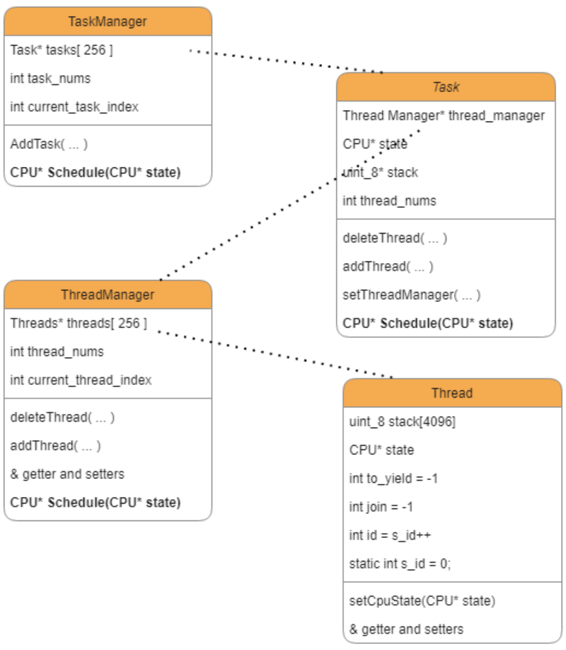
Task – Task Manager – Thread - Thread Manager Structure



Task Manager has tasks to Schedule and chosing the next task to run.

Task has the thread manager to get the next thread from the thread manager schedule.

Thread Manager has the threads to Schedule them and chosing the next thread to execute.

Thread has its own registers and stack area to store data and start of the assiging process location

Task and Thread Scheduler Management

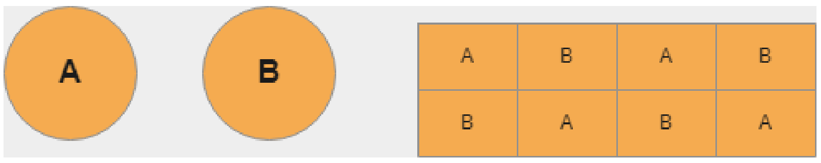
İn this homework our task is implementing our own multi-thread library with our design. With our scheduler.

When we install the required operating system has its own multi-tasking property. And it uses the round robbing scheduler algorithm. What I want to do creating the threads in the task process. After that task works as process and then provide the multi-thread with in a process.

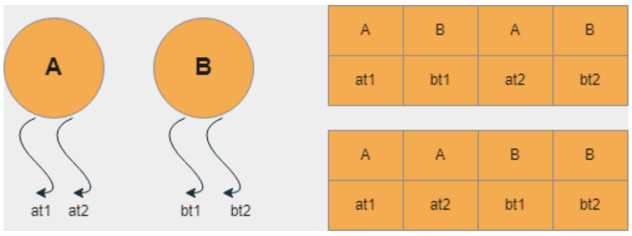
After this thread creation operation, we need to schedule these threads using round robin algorithm.

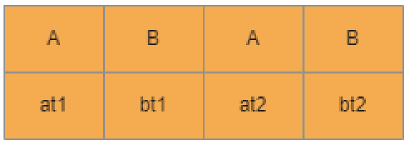
As an example:

Let says we have two process called A and B.
According to the task manager scheduler algorithm process work with the figure.



When we add this process 2 thread each one of them called at1, at2, bt1, bt2. We add another scheduler for schedule algorithm for manage the threads. We have more than one option. According to the round robin algorithm I pick the first table to manage the threads





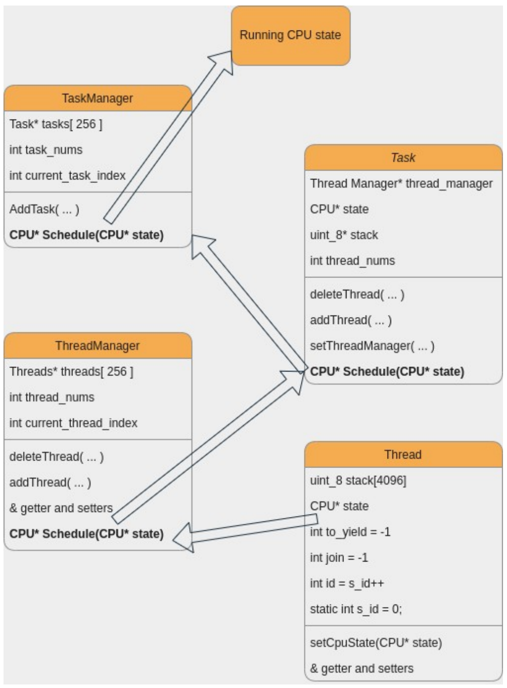
A	B	A	B
at1	bt1	at2	bt2

Chosen table

What we need to do is firstly managing the task using task manager then manage the thread in the task with thread manager with in order.

