

LAB 1

INSTALLING WINDOWS AND LINUX

Submitted by: Submitted to:

Nabin Katwal Santosh Dhungana

Class of 2022 Operating System

Roll No: 824

INTRODUCTION

Operating system is a system software which that manages computer hardware, software resources and provides common services for computer programs. For computers hardware to function properly on basic tasks like input and output, memory allocation, the operating system acts as intermediary between programs and the computer hardware. Every operating system is based upon some kernel (Latest version of Windows is based on windows NT kernel whereas Linux distribution are based upon Linux kernel) and with the help of firmware and device driver, the kernel provides the most basic level of control over all of the computer’s hardware device. It manages memory access for programs in the RAM, it determines which programs get access to which hardware resources, it sets up or resets the CPU's operating states for optimal operation at all times, and it organizes the data for long-term non-volatile storage with file systems on such media as disks, tapes, flash memory, etc.

INSTALLING WINDOWS

The process of installing windows on your PC is quite simple, the steps are as follows:

* Download the required ISO file. In case for windows you can download it from Microsoft’s website.
* Burn that ISO file into any USB flash drive using a burning tool.
* Insert that USB device to your PC and boot the device.
* After pressing power button press the appropriate key rapidly/continuously to select correct boot media. Usually the key are F7, F8, F9, F10 and F11. In the rare case it may be F2 and it completely depends on the motherboard manufacturer.
* After clicking the correct key for required time a new pop-up screen will pop up. It has the list of all the storage media connected to that device at that particular time. Choose the correct storage device and press “ENTER” key.
* A new screen will appear containing all the storage device connected to that device. Choose the correct drive and give the volume amount you want to allocate for windows. After selecting correct drive and allocating the required volume it is required to format in NTFS file format.
* Give some time to install windows on the drive. Some pop-up screen may come in the middle of the process. Read carefully what it wants to say and proceed.
* Windows is installed in your device and you are ready to use it.

INSTALLING LINUX

The process of installing Linux on your PC or laptop is quite simple. The steps are as follows:

* Download the required ISO file. In case for Linux one quick google search is enough to locate download file.
* Burn that ISO file onto any USB flash drive using burning/flashing tools.
* Insert that USB drive into your PC or laptop’s USB port and boot the device
* After pressing power button press the appropriate key rapidly/continuously to select correct boot media. Usually the key are F7, F8, F9, F10 and F11. In the rare case it may be F2 and it completely depends on the motherboard manufacturer.
* After clicking the correct key for required time a new pop-up screen will pop up. It has the list of all the storage media connected to that device at that particular time. Choose the correct storage device and press “ENTER” key.
* A new screen will popup which has different options like booting live version, installing it on the device or booting in safe mode.
* If you want to try certain version of Linux then boot into live version and use it onward. If you like that version and want to install then there is install Linux button in the desktop of the live booted Linux.
* If you want to install Linux directly without trying in the live booted environment then choose “Install on the device” option and boot into it.
* After choosing “Install on the device” option basic questions are asked like your keyboard layout, default language of the device and the location of the repositories from where you want to install the packages.
* After this a new window will open up which has certain options like wiping the existing OS in your drive and installing Linux, dual booting Linux along side with existing OS or installing Linux on certain volume of the drive. Choose according to your own needs and continue
* Linux is installed on your device and you are ready to use it.

CONCLUSION

Installing OS either Windows or Linux is quiet an easy process. Even if you haven’t done it before than this you will be able to install it on your device without any problem as long as you are following correct steps and know what you are doing.



LAB 2

COMMON BASIC LINUX COMMANDS

Submitted by: Submitted to:

Nabin Katwal Santosh Dhungana

Class of 2022 Operating System

Roll No: 824

INTRODUCTION

Linux is a family of free and open-source software operating systems based on the Linux kernel, an operating system kernel first released by Linus Torvalds. We installed Linux on our device either on virtual machine or by dual booting our devices and the functionality of the different commands was discovered. The below mentioned commands were executed on Linux terminal. The main objective for us to use Linux is to get familiar with its commands

Here are some lists of commands:

