Gebze Technical University

Department Of Computer Engineering

CSE 312 /CSE 504 Spring 2023

Operating Systems

Homework #01
Makeup and Bonus

Due Date: 30.04.2023

Abdurrahman Bulut
1901042258

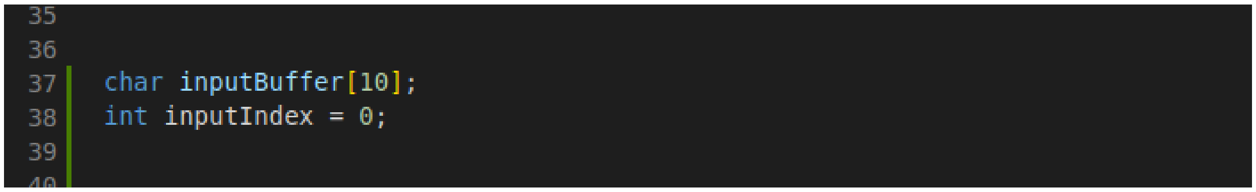
INTRODUCTION

In this assignment, it was requested to add some new things on the implemented versions of the previous assignment. 3 different implementations were sent to us. I selected the second folder. Because only it works fine on my device. The 3rd one does not include keyboard and mouse driver files. Therefore, I chose the 2nd folder. In the first assignment, it is asked to use timer interrupts and I wouldnt it. I explain what I did in the first assignment and what I couldnt that included a timer interrupt. In this homework, it is asked to add mouse and keyboard interrupts to one of the project folders. In this project folder, syscalls were implemented also. The kernel can add multiple programs into memory.

In short, most of what was mentioned in the first assignment and the current assignment has already been implemented. As far as I understand, only mouse and keyboard interrupts will be added and numbers will be sent over the keyboard to the algorithms that will be tested for different life cycles. So I did what I understood to be implemented.

