YAKUP TALHA YOLCU

CSE 312 OS HW1 REPORT

1801042609

DESIGN DECISIONS

I designed my multhithreading library as tasks. I assumed that tasks are threads.

```
int create_thread(Task* task) {
    return taskManager.AddTask(task);
}
```

I take task and add it to the taskmanagers array

I create tasks like that:

```
Task prod(&gdt,producer);
Task con(&gdt,consumer);


Task task1(&gdt, taskA);
Task task2(&gdt, taskB);
Task task3(&gdt, taskC);
Task task4(&gdt, taskD);
Task task5(&gdt, taskE);
```

taskA, taskB … are entrypoints of tasks.

```
71      TaskManager();
72      ~TaskManager();
73      bool TerminateTask(Task* task);
74      bool AddTask(Task* task);
75      bool YieldTask(Task* task);
76      bool YieldTask();
77      bool JoinTask();
78      bool JoinTask(Task* task);
79      bool ExitTask();
80      int getnumTask();
81
```

I decided to make additions to the current source code

This is taskmanager class and I have overloaded functions because I call one of them in my thread library and other in the inside of first one

```
bool TaskManager::TerminateTask(Task* task) {

    task->taskState=FINISHED_T;

    printf("TERMINATED\n");
    int_bottom();

}
```

In my terminate, I set the tasks state as finished and call the interrupt function

```
bool TaskManager::YieldTask(Task* task) {
    yielded=currentTask;
```

In my yield function I hold the yielded index.

```
bool TaskManager::JoinTask(Task* task) {
    printf("\nTASK JOINED\n");
    joined=currentTask;
    //int_bottom();
}
```

In my jiin function I hold the current joined tasks index

```
    for(int i=0;i<numTasks;i++) {
        if(tasks[i]->taskState==FINISHED_T) {
            int index_s=i;
            for(index_s=i;index_s<(numTasks-1);index_s++) {
                tasks[index_s]=tasks[index_s+1];
            }
            numTasks--;
            currentTask--;
        }
    }
    if(yielded!=-1) {
        char*buf;
        currentTask=yielded-1;
        yielded=-1;

    }

    if(joined!=-1) {
        currentTask=joined-1;
        joined=-1;
    }
```

In my Schedule function I determine finished ones and remove from array. and I also detect the yield and joined ones

```
#define BLOCKED_T 1
#define READY_T 0
#define FINISHED_T 2
```

I determine the states like that

```
void taskA()
{
    for(int i=0;i<50;i++) {
        printf("A");
        sleep();
    }
    taskManager.ExitTask();
}
```

I designed the sample task like that.
Sleep function just does busy waiting

```
void sleep() {
    int i=0;
    while(i<500000) {
        i++;
    }
}
```

I used my own sleep function because there is no sleep function, we cant use any library
I used my own sleep function because when I want to print something I was not able to see what I printed.

I can also print integers successfully, I have functions for it

`int calculate_digit(int& num)` This function calculates digit number of a number

`void toChar(char* buf,int num)`

This function writes number into the buffer as string

I have producer - consumer model
I have insert item,remove item, consume item, produce item functions
Insert item just increments the start number by one
Produce item returns the start number
consume item decrements the start number by one
remove item returns the start number