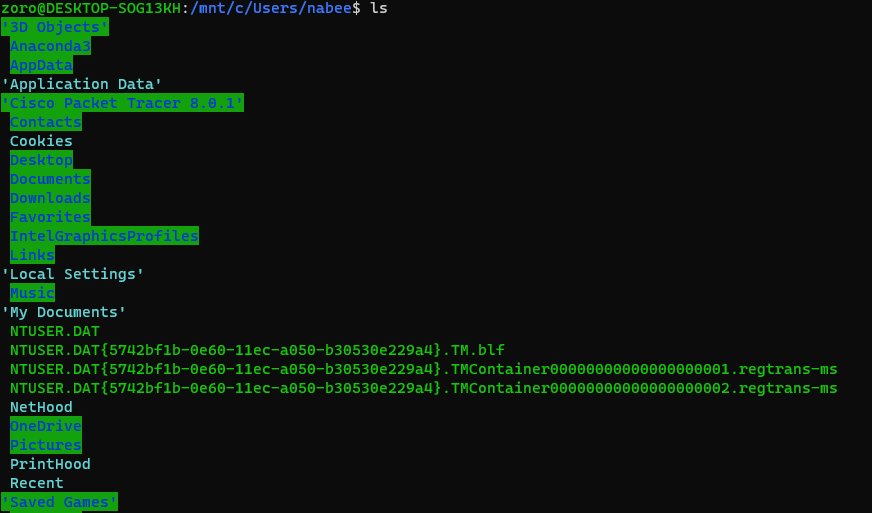
* Pwd

It stands for print working directory. It prints the directory that we are currently working on. The syntax for this command is pwd



* Ls

This command is used to list the files that are present in the current directory. However if you want to see the hidden files then you must type ls-a command. To list all the file in that directory including all the files including in the sub-directory, we must use the command ls -R.



* Cd

The cd command can be used to go to a directory. The syntax for this command is ls <directory name>



* Cd ..

This command is used to go back from a folder that we are currently in to the previous directory. The syntax for this command is cd ..

* Mkdir

This command is used when you need to create a folder or a directory. The syntax for this

command is mkdir



* Rmdir

This command is used when you need to delete an empty directory. The syntax for this command is

rmdir <Directory name>



* Rm

This command is used when we have to delete some files. This command delete files without needing any confirmation from user.

* Touch

This command is used to create a blank new file through the terminal. The syntax for this

command is touch <Filename with extension>

CONCLUSION

From this lab, we tried different Linux command and its outputs were observed. At the end, we can conclude that different commands were tried and the goal of this lab was achieved.



LAB 3

Simulating FCFS CPU Scheduling Algorithm

Submitted by: Submitted to:

Nabin Katwal Santosh Dhungana

Class of 2022 Operating System

Roll No: 824

PROGRAM CODE

class FCFS:

    def \_\_init\_\_(self, process, burst\_time):

        self.process = process

        self.burst\_time = burst\_time

        self.average\_turnaround\_time = 0

        self.waiting\_time = []

        self.average\_waiting\_time = 0

    def waiting\_time\_fn(self):

        for i in range(0, len(self.process)):

            if i == 0:

                self.waiting\_time.append(0)

            else:

                self.waiting\_time.append(self.waiting\_time[i-1]+self.burst\_time[i-1])

        return (self.waiting\_time)

    def average\_waiting\_time\_fn(self):

        self.waiting\_time\_fn()

        self.average\_waiting\_time = sum(self.waiting\_time)/len(self.process)

    def average\_turnaround\_time\_fn(self):

        self.average\_turnaround\_time = (sum(self.waiting\_time)+sum(self.burst\_time))/len(self.process)

    def display\_results(self):

        self.average\_waiting\_time\_fn()

        self.average\_turnaround\_time\_fn()

        print("Processes   Burst time  Waiting Time")

        for i in range(0, len(self.process)):

            print(f"{self.process[i]}            {self.burst\_time[i]}            {self.waiting\_time\_fn()[i]}")

        print('\n')

        print(f"Average waiting time {self.average\_waiting\_time}")

        print('\n')

        print(f'Average turnaround time {self.average\_turnaround\_time}')

if \_\_name\_\_ == "\_\_main\_\_":

    process\_list = []

    burst\_time\_list = []

    while (True):

        process = int(input("Enter process time->"))

        burst\_time = int(input("Enter burst time->"))

        process\_list.append(process)

        burst\_time\_list.append(burst\_time)

        choice = input("Enter another?(y/n)")

        if choice.lower() == 'y':

            continue

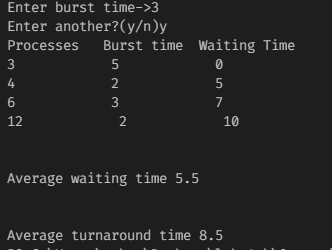
        else:

            break

    batch1 = FCFS(process\_list, burst\_time\_list)

    batch1.display\_results()