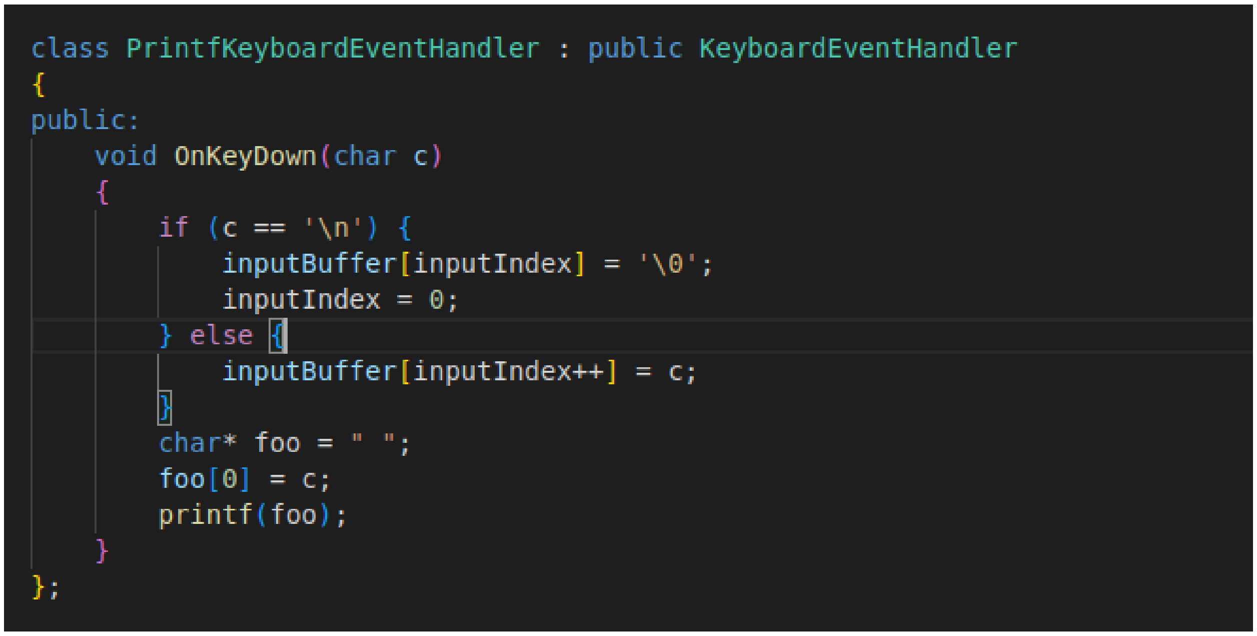
IMPLEMENTATIONS

Keyboard and mouse drivers were added to the kernel.cpp file. I needed to use them. The keyboard driver was implemented before. I use a combination of hardware communication and event handler. Its job is to handle interrupts and send keys to the event handler. There was a class in the kernel.cpp file called PrintKeyboardEventHandler. It has a method called OnKeyDown. It was responsible for processing key events.In my implementation I created a global variable and saved the value on the buffer of the event handler. It stores until the user press Enter. I used this buffer for search algorithms and in other input needed parts.



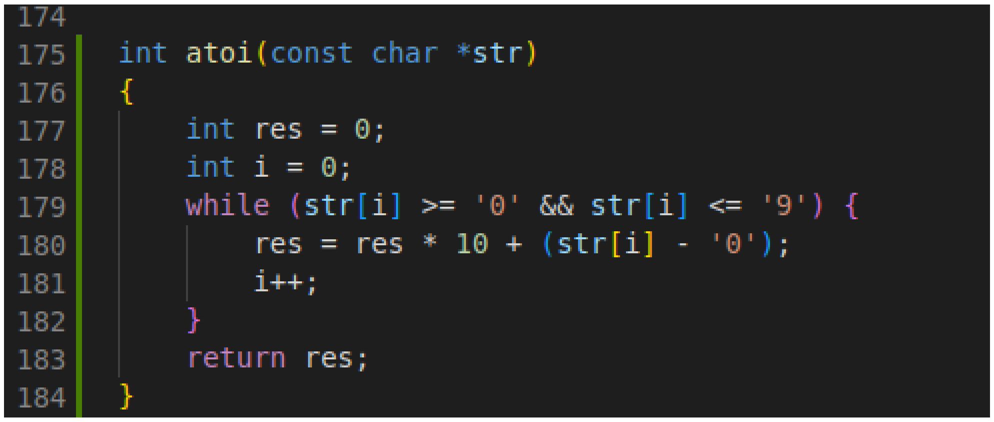
```
char inputBuffer[10];
int inputIndex = 0;
```

I have changed the OnKeyDown method to store value in the buffer that I created globally.



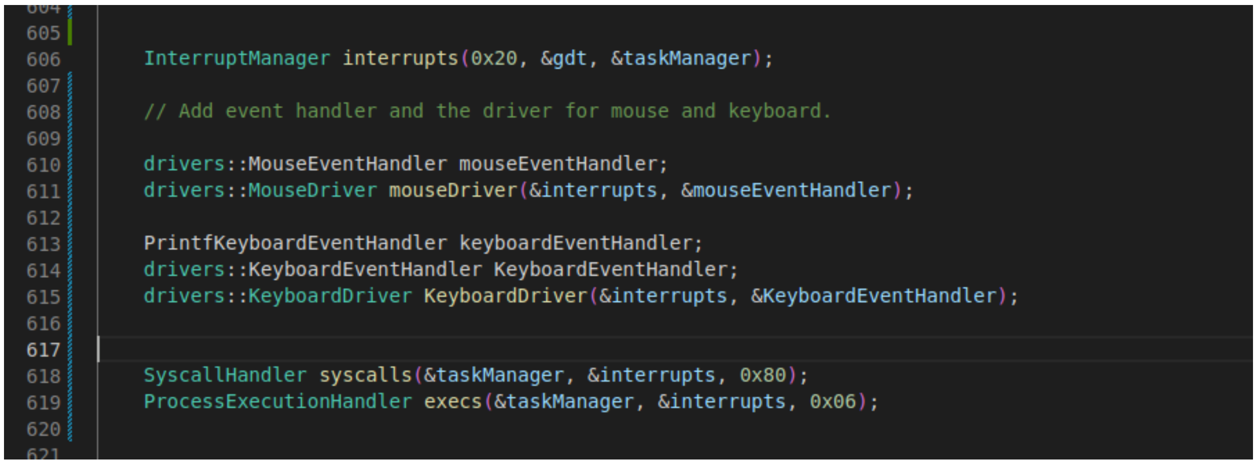
```
class PrintfKeyboardEventHandler : public KeyboardEventHandler
{
public:
    void OnKeyDown(char c)
    {
        if (c == '\n') {
            inputBuffer[inputIndex] = '\0';
            inputIndex = 0;
        } else {
            inputBuffer[inputIndex++] = c;
        }
        char* foo = " ";
        foo[0] = c;
        printf(foo);
    }
};
```

Also I created a method called atoi to convert string to number.



