

|  |  |  |
| --- | --- | --- |
| **Operating Systems** | | |
| Lab Manual | | |
| **Department of Computer Science and Engineering**  **The NorthCap University, Gurugram** | | |
|  | |
|  | |
|  | |
|  | |
|  |

**Operating Systems Lab Manual**

**CSL 303**

**Dr. Priyanka Vasisth**

**Dr. Divya Sharma**

**Ms. Poonam Chaudhary**



Department of Computer Science and Engineering

NorthCap University, Gurugram- 122001, India

Session 2019-20

*Published by:*

**School of Engineering and Technology**

**Department of Computer Science & Engineering**

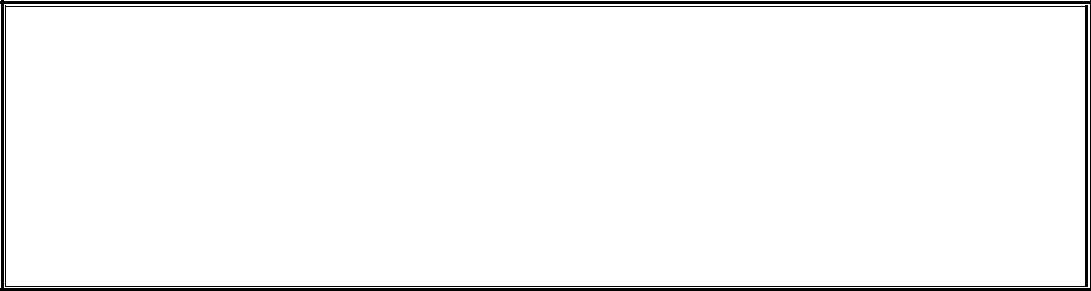
**The NorthCap University Gurugram**

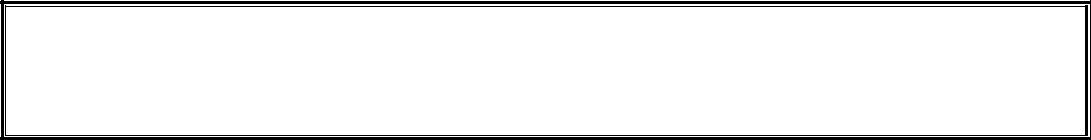
**• Laboratory Manual is for Internal Circulation only**

**© Copyright Reserved**

*No part of this Practical Record Book may be*

*reproduced, used, stored without prior permission of The NorthCap University*



Copying or facilitating copying of lab work comes under cheating and is considered as use of unfair means. Students indulging in copying or facilitating copying shall be awarded zero marks for that particular experiment. Frequent cases of copying may lead to disciplinary action. Attendance in lab classes is mandatory.

Labs are open up to 7 PM upon request. Students are encouraged to make full use of labs beyond normal lab hours.

**PREFACE**

Operating System Lab Manual is designed to meet the course and program requirements of NCU curriculum for B.Tech III year students of CSE branch. The concept of the lab work is to give brief practical experience for basic lab skills to students. It provides the space and scope for self-study so that students can come up with new and creative ideas.

The Lab manual is written on the basis of “teach yourself pattern” and expected that students who come with proper preparation should be able to perform the experiments without any difficulty. Brief introduction to each experiment with information about self-study material is provided. The laboratory exercises will include familiarization with LINUX system calls for process management and inter-process communication; Experiments on process scheduling and other operating system tasks through simulation/implementation. Students would require design process synchronization, CPU scheduling algorithms, memory management and disc management algorithms in high level languages like c, c++, python. Finally, the students would require applying the operating system concepts by experimenting on either xv6/minix operating systems. At the start of each experiment a question bank for preparation and practice is suggested which may be used to test the basic understanding of the students about the experiment. Students are expected to come thoroughly prepared for the lab. General disciplines, safety guidelines and report writing are also discussed.

The lab manual is a part of curriculum for the TheNorthCap University, Gurugram. Teacher’s copy of the experimental results and answer for the questions are available as sample guidelines.

We hope that lab manual would be useful to students of CSE, IT, ECE and BSc branches and author requests the readers to kindly forward their suggestions / constructive criticism for further improvement of the work book.

Author expresses deep gratitude to Members, Governing Body-NCU for encouragement and motivation.