Each thread has its own CPU state and stack area but the executing thing in the operating system is tasks CPU state and stack are. So what we need to do is moving the thread CPU state and stack area to Task’s CPU state and stack area.

So in the task class I keep these variable as pointer for easily pointing the threads special areas to not copy the whole structure each time.

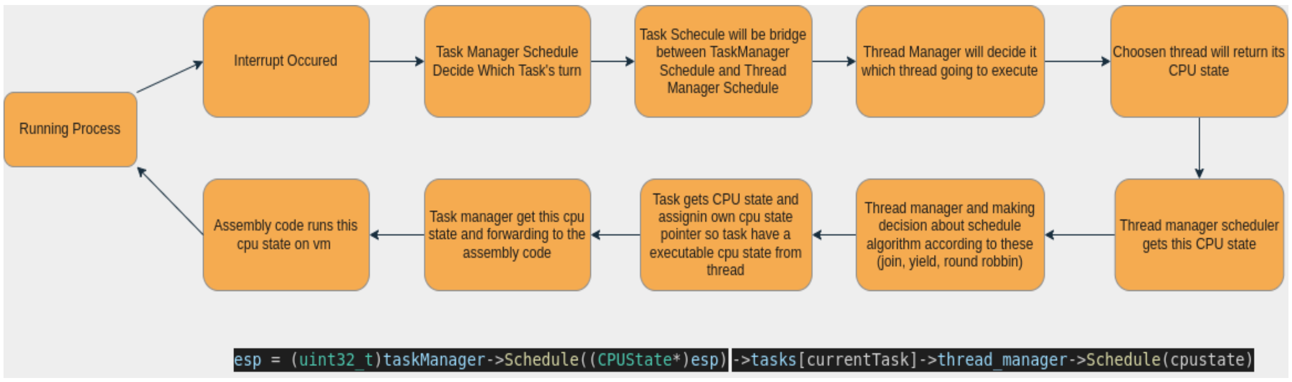


According to manage thread like this we design our Task, Task Manager, Thread and Thread showing in the figure.

First what we did is creating one or more threads and giving them a function for the execution register. then creating a thread manager to manage the created these threads. After that to execute these thread we are creating a Task object and adding the thread manager to Task. Then to manage these created tasks we are creating a task manager object and adding these tasks to task manager.

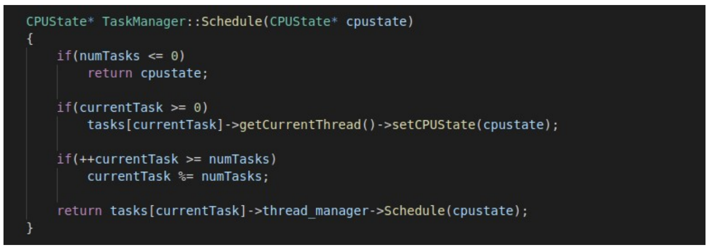
And creating an interrupt handler for the task manager to switch between threads and tasks.

When an interrupt occurred process switching algorithm shown in the following diagram.


Running Process
Interrupt Occured
Task Manager Schedule Decide Which Task's turn
Task Schecule will be bridge between TaskManager Schedule and Thread Manager Schedule
Thread Manager will decide it which thread going to execute
Choosen thread will return its CPU state
Thread manager scheduler gets this CPU state
Thread manager and making decision about schedule algorithm according to these (join, yield, round robbin)
Task gets CPU state and assignin own cpu state pointer so task have a executable cpu state from thread
Task manager get this cpu state and forwarding to the assembly code
Assembly code runs this cpu state on vm

```
esp = (uint32_t)taskManager->Schedule((CPUState*)esp) ->tasks[currentTask]->thread_manager->Schedule(cpustate)
```