OUTPUT





LAB 4

Simulating Shortest Job First Algorithm

Submitted by: Submitted to:

Nabin Katwal Santosh Dhungana

Class of 2022 Operating System

Roll No: 824

PROGRAM CODE

from six.moves import input

t = []

n = int(input("Enter the number of process->"))

processes = []

for i in range(0,n):

    processes.insert(i,i+1)

print("Enter the burst time of the processes: \n")

bt = list(map(int, input().split()))

for i in range(0, len(bt)-1):

    for j in range(0, len(bt)-i-1):

        if(bt[j]>bt[j+1]):

            temp = bt[j]

            bt[j]=bt[j+1]

            bt[j+1] = temp

            temp = processes[j]

            processes[j] = processes[j+1]

            processes[j+1] = temp

wt = []

avgwt = 0

tat = []

avgtat = 0

wt.insert(0,0)

tat.insert(0,bt[0])

for i in range(1, len(bt)):

    wt.insert(i, wt[i-1]+bt[i-1])

    tat.insert(i, wt[i]+bt[i])

    avgwt+=wt[i]

    avgtat += tat[i]

avgwt = float(avgwt)/n

avgtat = float(avgtat)/n

print('\n')

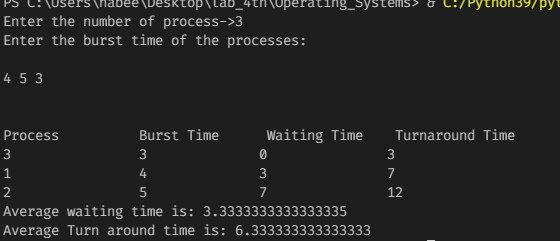
print("Process \t Burst Time\t Waiting Time\t Turnaround Time")

for i in range(0,n):

    print(f"{str(processes[i])} \t\t {str(bt[i])}\t\t{str(wt[i])}\t\t{str(tat[i])}")

print("Average waiting time is: "+str(avgwt))

print("Average Turn around time is: "+str(avgtat))





LAB 5

Simulating Priority CPU Scheduling

Submitted by: Submitted to:

Nabin Katwal Santosh Dhungana

Class of 2022 Operating System

Roll No: 824

PROGRAM CODE

totalprocess = 5

proc = []

for i in range(5):

    l = []

    for j in range(4):

        l.append(0)

    proc.append(l)

# Using FCFS Algorithm to find Waiting time

def get\_wt\_time( wt):

    # declaring service array that stores

    # cumulative burst time

    service = [0] \* 5

    # Initialising initial elements

    # of the arrays

    service[0] = 0

    wt[0] = 0

    for i in range(1, totalprocess):

        service[i] = proc[i - 1][1] + service[i - 1]

        wt[i] = service[i] - proc[i][0] + 1

        # If waiting time is negative,

        # change it o zero

        if(wt[i] < 0) :

            wt[i] = 0

def get\_tat\_time(tat, wt):

    # Filling turnaroundtime array

    for i in range(totalprocess):

        tat[i] = proc[i][1] + wt[i]

def findgc():

    # Declare waiting time and

    # turnaround time array

    wt = [0] \* 5

    tat = [0] \* 5

    wavg = 0

    tavg = 0

    # Function call to find waiting time array

    get\_wt\_time(wt)

    # Function call to find turnaround time

    get\_tat\_time(tat, wt)

    stime = [0] \* 5

    ctime = [0] \* 5

    stime[0] = 1

    ctime[0] = stime[0] + tat[0]

    # calculating starting and ending time

    for i in range(1, totalprocess):

        stime[i] = ctime[i - 1]

        ctime[i] = stime[i] + tat[i] - wt[i]

    print("Process\_no\tStart\_time\tComplete\_time",

            "\tTurn\_Around\_Time\tWaiting\_Time")