**Authors**

**The NorthCap University**

**Gurugram, India**

|  |  |  |  |
| --- | --- | --- | --- |
| **CONTENTS** | | | |
| S.N. | **Details** | Page No. | |
|  | **Syllabus** |  | |
| 1 | **Introduction** |  | |
| 2 | **Lab Requirement** |  |
| 3 | **General Instructions** |  |
| 4 | **List of Experiments** |  |
| 5 | **List of Flip Assignment** |  |
| 6 | **List of Projects** |  |
| 7 | **Rubrics** |  |
| 8 | **Annexure 1 (Format of Lab Report)** |  |
| 9 | **Annexure 2 (Format of Lab Certificate)** |  |

**SYLLABUS**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1. **Department:** | | **Department of CSE** | | |
| 1. **Course Name: Operating Systems** | | 1. **Course Code :** | 1. **L- P** | 1. **Credits** |
| **Code: CSL 303** | 3-2 | 4 |
| 1. **Type of Course (Check one):** | | Programme Core Programme Elective Open Elective  **✓** | | |
| **✓**   1. **Frequency of offering (check one):** Odd Even Either Sem. Every Sem. | | | | |
| 1. **Brief Syllabus:** This is an introductory course which briefs LINUX Operating System Concepts that forms an integral part of computer science engineering in development of software applications in many diverse areas, including Web Development, Windows Applications, Research, Analytics and Processing. It lays the foundation of Process Management & Scheduling, Memory Management, Deadlocks and other Operating system Concepts. | | | | |
| 1. **Total lecture and Practical Hours for this course: 30 Hours**   The class size is maximum 30 learners. | | | | |
| 1. **Course Outcomes (COs)**   Possible usefulness of this course after its completion i.e. how this course will be practically useful to him once it is completed | | | | |
| **CO 1** | The students will be able to understand the basic architecture of Linux. | | | |
| **CO 2** | The students will be able to understand the process management & scheduling of Linux. | | | |
| **CO 3** | The students will be able to understand the memory management of Linux. | | | |
| **CO 4** | The students will be able to understand the inter process communication of Linux. | | | |
| **CO 5** | They will understand the main principles and techniques to handle the deadlocks. | | | |
| **CO6** | They will understand the I/O device management & the VFS of Linux. | | | |
| 1. **UNIT WISE DETAILS No. of Units: -05** | | | | |
| **Unit 1: Introduction to Linux OS Hours: 6**  Introduction & overview: functions of operating systems, Overview of various Operating Systems, Linux architecture, Boot strap loader of Linux, Tasks of the kernel, implementation strategies of kernel, System Calls. | | | | |
| **Unit II: Process Management & Scheduling Hours: 6**  Process priorities, process life cycle of Linux, process representation: process types, process identification numbers, process management system calls, kernel thread, overview of different scheduling algorithms, Linux scheduler: priority and completely fair share scheduling algorithm. | | | | |
| **Unit III: Process Synchronization and Memory Management Hours: 8**  Implementation of Producer- Consumer problem, implementation of semaphores, Page-Replacement Algorithms. | | | | |
| **Unit IV: Deadlocks Hours: 6**  Implementation of Banker’s Algorithm, | | | | |
| **Unit V: Virtual File System Hours: 4**  Disk scheduling algorithms, Introduction to VFS File System types, Common File model, Structure of the VFS | | | | |
| 1. **Guided Project (No. of Hours):** Case Study on Windows OS 2. **Unguided Project (No. of Hours):** Case Study ofLinux, Window, MAC OS | | | | |
| 1. **Brief Description of Self-learning component by students (through books/resource material etc.): Topics:** Linux syntax for shell scripting, revise c/c++/Python and data structure concepts from previous semesters | | | | |
| 1. **Suggested Readings**   GNU/Linux Command−Line Tools Summary [eBook]  <http://www.tldp.org/LDP/Bash-Beginners-Guide/Bash-Beginners-Guide.pdf>  **websites:**   * <https://www.linuxjournal.com/> * <https://www.omgubuntu.co.uk/> | | | | |

1. **INTRODUCTION**

That ‘learning is a continuous process’ cannot be over emphasized. The theoretical knowledge gained during lecture sessions need to be strengthened through practical experimentation. Thus practical makes an integral part of a learning process.

The purpose of conducting experiments can be stated as follows:

* To familiarize the students with the basic concepts, programming skill development and the take home laboratory assignments mainly implementation-oriented which have to be coded in high level language. The lab sessions will be based on exploring the concepts discussed in class.
* Observing basic structure and characteristics of Operating Systems
* Reporting and analyzing the complexities.
* Hands on experience on the experimental setup and software tools

1. **LAB REQUIREMENTS**

|  |  |  |
| --- | --- | --- |
| **S.No.** | **Requirements** | **Details** |
| **1** | **Software Requirements** | Linux’s Shell, Python/c/c++ |
| **2** | **Operating System** | Linux Operating System |
| **3** | **Hardware Requirements** | Windows and Linux: Intel 64/32 or AMD Athlon 64/32, or AMD Opteron processor  2 GB RAM  80 GB hard disk space |
| **4** | **Required Bandwidth** | NA |