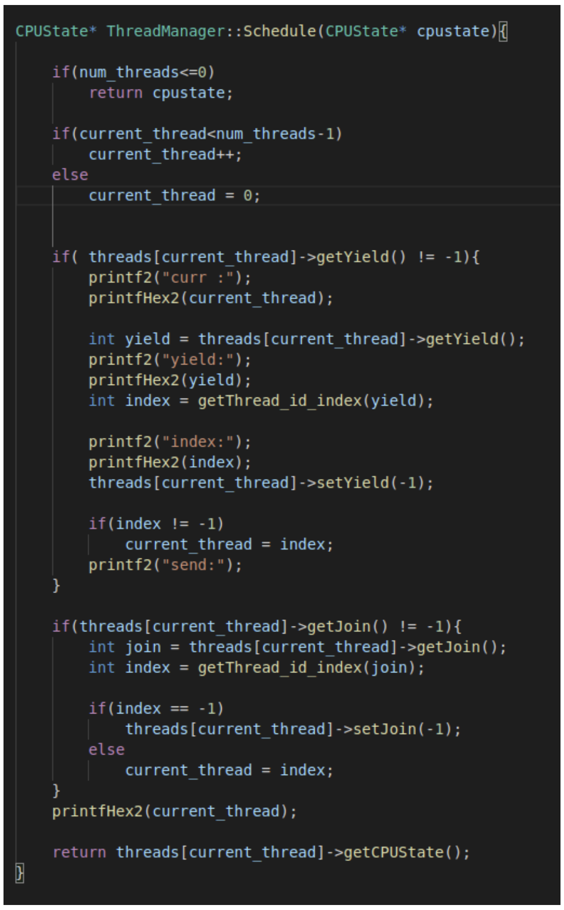
Our Task manager algorithm is very basic store the current register to task table then it runs the next task.



```
CPUState* TaskManager::Schedule(CPUState* cpustate)
{
    if(numTasks <= 0)
        return cpustate;

    if(currentTask >= 0)
        tasks[currentTask]->getCurrentThread()->setCPUState(cpustate);

    if(++currentTask >= numTasks)
        currentTask %= numTasks;

    return tasks[currentTask]->thread_manager->Schedule(cpustate);
}
```

When we came to the thread manager scheduler algorithm is little complex according to the task manager scheduler algorithm because of the thread_yield and thread_joint methods.

In case of these method our algorithm is showing the following figure.



```
CPUState* ThreadManager::Schedule(CPUState* cpustate){

    if(num_threads<=0)
        return cpustate;

    if(current_thread<num_threads-1)
        current_thread++;
    else
        current_thread = 0;


    if( threads[current_thread]->getYield() != -1){
        printf2("curr :");
        printfHex2(current_thread);

        int yield = threads[current_thread]->getYield();
        printf2("yield:");
        printfHex2(yield);
        int index = getThread_id_index(yield);

        printf2("index:");
        printfHex2(index);
        threads[current_thread]->setYield(-1);

        if(index != -1)
            current_thread = index;
        printf2("send:");
    }

    if(threads[current_thread]->getJoin() != -1){
        int join = threads[current_thread]->getJoin();
        int index = getThread_id_index(join);

        if(index == -1)
            threads[current_thread]->setJoin(-1);
        else
            current_thread = index;
    }
    printfHex2(current_thread);

    return threads[current_thread]->getCPUState();
}
```

Increasing the current thread index to pick the next thread.

If a thread called yield by another thread it gives the its turn to other thread for one time.

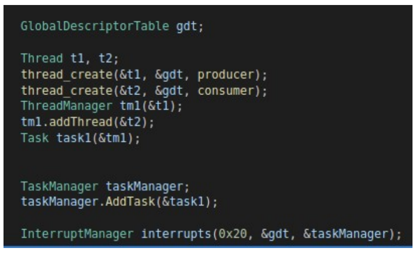
If a thread joint by another thread its gives the turn the other thread until the thread is terminate.

Return the chosen thread CPU state according to the scheduler algorithm.

