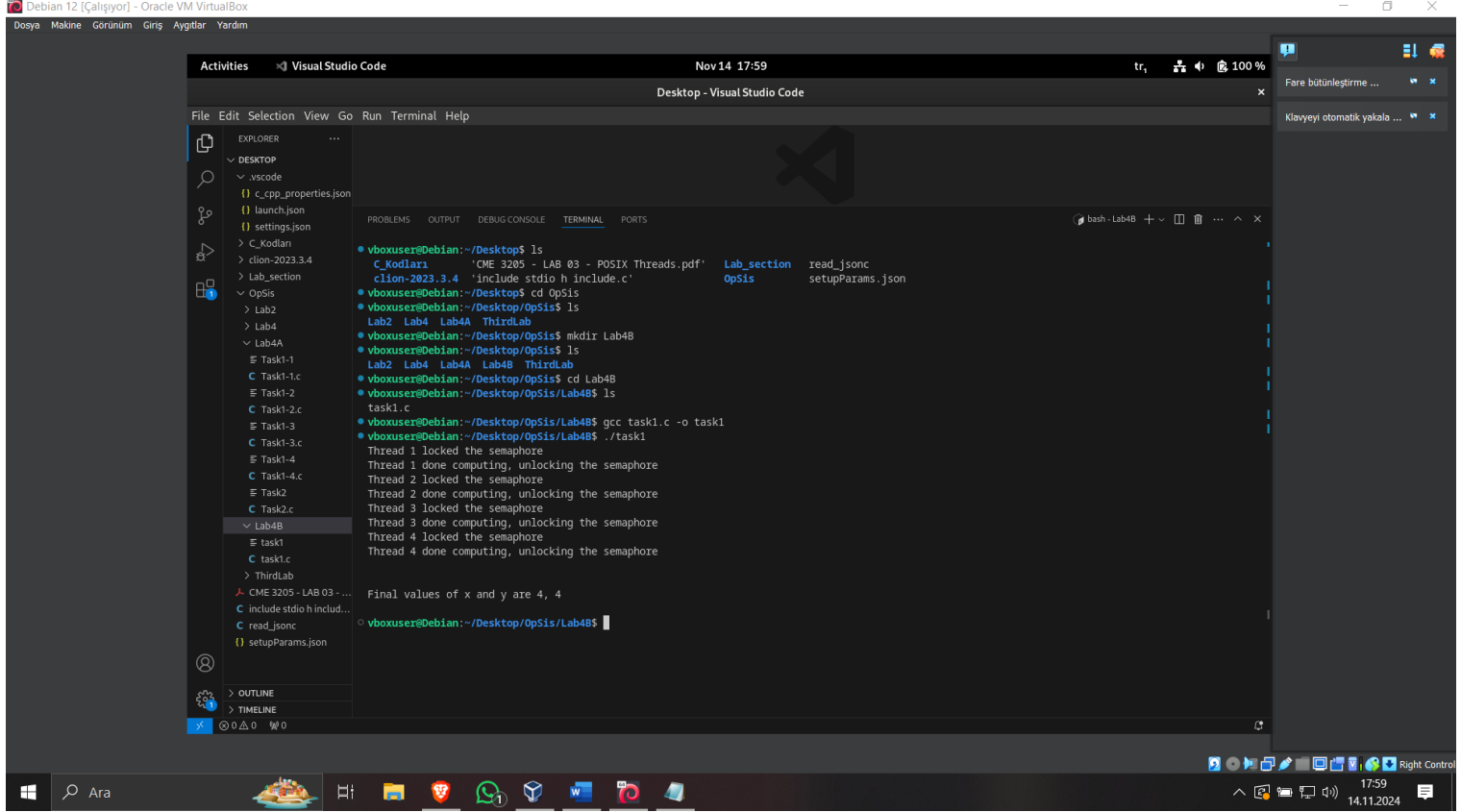


2021510010 – Çağrı AYDIN – Lab 04 Part B

TASK 1:

ScreenShot:



```
Debian 12 [Çalışıyor] - Oracle VM VirtualBox
Dosya Makine Görünüm Giriş Aygıtlar Yardım

Activities Visual Studio Code Nov 14 17:59 tr 100 %
Desktop - Visual Studio Code

File Edit Selection View Go Run Terminal Help

EXPLORER
DESKTOP
.vscode
c_cpp_properties.json
launch.json
settings.json
C_Kodlari
clion-2023.3.4
Lab_section
OpSis
Lab2
Lab4
Lab4A
Task1-1
Task1-1.c
Task1-2
Task1-2.c
Task1-3
Task1-3.c
Task1-4
Task1-4.c
Task2
Task2.c
Lab4B
task1
task1.c
ThirdLab
CME 3205 - LAB 03 - ...
include stdio h includ...
read_jsonc
setupParams.json

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
bash - Lab4B

vboxuser@Debian:~/Desktop$ ls
C_Kodlari 'CME 3205 - LAB 03 - POSIX Threads.pdf' Lab_section read_jsonc
clion-2023.3.4 'include stdio h include.c' OpSis setupParams.json

vboxuser@Debian:~/Desktop$ cd OpSis

vboxuser@Debian:~/Desktop/OpSis$ ls
Lab2 Lab4 Lab4A ThirdLab

vboxuser@Debian:~/Desktop/OpSis$ mkdir Lab4B

vboxuser@Debian:~/Desktop/OpSis$ ls
Lab2 Lab4 Lab4A Lab4B ThirdLab

vboxuser@Debian:~/Desktop/OpSis$ cd Lab4B

vboxuser@Debian:~/Desktop/OpSis/Lab4B$ ls
task1.c

vboxuser@Debian:~/Desktop/OpSis/Lab4B$ gcc task1.c -o task1

vboxuser@Debian:~/Desktop/OpSis/Lab4B$ ./task1
Thread 1 locked the semaphore
Thread 1 done computing, unlocking the semaphore
Thread 2 locked the semaphore
Thread 2 done computing, unlocking the semaphore
Thread 3 locked the semaphore
Thread 3 done computing, unlocking the semaphore
Thread 4 locked the semaphore
Thread 4 done computing, unlocking the semaphore

Final values of x and y are 4, 4

vboxuser@Debian:~/Desktop/OpSis/Lab4B$
```