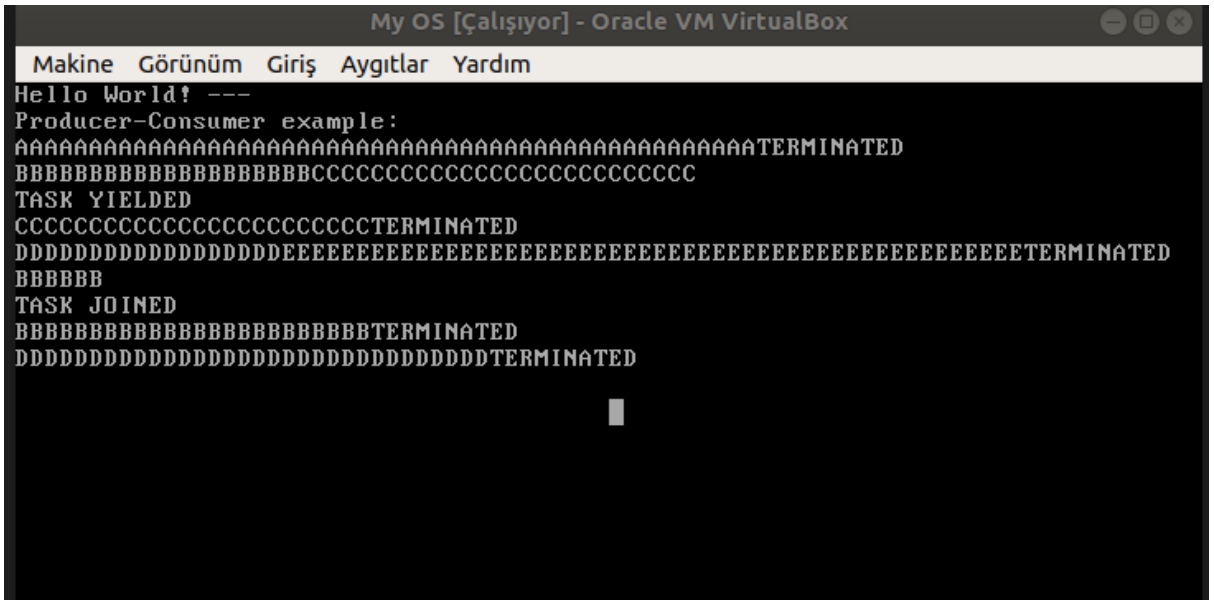
In my model, I have infinite while loop in producer() and consumer()
I call enter region function which is peterson s Solution

```
int enter_region(int process) {
    int other;
    other=(1-process);
    interested[process]=TRUE;
    turn=process;
    int flag=0;

    while(turn==process && interested[other==true]) {
        //printf("WAITING\n");
    }
    flag=1;
    return flag;

}
void leave_region(int process) {
    interested[process]=FALSE;
```

and at the end of critical region which is producing an item or consuming an item, I call the leave region function



As you can see, my yield is not working properly because when I tried to yield the cpu, I was not able to run the task from where it left off. It never go into that task again…

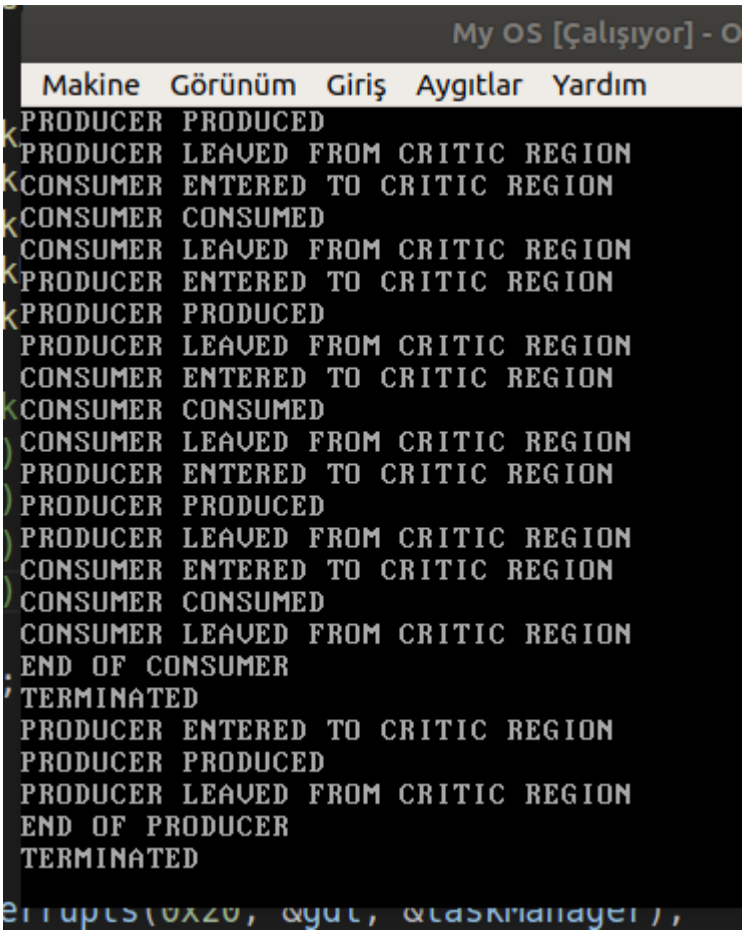
In my join function, I just continue the functions execution.

In terminate I just terminate it and remove it from the task array

I can also create the threads successfully as we can see

My PRODUCER CONSUMER MODEL

To able to take this output, I had to pause the VM. As you can see my producer consumer code is working properly. But if I add the other tasks, it is hard to see their work.



At the end as you can see, consumer and producer is terminated properly, I terminated at the 500th step to make it end.

```
while(turn==process && interested[other==true]) {
    //printf("WAITING\n");
}
```

This while loop-inside of the enter region- does not allow program to have race condition.