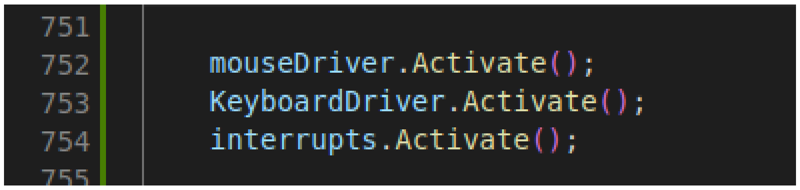
```

int atoi(const char *str)
{
    int res = 0;
    int i = 0;
    while (str[i] >= '0' && str[i] <= '9') {
        res = res * 10 + (str[i] - '0');
        i++;
    }
    return res;
}
```

I added setup codes for drivers and event handlers of mouse and keyboard.I also created an object from PrintKeyboardEventHandler to read input from keyboard.



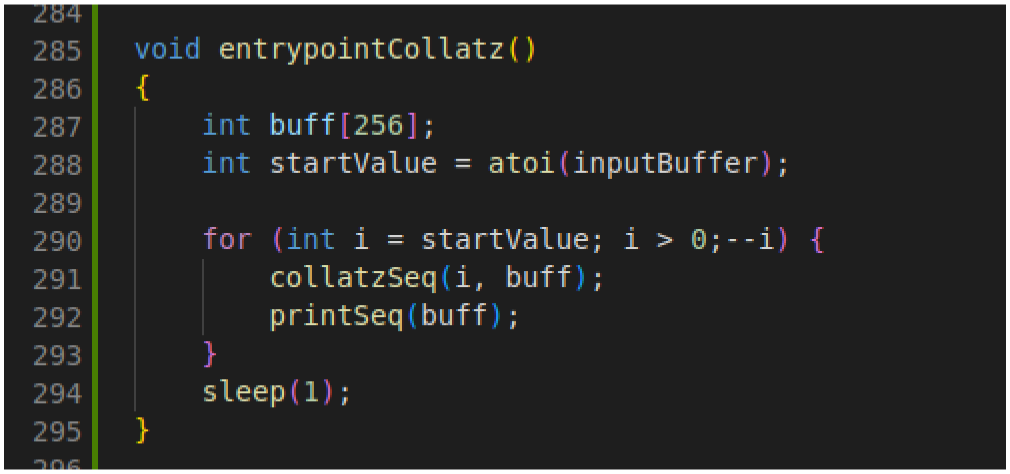
```
    InterruptManager interrupts(0x20, &gdt, &taskManager);

    // Add event handler and the driver for mouse and keyboard.

    drivers::MouseEventHandler mouseEventHandler;
    drivers::MouseDriver mouseDriver(&interrupts, &mouseEventHandler);

    PrintfKeyboardEventHandler keyboardEventHandler;
    drivers::KeyboardEventHandler KeyboardEventHandler;
    drivers::KeyboardDriver KeyboardDriver(&interrupts, &KeyboardEventHandler);


    SyscallHandler syscalls(&taskManager, &interrupts, 0x80);
    ProcessExecutionHandler execs(&taskManager, &interrupts, 0x06);
```

I activated the drivers to use them.