Producer Consumer Example

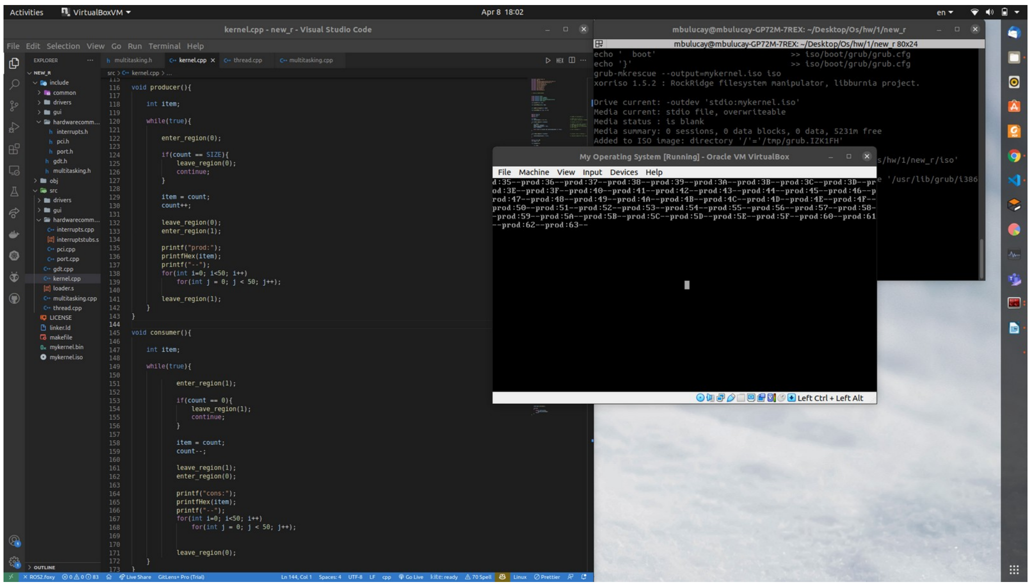
In the producer consumer example changes the waiting time in the producer and consumer if we don’t wait in the function producer will produce until when an interrupt happen. And when the interrupt happened consumer consume the produced item in the list until another interrupt happened.

So that the result may change the wait time in the functions as an example added a few screen shoots



```
GlobalDescriptorTable gdt;

Thread t1, t2;
thread_create(&t1, &gdt, producer);
thread_create(&t2, &gdt, consumer);
ThreadManager tm1(&t1);
tm1.addThread(&t2);
Task task1(&tm1);


TaskManager taskManager;
taskManager.AddTask(&task1);

InterruptManager interrupts(0x20, &gdt, &taskManager);
```

Short Wait:

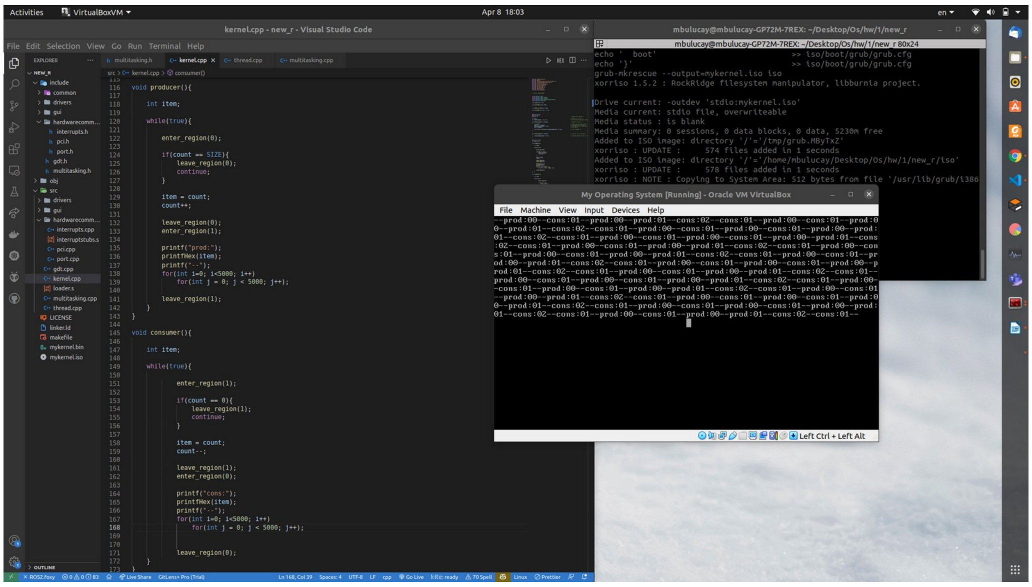


```
void producer(){

    int item;

    while(true){

        enter_region(0);

        if(count == SIZE){
            leave_region(0);
            continue;
        }

        item = count;
        count++;

        leave_region(0);
        enter_region(1);

        printf("prod:");
        printfHex(item);
        printf("--");
        for(int i=0; i<50; i++)
            for(int j = 0; j < 50; j++);

        leave_region(1);
    }
}

void consumer(){

    int item;

    while(true){

            enter_region(1);

            if(count == 0){