New Code Below:

```

#include <pthread.h>
#include <stdlib.h>
#include <unistd.h>
#include <stdio.h>
#include <string.h>
#include <semaphore.h>
#include <stdint.h> // For intptr_t

#define THREAD_COUNT 4
pthread_t tid[THREAD_COUNT];
int x = 0, y = 0;
sem_t semaphore;

void* doSomething(void *arg) {
    int id = (int)(intptr_t)arg;
    sem_wait(&semaphore);
    printf("Thread %d locked the semaphore\n", id);
    x = y + 1;
    unsigned long i = 0;
    for (i = 0; i < 100000000; i++);
    y = x;
    printf("Thread %d done computing, unlocking the semaphore\n", id);
    sem_post(&semaphore);
    pthread_exit(NULL);
}

int main(void) {
    int i = 0, j = 0;
    int err;

    // Initialize the binary semaphore with 1 (allowing one thread at a time)
    if (sem_init(&semaphore, 0, 1) != 0) {
        printf("Semaphore init failed!\n");
        return 1;
    }

    // Create threads
    while (i < THREAD_COUNT) {
        err = pthread_create(&tid[i], NULL, &doSomething, (void *) (intptr_t)(i + 1));
        if (err != 0)
            printf("Can't create thread :[%s]!\n", strerror(err));
        i++;
    }

    // Wait for threads to finish
    while (j < THREAD_COUNT) {
        pthread_join(tid[j], NULL);
        j++;
    }

    printf("\n\nFinal values of x and y are %d, %d\n\n", x, y);

    // Destroy the semaphore
    sem_destroy(&semaphore);

    return 0;
}

```

Task 2:

Thread Count is 2 and Semaphore is 1:

The screenshot shows the Visual Studio Code interface with the file explorer on the left, the code editor in the center, and the terminal at the bottom. The code editor displays the source code for task2.c, which defines a semaphore and two threads. The terminal shows the output of the program, including the semaphore's value and the threads' execution status.

```
task2.c:49:10: error: expected ';' before '}' token
49 |     return 0;
    |     ^
50 | }
    |

vboxuser@Debian:~/Desktop/OpSis/Lab4B$ gcc task2.c -o task2
task2.c: In function 'doSomething':
task2.c:11:10: warning: cast from pointer to integer of different size [-Wpointer-to-int-cast]
11 |     int id = (int)arg;
    |              ^
task2.c: In function 'main':
task2.c:34:11: warning: cast to pointer from integer of different size [-Wint-to-pointer-cast]
34 |     (void *) (i + 1));
    |           ^

vboxuser@Debian:~/Desktop/OpSis/Lab4B$ ./task2
Thread 1: Waiting to enter critical region...
Thread 1: Now in critical region...
Thread 1: Semaphore's value is: 0
Thread 2: Waiting to enter critical region...
Thread 1: Exiting critical region...
Thread 2: Now in critical region...
Thread 2: Semaphore's value is: 0
Thread 2: Exiting critical region...
Main thread: Final value of the semaphore is 1
```

Thread Count is 6 and Semaphore is 3:

The screenshot shows the Visual Studio Code interface with the file explorer on the left, the code editor in the center, and the terminal at the bottom. The code editor displays the source code for task2.c, which defines three semaphores and six threads. The terminal shows the output of the program, including the semaphores' values and the threads' execution status.

```
task2.c:15:10: error: expected ';' before '}' token
15 |     return 0;
    |     ^
16 | }
    |

vboxuser@Debian:~/Desktop/OpSis/Lab4B$ ./task2
Thread 1: Waiting to enter critical region...
Thread 1: Now in critical region...
Thread 3: Waiting to enter critical region...
Thread 3: Now in critical region...
Thread 3: Semaphore's value is: 1
Thread 1: Semaphore's value is: 2
Thread 2: Waiting to enter critical region...
Thread 2: Now in critical region...
Thread 2: Semaphore's value is: 0
Thread 5: Waiting to enter critical region...
Thread 4: Waiting to enter critical region...
Thread 6: Waiting to enter critical region...
Thread 3: Exiting critical region...
Thread 5: Now in critical region...
Thread 5: Semaphore's value is: 0
Thread 2: Exiting critical region...
Thread 4: Now in critical region...
Thread 4: Semaphore's value is: 0
Thread 1: Exiting critical region...
Thread 6: Now in critical region...
Thread 6: Semaphore's value is: 0
Thread 5: Exiting critical region...
Thread 6: Exiting critical region...
Thread 4: Exiting critical region...
Main thread: Final value of the semaphore is 3
```

When we change the thread count to 6 and the semaphore value to 3, the first three threads can enter the critical region at the same time because the semaphore allows up to three threads. The other threads will have to wait until one of the first three threads finishes and releases the semaphore.