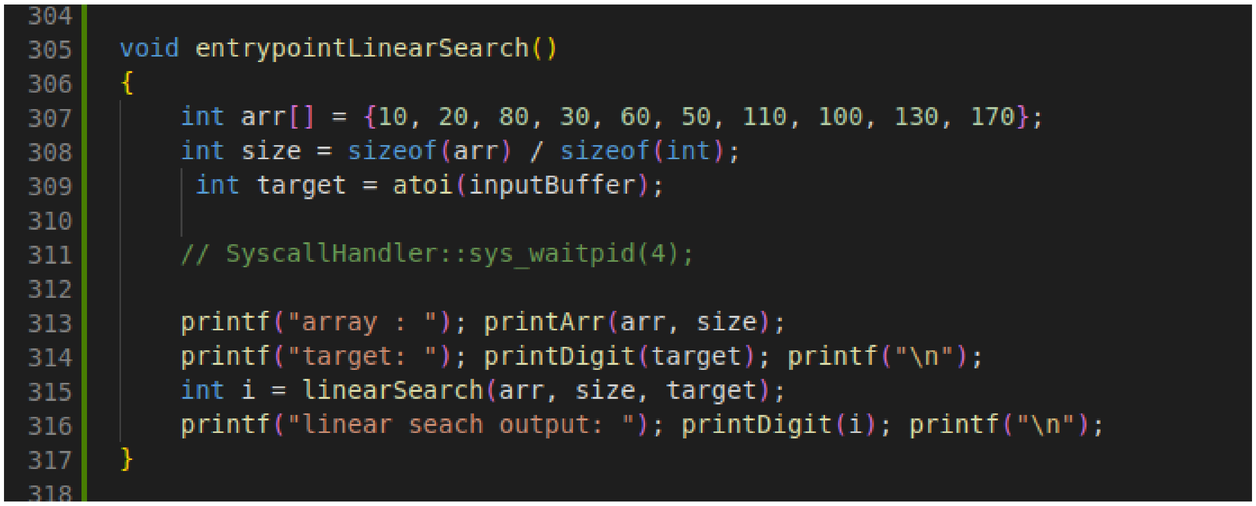
```
    mouseDriver.Activate();
    KeyboardDriver.Activate();
    interrupts.Activate();
```

I changed the entryPointCollatz method to use the user input value.

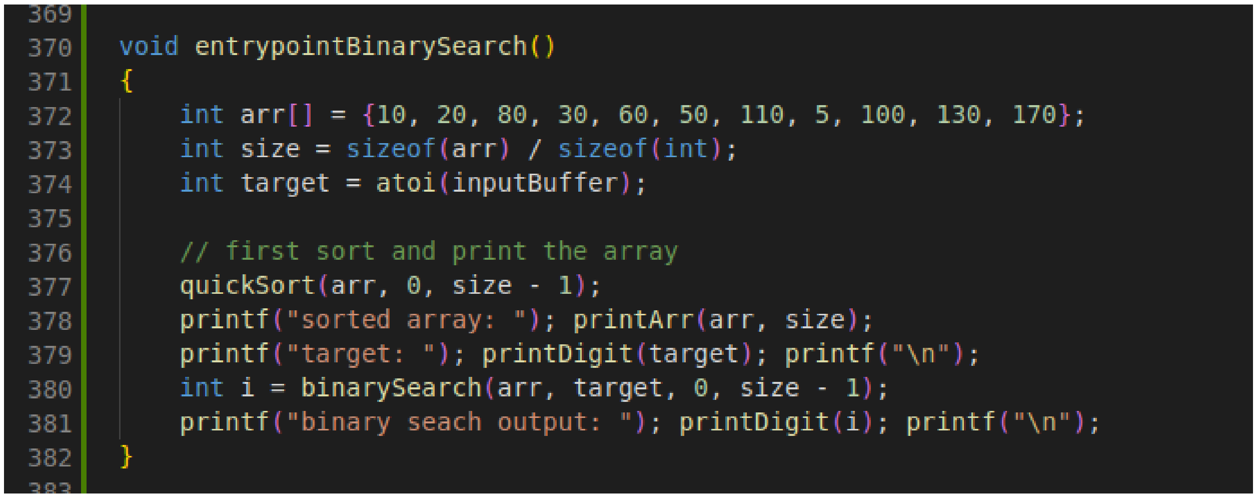


```
void entrypointCollatz()
{
    int buff[256];
    int startValue = atoi(inputBuffer);

    for (int i = startValue; i > 0;--i) {
        collatzSeq(i, buff);
        printSeq(buff);
    }
    sleep(1);
}
```

I also changed the entrypointLinearSearch method and entrypointBinarySearch method.