                leave_region(1);
                continue;
            }

            item = count;
            count--;

            leave_region(1);
            enter_region(0);

            printf("cons:");
            printfHex(item);
            printf("--");
            for(int i=0; i<50; i++)
                for(int j = 0; j < 50; j++);


            leave_region(0);
    }
}
```

```
echo '  boot'                              >> iso/boot/grub/grub.cfg
echo '}'                                   >> iso/boot/grub/grub.cfg
grub-mkrescue --output=mykernel.iso iso
xorriso 1.5.2 : RockRidge filesystem manipulator, libburnia project.

Drive current: -outdev 'stdio:mykernel.iso'
Media current: stdio file, overwriteable
Media status : is blank
Media summary: 0 sessions, 0 data blocks, 0 data, 5231m free
Added to ISO image: directory '/'='/tmp/grub.IZK1FH'
s/hw/1/new_r/iso'
e '/usr/lib/grub/i386
```

```
d:35--prod:36--prod:37--prod:38--prod:39--prod:3A--prod:3B--prod:3C--prod:3D--pr
od:3E--prod:3F--prod:40--prod:41--prod:42--prod:43--prod:44--prod:45--prod:46--p
rod:47--prod:48--prod:49--prod:4A--prod:4B--prod:4C--prod:4D--prod:4E--prod:4F--
prod:50--prod:51--prod:52--prod:53--prod:54--prod:55--prod:56--prod:57--prod:58-
-prod:59--prod:5A--prod:5B--prod:5C--prod:5D--prod:5E--prod:5F--prod:60--prod:61
--prod:62--prod:63--
```