1. **GENERAL INSTRUCTIONS** 
   1. **General discipline in the lab**
   * Students must turn up in time and contact concerned faculty for the experiment they are supposed to perform.
   * Students will not be allowed to enter late in the lab.
   * Students will not leave the class till the period is over.
   * Students should come prepared for their experiment.
   * Experimental results should be entered in the lab report format and certified/signed by concerned faculty/ lab Instructor.
   * Students must get the connection of the hardware setup verified before switching on the power supply.
   * Students should maintain silence while performing the experiments. If any necessity arises for discussion amongst them, they should discuss with a very low pitch without disturbing the adjacent groups.
   * Violating the above code of conduct may attract disciplinary action.
   * Damaging lab equipment or removing any component from the lab may invite penalties and strict disciplinary action.
   1. **Attendance**

* Attendance in the lab class is compulsory.
* Students should not attend a different lab group/section other than the one assigned at the beginning of the session.
* On account of illness or some family problems, if a student misses his/her lab classes, he/she may be assigned a different group to make up the losses in consultation with the concerned faculty / lab instructor. Or he/she may work in the lab during spare/extra hours to complete the experiment. No attendance will be granted for such case**.**
  1. **Preparation and Performance**
* Students should come to the lab thoroughly prepared on the experiments they are assigned to perform on that day. Brief introduction to each experiment with information about self study reference is provided on LMS.
* Students must bring the lab report during each practical class with written records of the last experiments performed complete in all respect.
* Each student is required to write a complete report of the experiment he has performed and bring to lab class for evaluation in the next working lab. Sufficient space in work book is provided for independent writing of theory, observation, calculation and conclusion.
* Students should follow the Zero tolerance policy for copying / plagiarism. Zero marks will be awarded if found copied. If caught further, it will lead to disciplinary action.
* Refer **Annexure 1** for Lab Report Format

1. **LIST OF EXPERIMENTS**

|  |  |  |
| --- | --- | --- |
| Exp. No. | Division of Experiments | List of Experiments |
| 1 | Basics of Linux | Explain the structure of Linux Operating System |
| 2 | Installation of Ubuntu Operating system |
| 1 | Shell Programs | Write a shell program to find factorial of a number. |
| 2 | Write a shell program to find gross salary of an employee. |
| 3 | Write a shell program to display the menu and execute instructions accordingly  (i)List of file (ii)Process Status (iii) Date (iv) users in program (v) Quit |
| 4 | Write a shell program to find Fibonacci series. |
| 5 | Write a shell program to find largest of three numbers. |
| 6 | Write a shell program to find average of N numbers |
| 7 | CPU Scheduling Algorithms | Write a C program to simulate the following non-preemptive CPU scheduling algorithms to find turnaround time and waiting time.  a) FCFS b) SJF c) Round Robin (pre-emptive) d) Priority |
| 8 | \*Write a C program to simulate multi-level queue scheduling algorithm considering the following scenario. All the processes in the system are divided into two categories – system processes and user processes. System processes are to be given higher priority than user processes. Use FCFS scheduling for the processes in each queue. |
| Implement the following CPU scheduling Algorithms.  i) Round Robin  ii) Priority Based |
| 9 | Deadlock Management  Technique | Write a program to simulate Bankers algorithm for the purpose of deadlock avoidance |
| 10 | Page Replacement  Algorithms | Write a C program to simulate page replacement algorithms  a) FIFO b) LRU c) LFU |
| 11 | Write a C program to simulate page replacement algorithms  a) Optimal |

1. **LIST OF FLIP EXPERIMENTS**
2. Execute the **who** command written in a file to instruct the shell to read input from a file called "myfile1" instead of from the keyboard. Use the **more** command to see the contents of myfile1.
3. Use the date and who commands in sequence (in one line) such that the output of date will display on the screen and the output of who will be redirected to a file called myfile2. Use the more command to check the contents of myfile2
4. Write a sed command that swaps the first and second words in each line in a file.
5. Write a shell script that takes a command –line argument and reports on whether it is directory, a file, or something else.
6. Write a shell script that accepts one or more file name as arguments and converts all of them to uppercase, provided they exist in the current directory.
7. Write a shell script that determines the period for which a specified user is working on the system.
8. Write a shell script that accepts a file name, starting and ending line numbers as arguments and displays all the lines between the given line numbers.
9. Write a shell script that deletes all lines containing a specified word in one or more files supplied as arguments to it.
10. Write a shell script to perform the following string operations:
    * 1. To extract a sub-string from a given string
      2. To find the length of a given string
11. **LIST OF PROJECTS**
    * + 1. Case Study of Window OS
        2. Case Study of Linux OS
        3. Case Study of MAC OS
12. **RUBRICS**

|  |  |
| --- | --- |
| **Marks Distribution** | |
| **Continuous Evaluation(50 Marks)** | **End Semester Exam (20 Marks)** |
| Each experiment shall be evaluated for 10 marks and at the end of the semester proportional marks shall be awarded out of 50. | End semester practical evaluation including Mini project (if any) carries 20 marks. |
| Following is the breakup of 10 marks for each  **4 Marks**: Observation & conduct of experiment. Teacher may ask questions about experiment.  **3 Marks:** For report writing  **3 Marks:** For the 15 minutes quiz to be conducted in every lab. |