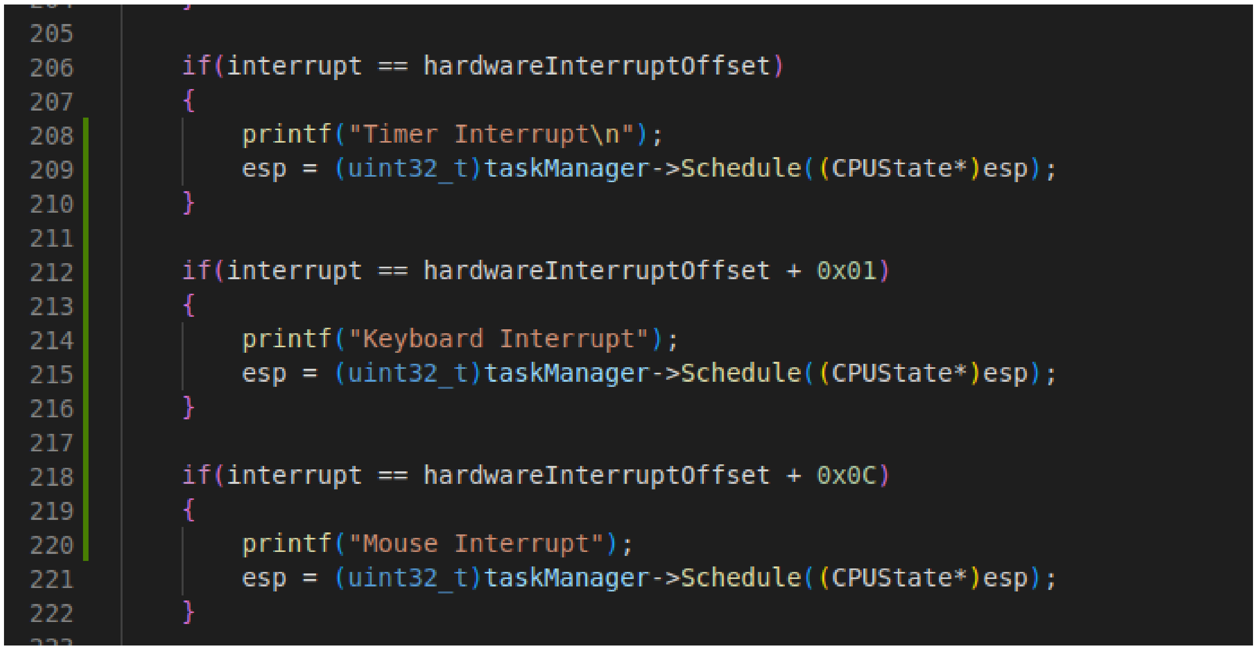


```
void entrypointLinearSearch()
{
    int arr[] = {10, 20, 80, 30, 60, 50, 110, 100, 130, 170};
    int size = sizeof(arr) / sizeof(int);
     int target = atoi(inputBuffer);

    // SyscallHandler::sys_waitpid(4);

    printf("array : "); printArr(arr, size);
    printf("target: "); printDigit(target); printf("\n");
    int i = linearSearch(arr, size, target);
    printf("linear seach output: "); printDigit(i); printf("\n");
}
```



```
void entrypointBinarySearch()
{
    int arr[] = {10, 20, 80, 30, 60, 50, 110, 5, 100, 130, 170};
    int size = sizeof(arr) / sizeof(int);
    int target = atoi(inputBuffer);

    // first sort and print the array
    quickSort(arr, 0, size - 1);
    printf("sorted array: "); printArr(arr, size);
    printf("target: "); printDigit(target); printf("\n");
    int i = binarySearch(arr, target, 0, size - 1);
    printf("binary seach output: "); printDigit(i); printf("\n");
}
```

I made some changes to interrupts.cpp file to catch mouse and keyboard interrupts.