Long Wait:

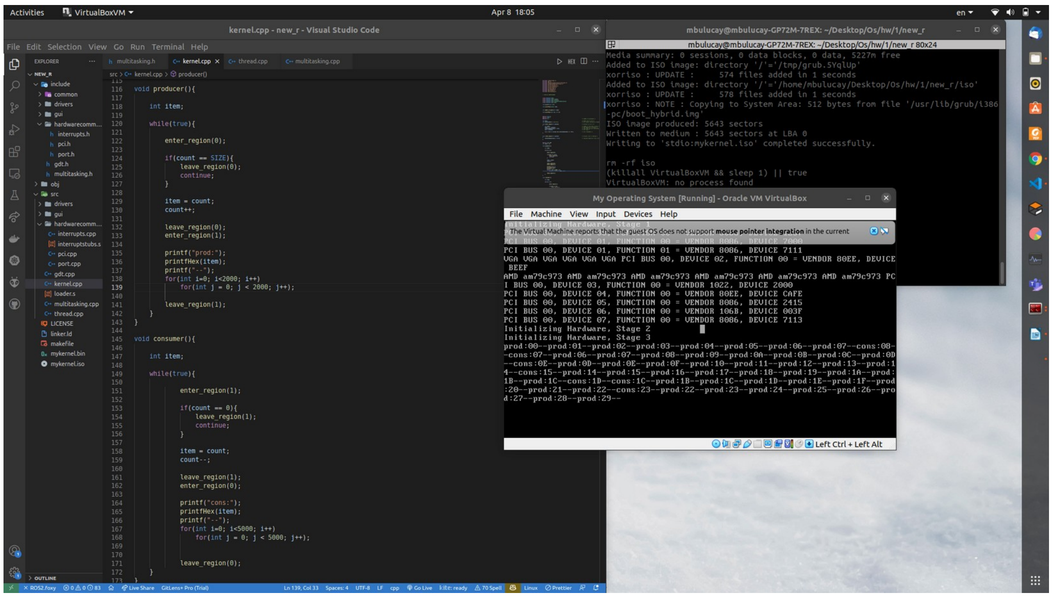


```
void producer(){

    int item;

    while(true){

        enter_region(0);

        if(count == SIZE){
            leave_region(0);
            continue;
        }

        item = count;
        count++;

        leave_region(0);
        enter_region(1);

        printf("prod:");
        printfHex(item);
        printf("--");
        for(int i=0; i<5000; i++)
            for(int j = 0; j < 5000; j++);

        leave_region(1);
    }
}

void consumer(){

    int item;

    while(true){

            enter_region(1);

            if(count == 0){
                leave_region(1);
                continue;
            }

            item = count;
            count--;

            leave_region(1);
            enter_region(0);

            printf("cons:");
            printfHex(item);
            printf("--");
            for(int i=0; i<5000; i++)
                for(int j = 0; j < 5000; j++);


            leave_region(0);
    }
}
```

```
echo '  boot'                              >> iso/boot/grub/grub.cfg
echo '}'                                   >> iso/boot/grub/grub.cfg
grub-mkrescue --output=mykernel.iso iso
xorriso 1.5.2 : RockRidge filesystem manipulator, libburnia project.

Drive current: -outdev 'stdio:mykernel.iso'
Media current: stdio file, overwriteable
Media status : is blank
Media summary: 0 sessions, 0 data blocks, 0 data, 5230m free
Added to ISO image: directory '/'='/tmp/grub.MByTxZ'
xorriso : UPDATE :     574 files added in 1 seconds
Added to ISO image: directory '/'='/home/mbulucay/Desktop/Os/hw/1/new_r/iso'
xorriso : UPDATE :     578 files added in 1 seconds
xorriso : NOTE : Copying to System Area: 512 bytes from file '/usr/lib/grub/i386
```

```
--prod:00--cons:01--prod:00--prod:01--cons:02--cons:01--prod:00--cons:01--prod:0
0--prod:01--cons:02--cons:01--prod:00--cons:01--prod:00--cons:01--prod:00--prod:
01--cons:02--cons:01--prod:00--cons:01--prod:00--cons:01--prod:00--prod:01--cons
:02--cons:01--prod:00--cons:01--prod:00--prod:01--cons:02--cons:01--prod:00--con
s:01--prod:00--cons:01--prod:00--prod:01--cons:02--cons:01--prod:00--cons:01--pr
od:00--prod:01--cons:02--cons:01--prod:00--cons:01--prod:00--cons:01--prod:00--p
rod:01--cons:02--cons:01--prod:00--cons:01--prod:00--prod:01--cons:02--cons:01--
prod:00--cons:01--prod:00--cons:01--prod:00--prod:01--cons:02--cons:01--prod:00-
-cons:01--prod:00--cons:01--prod:00--prod:01--cons:02--cons:01--prod:00--cons:01
--prod:00--prod:01--cons:02--cons:01--prod:00--cons:01--prod:00--cons:01--prod:0
0--prod:01--cons:02--cons:01--prod:00--cons:01--prod:00--cons:01--prod:00--prod:
01--cons:02--cons:01--prod:00--cons:01--prod:00--prod:01--cons:02--cons:01--
```

Unbalance Wait:

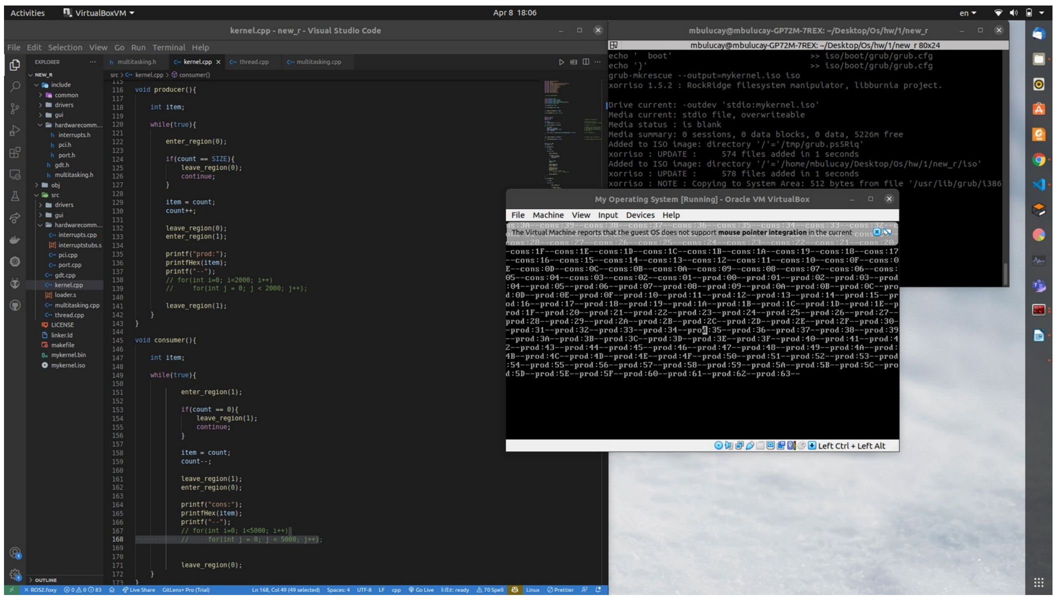

kernel.cpp - new_r - Visual Studio Code
mbulucay@mbulucay-GP72M-7REX: ~/Desktop/Os/hw/1/new_r
My Operating System [Running] - Oracle VM VirtualBox
The Virtual Machine reports that the guest OS does not support mouse pointer integration in the current
Ln 139, Col 33

```cpp
void producer(){

    int item;

    while(true){

        enter_region(0);

        if(count == SIZE){
            leave_region(0);
            continue;
        }

        item = count;
        count++;

        leave_region(0);
        enter_region(1);

        printf("prod:");
        printfHex(item);
        printf("--");
        for(int i=0; i<2000; i++)
            for(int j = 0; j < 2000; j++);

        leave_region(1);
    }
}

void consumer(){

    int item;

    while(true){

            enter_region(1);

            if(count == 0){
                leave_region(1);
                continue;
            }

            item = count;
            count--;

            leave_region(1);
            enter_region(0);

            printf("cons:");
            printfHex(item);
            printf("--");
            for(int i=0; i<5000; i++)
                for(int j = 0; j < 5000; j++);


            leave_region(0);
    }
}
```

```
Media summary: 0 sessions, 0 data blocks, 0 data, 5227m free
Added to ISO image: directory '/'='/tmp/grub.5YqlUp'
xorriso : UPDATE :     574 files added in 1 seconds
Added to ISO image: directory '/'='/home/mbulucay/Desktop/Os/hw/1/new_r/iso'
xorriso : UPDATE :     578 files added in 1 seconds
xorriso : NOTE : Copying to System Area: 512 bytes from file '/usr/lib/grub/i386
-pc/boot_hybrid.img'
ISO image produced: 5643 sectors
Written to medium : 5643 sectors at LBA 0
Writing to 'stdio:mykernel.iso' completed successfully.

rm -rf iso
(killall VirtualBoxVM && sleep 1) || true
VirtualBoxVM: no process found
```

```
Initializing Hardware, Stage 1
PCI BUS 00, DEVICE 01, FUNCTION 00 = VENDOR 8086, DEVICE 7000
PCI BUS 00, DEVICE 01, FUNCTION 01 = VENDOR 8086, DEVICE 7111
VGA VGA VGA VGA VGA VGA PCI BUS 00, DEVICE 02, FUNCTION 00 = VENDOR 80EE, DEVICE
 BEEF
AMD am79c973 AMD am79c973 AMD am79c973 AMD am79c973 AMD am79c973 AMD am79c973 PC
I BUS 00, DEVICE 03, FUNCTION 00 = VENDOR 1022, DEVICE 2000
PCI BUS 00, DEVICE 04, FUNCTION 00 = VENDOR 80EE, DEVICE CAFE
PCI BUS 00, DEVICE 05, FUNCTION 00 = VENDOR 8086, DEVICE 2415
PCI BUS 00, DEVICE 06, FUNCTION 00 = VENDOR 106B, DEVICE 003F
PCI BUS 00, DEVICE 07, FUNCTION 00 = VENDOR 8086, DEVICE 7113
Initializing Hardware, Stage 2
Initializing Hardware, Stage 3
prod:00--prod:01--prod:02--prod:03--prod:04--prod:05--prod:06--prod:07--cons:08-
-cons:07--prod:06--prod:07--prod:08--prod:09--prod:0A--prod:0B--prod:0C--prod:0D
--cons:0E--prod:0D--prod:0E--prod:0F--prod:10--prod:11--prod:12--prod:13--prod:1
4--cons:15--prod:14--prod:15--prod:16--prod:17--prod:18--prod:19--prod:1A--prod:
1B--prod:1C--cons:1D--cons:1C--prod:1B--prod:1C--prod:1D--prod:1E--prod:1F--prod
:20--prod:21--prod:22--cons:23--prod:22--prod:23--prod:24--prod:25--prod:26--pro
d:27--prod:28--prod:29--
```