    # display the process details

    for i in range(totalprocess):

        wavg += wt[i]

        tavg += tat[i]

        print(proc[i][3], "\t\t", stime[i],

                        "\t\t", end = " ")

        print(ctime[i], "\t\t", tat[i], "\t\t\t", wt[i])

    # display the average waiting time

    # and average turn around time

    print("Average waiting time is : ", end = " ")

    print(wavg / totalprocess)

    print("average turnaround time : " , end = " ")

    print(tavg / totalprocess)

# Driver code

if \_\_name\_\_ =="\_\_main\_\_":

    arrivaltime = [1, 2, 3, 4, 5]

    bursttime = [3, 5, 1, 7, 4]

    priority = [3, 4, 1, 7, 8]

    for i in range(totalprocess):

        proc[i][0] = arrivaltime[i]

        proc[i][1] = bursttime[i]

        proc[i][2] = priority[i]

        proc[i][3] = i + 1

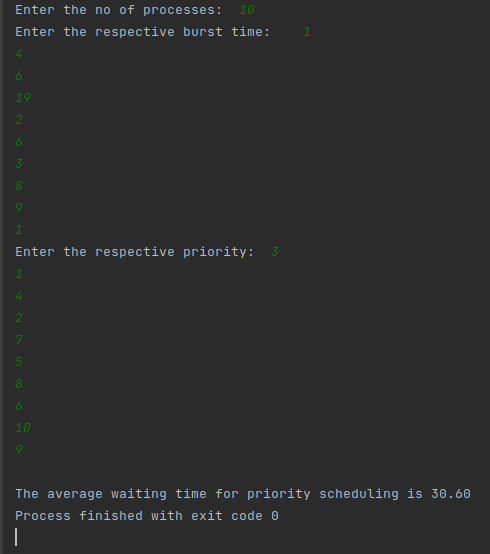
    # Using inbuilt sort function

    proc = sorted (proc, key = lambda x:x[2])

    proc = sorted (proc)

    findgc()

OUTPUT





LAB 6

Simulating Round Robin CPU Scheduling

Submitted by: Submitted to:

Nabin Katwal Santosh Dhungana

Class of 2022 Operating System

Roll No: 824

PROGRAM CODE

class Round\_robin:

    def \_\_init\_\_(self, burst\_time, process, time\_quantum):

        self.process = process

        self.burst\_time = burst\_time

        self.rem\_burst\_time = [burst\_time[i] for i in range(len(process))]

        self.waiting\_time = [0]\*len(process)

        self.time = 0

        self.time\_quantum = time\_quantum

        self.turnaround\_time = [0]\*len(process)

    def calculate\_waiting\_time(self):

        while(1):

            done = True

            for i in range(len(process)):

                if self.rem\_burst\_time[i]>0:

                    done = False

                    if self.rem\_burst\_time[i]>self.time\_quantum:

                        self.time = self.time+self.time\_quantum

                        self.rem\_burst\_time[i] -= self.time\_quantum

                    else:

                        self.time = self.time + self.rem\_burst\_time[i]

                        self.waiting\_time[i]=self.time - self.burst\_time[i]

                        self.rem\_burst\_time[i] = 0

            if done == True:

                    break

        return self.waiting\_time

    def turnaroundtime(self):

        for i in range(len(process)):

            self.turnaround\_time[i] = self.burst\_time[i]+self.waiting\_time[i]

        return self.turnaround\_time

    def display\_results(self):

        self.calculate\_waiting\_time()

        self.turnaroundtime()

        average\_wt = sum(self.waiting\_time)/len(process)

        average\_turnaround = (sum(self.waiting\_time)+sum(self.burst\_time))/len(self.process)

        print("Processes\t Burst Time\t Waiting Time\t Turn around time")

        for i in range(0, len(process)):

            print(f"{self.process[i]}\t\t {self.burst\_time[i]}\t\t {self.waiting\_time[i]}\t\t\t{self.turnaround\_time[i]}")

        print('\n')

        print(f"Average waiting time: {average\_wt}")

        print('\n')

        print(f"Average turnaround time: {average\_turnaround}")

if \_\_name\_\_ == "\_\_main\_\_":

    process = []

    burst\_time = []

    time\_quantum = int(input("Enter time quantum->"))

    while(True):