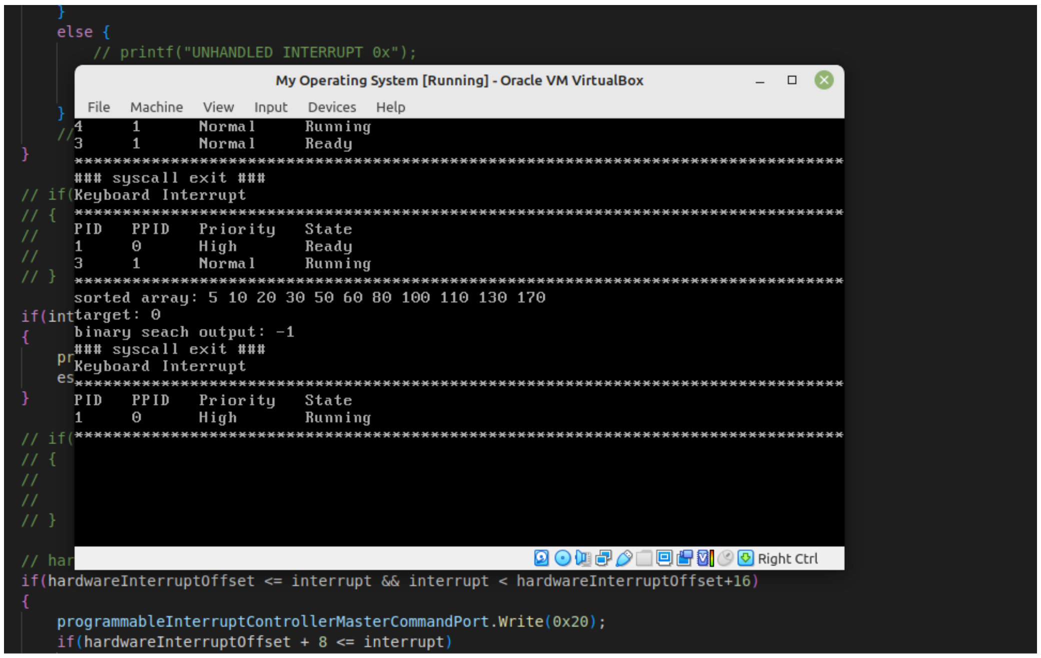


```
    if(interrupt == hardwareInterruptOffset)
    {
        printf("Timer Interrupt\n");
        esp = (uint32_t)taskManager->Schedule((CPUState*)esp);
    }

    if(interrupt == hardwareInterruptOffset + 0x01)
    {
        printf("Keyboard Interrupt");
        esp = (uint32_t)taskManager->Schedule((CPUState*)esp);
    }

    if(interrupt == hardwareInterruptOffset + 0x0C)
    {
        printf("Mouse Interrupt");
        esp = (uint32_t)taskManager->Schedule((CPUState*)esp);
    }
```

The number 1 which is 0x01 as hexadecimal value is often used for keyboard interrupts. If a key is pressed on the keyboard, an interrupt signal will be sent to the CPU. Therefore I added 0x01 to hardwareInterruptOffset for keyboard interrupt. The test screenshots are below in the Test title. And the number 12 which is 0x0C as hexadecimal value is often used for mouse interrupts. So I used those hexadecimal values to handle each interrupt.

TEST

When I press a key on my keyboard, a new process will come. In this example, I stopped the timer interrupt and mouse interrupt.



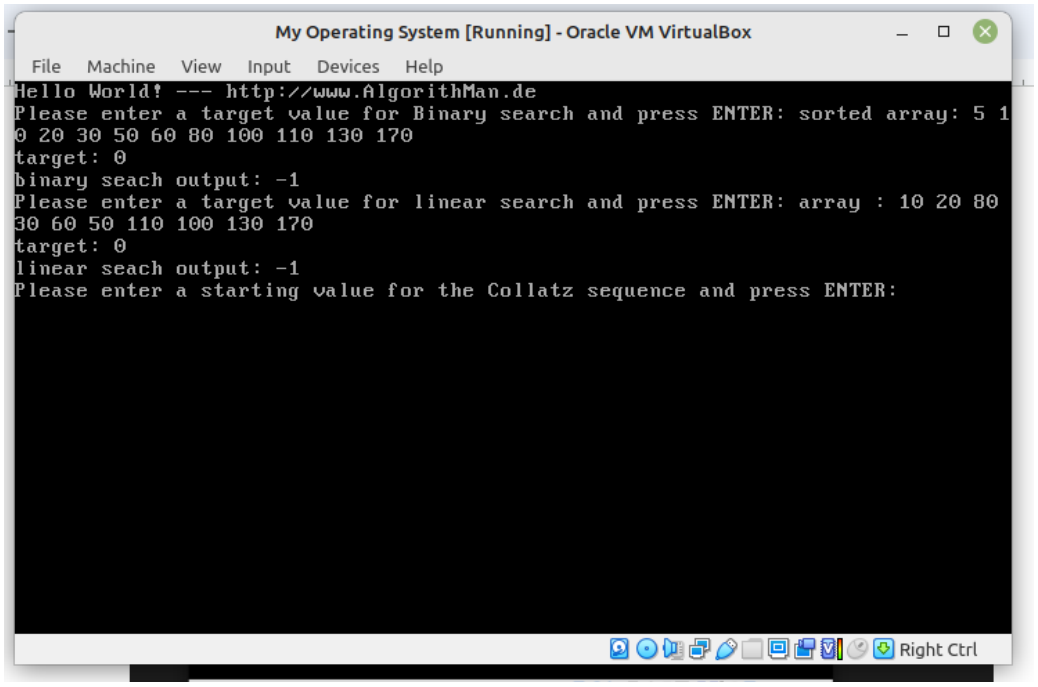
```
    }
    else {
        // printf("UNHANDLED INTERRUPT 0x");
```