No Wait:


kernel.cpp - new_r - Visual Studio Code
mbulucay@mbulucay-GP72M-7REX: ~/Desktop/Os/hw/1/new_r
My Operating System [Running] - Oracle VM VirtualBox
The Virtual Machine reports that the guest OS does not support mouse pointer integration in the current
Ln 168, Col 49 (49 selected)

```cpp
void producer(){

    int item;

    while(true){

        enter_region(0);

        if(count == SIZE){
            leave_region(0);
            continue;
        }

        item = count;
        count++;

        leave_region(0);
        enter_region(1);

        printf("prod:");
        printfHex(item);
        printf("--");
        // for(int i=0; i<2000; i++)
        //     for(int j = 0; j < 2000; j++);

        leave_region(1);
    }
}

void consumer(){

    int item;

    while(true){

            enter_region(1);

            if(count == 0){
                leave_region(1);
                continue;
            }

            item = count;
            count--;

            leave_region(1);
            enter_region(0);

            printf("cons:");
            printfHex(item);
            printf("--");
            // for(int i=0; i<5000; i++)
            //     for(int j = 0; j < 5000; j++);


            leave_region(0);
    }
}
```

```
echo '  boot'                              >> iso/boot/grub/grub.cfg
echo '}'                                   >> iso/boot/grub/grub.cfg
grub-mkrescue --output=mykernel.iso iso
xorriso 1.5.2 : RockRidge filesystem manipulator, libburnia project.

Drive current: -outdev 'stdio:mykernel.iso'
Media current: stdio file, overwriteable
Media status : is blank
Media summary: 0 sessions, 0 data blocks, 0 data, 5226m free
Added to ISO image: directory '/'='/tmp/grub.ps5Riq'
xorriso : UPDATE :     574 files added in 1 seconds
Added to ISO image: directory '/'='/home/mbulucay/Desktop/Os/hw/1/new_r/iso'
xorriso : UPDATE :     578 files added in 1 seconds
xorriso : NOTE : Copying to System Area: 512 bytes from file '/usr/lib/grub/i386
```

```
ns:3A--cons:39--cons:38--cons:37--cons:36--cons:35--cons:34--cons:33--cons:32--c
cons:28--cons:27--cons:26--cons:25--cons:24--cons:23--cons:22--cons:21--cons:20-
-cons:1F--cons:1E--cons:1D--cons:1C--cons:1B--cons:1A--cons:19--cons:18--cons:17
--cons:16--cons:15--cons:14--cons:13--cons:12--cons:11--cons:10--cons:0F--cons:0
E--cons:0D--cons:0C--cons:0B--cons:0A--cons:09--cons:08--cons:07--cons:06--cons:
05--cons:04--cons:03--cons:02--cons:01--prod:00--prod:01--prod:02--prod:03--prod
:04--prod:05--prod:06--prod:07--prod:08--prod:09--prod:0A--prod:0B--prod:0C--pro
d:0D--prod:0E--prod:0F--prod:10--prod:11--prod:12--prod:13--prod:14--prod:15--pr
od:16--prod:17--prod:18--prod:19--prod:1A--prod:1B--prod:1C--prod:1D--prod:1E--p
rod:1F--prod:20--prod:21--prod:22--prod:23--prod:24--prod:25--prod:26--prod:27--
prod:28--prod:29--prod:2A--prod:2B--prod:2C--prod:2D--prod:2E--prod:2F--prod:30-
-prod:31--prod:32--prod:33--prod:34--prod:35--prod:36--prod:37--prod:38--prod:39
--prod:3A--prod:3B--prod:3C--prod:3D--prod:3E--prod:3F--prod:40--prod:41--prod:4
2--prod:43--prod:44--prod:45--prod:46--prod:47--prod:48--prod:49--prod:4A--prod:
4B--prod:4C--prod:4D--prod:4E--prod:4F--prod:50--prod:51--prod:52--prod:53--prod
:54--prod:55--prod:56--prod:57--prod:58--prod:59--prod:5A--prod:5B--prod:5C--pro
d:5D--prod:5E--prod:5F--prod:60--prod:61--prod:62--prod:63--
```