        process\_in = int(input("Enter process id->"))

        burst\_time\_in = int(input("Enter burst time->"))

        process.append(process\_in)

        burst\_time.append(burst\_time\_in)

        choice = input("Continue Input?(y/n)->")

        if choice.lower() == 'y':

            continue

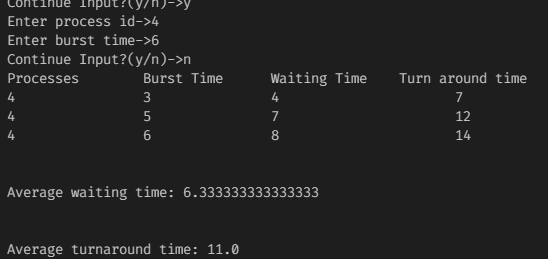
        else:

            break

    round1 = Round\_robin(burst\_time, process, time\_quantum)

    round1.display\_results()

OUTPUT





LAB 7

Simulating Bankers Algorithm

Submitted by: Submitted to:

Nabin Katwal Santosh Dhungana

Class of 2022 Operating System

Roll No: 824

PROGRAM CODE

# Banker's Algorithm

# Driver code:

if \_\_name\_\_=="\_\_main\_\_":

    # P0, P1, P2, P3, P4 are the Process names here

    n = 5 # Number of processes

    m = 3 # Number of resources

    # Allocation Matrix

    alloc = [[0, 1, 0 ],[ 2, 0, 0 ],

            [3, 0, 2 ],[2, 1, 1] ,[ 0, 0, 2]]

    # MAX Matrix

    max = [[7, 5, 3 ],[3, 2, 2 ],

            [ 9, 0, 2 ],[2, 2, 2],[4, 3, 3]]

    avail = [3, 3, 2] # Available Resources

    f = [0]\*n

    ans = [0]\*n

    ind = 0

    for k in range(n):

        f[k] = 0

    need = [[ 0 for i in range(m)]for i in range(n)]

    for i in range(n):

        for j in range(m):

            need[i][j] = max[i][j] - alloc[i][j]

    y = 0

    for k in range(5):

        for i in range(n):

            if (f[i] == 0):

                flag = 0

                for j in range(m):

                    if (need[i][j] > avail[j]):

                        flag = 1

                        break

                if (flag == 0):

                    ans[ind] = i

                    ind += 1

                    for y in range(m):

                        avail[y] += alloc[i][y]

                    f[i] = 1

    print("Following is the SAFE Sequence")

    for i in range(n - 1):

        print(" P", ans[i], " ->", sep="", end="")

    print(" P", ans[n - 1], sep="")

OUTPUT





LAB 8

Simulating FIFO Page Replacement

Submitted by: Submitted to:

Nabin Katwal Santosh Dhungana

Class of 2022 Operating System

Roll No: 824

PROGRAM CODE

from queue import Queue

class FCFS\_PR:

    def \_\_init\_\_(self, pages, capacity):

        self.pages = pages

        self.n = len(self.pages)

        self.capacity = capacity

        self.s = set()

        self.indexes = Queue()

        self.page\_faults = 0

    def page\_fault(self):

        for i in range(self.n):

            if len(self.s)<self.capacity:

                if(self.pages[i] not in self.s):

                    self.s.add(pages[i])

                    self.page\_faults += 1

                    self.indexes.put(pages[i])

            else:

                if (self.pages[i] not in self.s):

                    val = self.indexes.queue[0]

                    self.indexes.get()

                    self.s.remove(val)

                    self.s.add(pages[i])

                    self.indexes.put(pages[i])

                    self.page\_faults += 1

        return self.page\_faults

    def display\_results(self):

        self.page\_fault()

        print(f"{self.pages}")

        print(f"Page faults {self.page\_faults}")

        print(f"Page hits {len(self.pages)-self.page\_faults}")

if \_\_name\_\_ == '\_\_main\_\_':

    pages = []

    capacity = 4

    while(True):

        pages\_in = int(input("Enter pages->"))

        choice = input("Add more pages?(y/n)->")

        pages.append(pages\_in)

        if choice.lower()=='y':

            continue

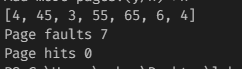
        else:

            break

    page1 = FCFS\_PR(pages, capacity)

    page1.display\_results()