My Operating System [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Right Ctrl

```
4     1      Normal     Running
3     1      Normal     Ready
********************************************************************************
### syscall exit ###
Keyboard Interrupt
********************************************************************************
PID   PPID   Priority   State
1     0      High       Ready
3     1      Normal     Running
********************************************************************************
sorted array: 5 10 20 30 50 60 80 100 110 130 170
target: 0
binary seach output: -1
### syscall exit ###
Keyboard Interrupt
********************************************************************************
PID   PPID   Priority   State
1     0      High       Running
********************************************************************************
```

```
if(hardwareInterruptOffset <= interrupt && interrupt < hardwareInterruptOffset+16)
{
    programmableInterruptControllerMasterCommandPort.Write(0x20);
    if(hardwareInterruptOffset + 8 <= interrupt)
```

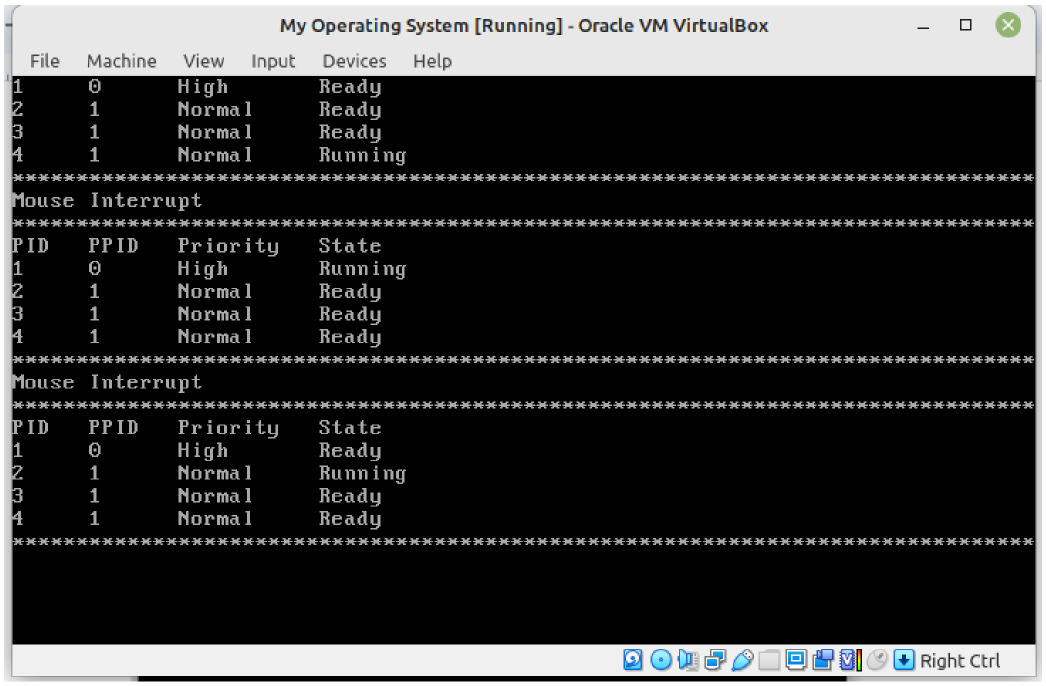
If I close keyboard interrupt and timer interrupt and enable only mouse interrupt, this screen will come on. Mouse is not captured by Virtual Machine at the beginning.