OUTPUT





LAB 9

Simulating LOOK disk scheduling

Submitted by: Submitted to:

Nabin Katwal Santosh Dhungana

Class of 2022 Operating System

Roll No: 824

PROGRAM CODE

size = 8

disk\_size = 200

def LOOK(arr, head, direction):

    seek\_count = 0

    distance = 0

    cur\_track = 0

    left = []

    right = []

    seek\_sequence = []

    for i in range(size):

        if (arr[i] < head):

            left.append(arr[i])

        if (arr[i] > head):

            right.append(arr[i])

    left.sort()

    right.sort()

    run = 2

    while (run):

        if (direction == "left"):

            for i in range(len(left) - 1, -1, -1):

                cur\_track = left[i]

                seek\_sequence.append(cur\_track)

                distance = abs(cur\_track - head)

                seek\_count += distance

                head = cur\_track

            direction = "right"

        elif (direction == "right"):

            for i in range(len(right)):

                cur\_track = right[i]

                seek\_sequence.append(cur\_track)

                distance = abs(cur\_track - head)

                seek\_count += distance

                head = cur\_track

            direction = "left"

        run -= 1

    print("Total number of seek operations =",

        seek\_count)

    print("Seek Sequence is")

    for i in range(len(seek\_sequence)):

        print(seek\_sequence[i])

arr = [ 176, 79, 34, 60, 92, 11, 41, 114 ]

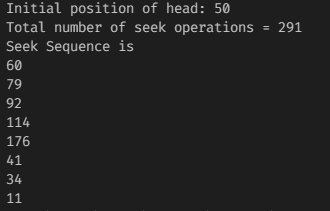
head = 50

direction = "right"

print("Initial position of head:", head)

LOOK(arr, head, direction)

OUTPUT





LAB 10

Simulating Producer Consumer problem

Submitted by: Submitted to:

Nabin Katwal Santosh Dhungana

Class of 2022 Operating System

Roll No: 824

PROGRAM CODE

from threading import Thread, Semaphore

import time

import random

queue = []

MAX\_NUM = 10

sem = Semaphore()

class ProducerThread(Thread):

    def run(self):

        nums = range(5)

        global queue

        while True:

            sem.acquire()

            if len(queue) == MAX\_NUM:

                print ("List is full, producer will wait")

                sem.release()

                print ("Space in queue, Consumer notified the producer")

            num = random.choice(nums)

            queue.append(num)

            print ("Produced", num)

            sem.release()

            time.sleep(random.random())

class ConsumerThread(Thread):

    def run(self):

        global queue

        while True:

            sem.acquire()

            if not queue:

                print ("List is empty, consumer waiting")

                sem.release()

                print ("Producer added something to queue and notified the consumer")

            num = queue.pop(0)

            print ("Consumed", num)

            sem.release()

            time.sleep(random.random())

def main():

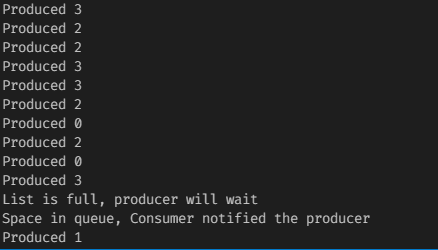
    ProducerThread().start()

    ConsumerThread().start()

if \_\_name\_\_ == '\_\_main\_\_':

    main()

OUTPUT





LAB 11

Simulating SCAN Disk Scheduling algorithm

Submitted by: Submitted to:

Nabin Katwal Santosh Dhungana

Class of 2022 Operating System

Roll No: 824

PROGRAM CODE

size = 8

disk\_size = 200

def SCAN(arr, head, direction):

    seek\_count = 0

    distance, cur\_track = 0, 0

    left = []

    right = []

    seek\_sequence = []

    if (direction == "left"):

        left.append(0)

    elif (direction == "right"):

        right.append(disk\_size - 1)

    for i in range(size):

        if (arr[i] < head):

            left.append(arr[i])

        if (arr[i] > head):

            right.append(arr[i])

    left.sort()

    right.sort()

    run = 2

    while (run != 0):

        if (direction == "left"):

            for i in range(len(left) - 1, -1, -1):

                cur\_track = left[i]

                seek\_sequence.append(cur\_track)

                distance = abs(cur\_track - head)

                seek\_count += distance

                head = cur\_track

            direction = "right"

        elif (direction == "right"):

            for i in range(len(right)):

                cur\_track = right[i]

                seek\_sequence.append(cur\_track)

                distance = abs(cur\_track - head)

                seek\_count += distance

                head = cur\_track

            direction = "left"

        run -= 1

    print("Total number of seek operations =",

        seek\_count)

    print("Seek Sequence is")

    for i in range(len(seek\_sequence)):

        print(seek\_sequence[i])

arr = [ 176, 79, 34, 60,

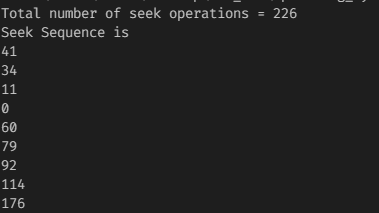
        92, 11, 41, 114 ]

head = 50

direction = "left"

SCAN(arr, head, direction)

OUTPUT





LAB 12

Simulating LOOK Disk Scheduling algorithm

Submitted by: Submitted to:

Nabin Katwal Santosh Dhungana

Class of 2022 Operating System

Roll No: 824

PROGRAM CODE

size = 8

disk\_size = 200

def LOOK(arr, head, direction):

    seek\_count = 0

    distance = 0

    cur\_track = 0

    left = []

    right = []

    seek\_sequence = []

    for i in range(size):

        if (arr[i] < head):

            left.append(arr[i])

        if (arr[i] > head):

            right.append(arr[i])

    left.sort()

    right.sort()

    run = 2

    while (run):

        if (direction == "left"):

            for i in range(len(left) - 1, -1, -1):

                cur\_track = left[i]

                seek\_sequence.append(cur\_track)

                distance = abs(cur\_track - head)

                seek\_count += distance

                head = cur\_track

            direction = "right"

        elif (direction == "right"):

            for i in range(len(right)):

                cur\_track = right[i]

                seek\_sequence.append(cur\_track)

                distance = abs(cur\_track - head)

                seek\_count += distance

                head = cur\_track

            direction = "left"

        run -= 1

    print("Total number of seek operations =",

        seek\_count)

    print("Seek Sequence is")

    for i in range(len(seek\_sequence)):

        print(seek\_sequence[i])

arr = [ 176, 79, 34, 60, 92, 11, 41, 114 ]

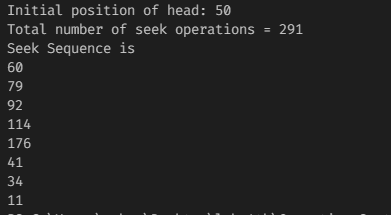
head = 50

direction = "right"

print("Initial position of head:", head)

LOOK(arr, head, direction)

OUTPUT





LAB 12

Simulating SSTF Disk Scheduling algorithm

Submitted by: Submitted to:

Nabin Katwal Santosh Dhungana

Class of 2022 Operating System

Roll No: 824

PROGRAM CODE

def calculateDifference(queue, head, diff):

    for i in range(len(diff)):

        diff[i][0] = abs(queue[i] - head)

def findMin(diff):

    index = -1

    minimum = 999999999

    for i in range(len(diff)):

        if (not diff[i][1] and

                minimum > diff[i][0]):

            minimum = diff[i][0]

            index = i

    return index

def shortestSeekTimeFirst(request, head):

        if (len(request) == 0):

            return

        l = len(request)

        diff = [0] \* l

        for i in range(l):

            diff[i] = [0, 0]

        seek\_count = 0

        seek\_sequence = [0] \* (l + 1)

        for i in range(l):

            seek\_sequence[i] = head

            calculateDifference(request, head, diff)

            index = findMin(diff)

            diff[index][1] = True

            seek\_count += diff[index][0]

            head = request[index]

        seek\_sequence[len(seek\_sequence) - 1] = head

        print("Total number of seek operations =",

                                    seek\_count)

        print("Seek Sequence is")

        for i in range(l + 1):

            print(seek\_sequence[i])

if \_\_name\_\_ =="\_\_main\_\_":

    proc = [176, 79, 34, 60,

            92, 11, 41, 114]

    shortestSeekTimeFirst(proc, 50)

OUTPUT