My Operating System [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Right Ctrl

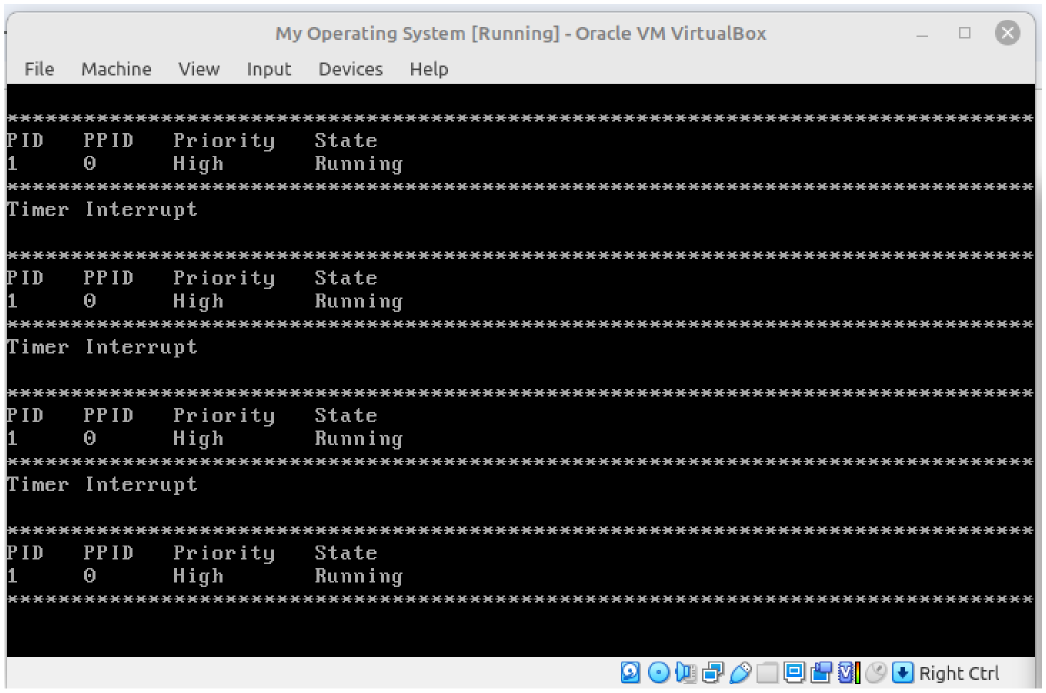
```
Hello World! --- http://www.AlgorithMan.de
Please enter a target value for Binary search and press ENTER: sorted array: 5 1
0 20 30 50 60 80 100 110 130 170
target: 0
binary seach output: -1
Please enter a target value for linear search and press ENTER: array : 10 20 80
30 60 50 110 100 130 170
target: 0
linear seach output: -1
Please enter a starting value for the Collatz sequence and press ENTER:
```

After clicking the virtual machine window area, the Virtual Machine capture the mouse pointer and the screen will be something like this.


My Operating System [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Right Ctrl

```
1      0      High       Ready
2      1      Normal     Ready
3      1      Normal     Ready
4      1      Normal     Running
********************************************************************************
Mouse Interrupt
********************************************************************************
PID    PPID   Priority   State
1      0      High       Running
2      1      Normal     Ready
3      1      Normal     Ready
4      1      Normal     Ready
********************************************************************************
Mouse Interrupt
********************************************************************************
PID    PPID   Priority   State
1      0      High       Ready
2      1      Normal     Running
3      1      Normal     Ready
4      1      Normal     Ready
********************************************************************************
```

When I enable all interrupts, keyboard and mouse actions will cause an interrupt.


My Operating System [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Right Ctrl

```
********************************************************************************
PID    PPID   Priority   State
1      0      High       Running
********************************************************************************
Timer Interrupt

********************************************************************************
PID    PPID   Priority   State
1      0      High       Running
********************************************************************************
Timer Interrupt

********************************************************************************
PID    PPID   Priority   State
1      0      High       Running
********************************************************************************
Timer Interrupt

********************************************************************************
PID    PPID   Priority   State
1      0      High       Running
********************************************************************************
```

As a result, I implemented what was asked of us. I could not show only the part that sends the value from the keyboard key to specific algorithms. I cannot test it. It is supposed to work.