**Annexure 1**

**Operating Systems**

**(CSL 303)**

Lab Practical Report



Faculty name: Dr. Priyanka Vasisth Student name: Chayan Gulati

Roll No.: 18csu054

Semester: 5

Group: A2

Department of Computer Science and Engineering

NorthCap University, Gurugram- 122001, India

Session 2019-20

**INDEX**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **S.No** | **Experiment** |  | **Date of Experiment** | **Date of Submission** | **Marks** | **Covered** | **Signature** |
| **0** | **Installation of Cygwin,**  **Write basic Linux commands.** |  | **23/08/20,**  **30/08/20** |  |  |  |  |
| **1** | **Write a shell program to find factorial of a no.** |  | **7/09/20** |  |  |  |  |
| **2** | **Write a shell program to find gross salary of an employee.** |  | **7/09/20** |  |  |  |  |
| **3** | **Write shell program to display the menu and execute instructions.** |  | **7/09/20** |  |  |  |  |
| **4** | **Write a shell program to find Fibonacci series.** |  | **7/09/20** |  |  |  |  |
| **5** | **Write a shell program to find largest of three no.** |  | **7/09/20** |  |  |  |  |
| **6** | **Write a shell program to find average of N no.** |  | **7/09/20** |  |  |  |  |
| **7** | **Implement FCFS (with/without arrival time), Implement SJF (with/without arrival time)** |  | **12/09/20,**  **19/09/20** |  |  |  |  |
| **8** | **Implement Priority scheduling,**  **Implement Round robin (with/without arrival time)** |  | **26/09/20.**  **1/10/20** |  |  |  |  |
| **--** | **Case study on different Windows versions** |  | **10/8/20** |  |  |  |  |
| **9** | **Write a program to simulate Reader Writer algorithm** |  | **28/10/20** |  |  |  |  |
| **10** | **Write a program to simulate Bankers algorithm for the purpose of deadlock avoidance** |  | **11/11/20** |  |  |  |  |

**Experiment No: 0**

Student Name and Roll Number: Chayan 18csu054

Semester /Section: 5/A

Link to Code: https://github.com/chayangulati321/Operating-System

Date: 23/08/20

Faculty Signature:

Remarks:

**Objective**

To familiarize the students to Linux interface.

**Program Outcome**

* The students will understand commands used in Linux.

**Problem Statement**

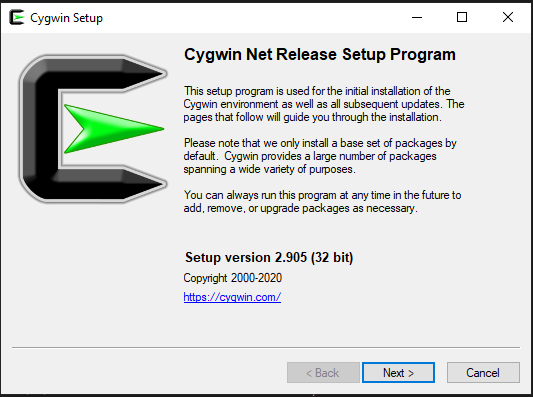
Implement the following things:

* Cygwin Installation
* Basic Linux commands

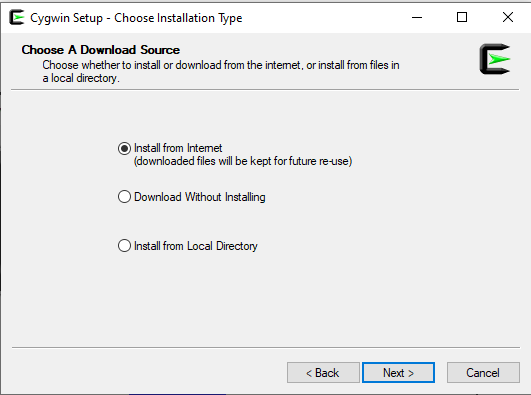
**CYGWIN INSTALLATION**

**Step 1**: Download the Cygwin setup from <https://www.cygwin.com/>.

After downloading double click on setup and Click next.

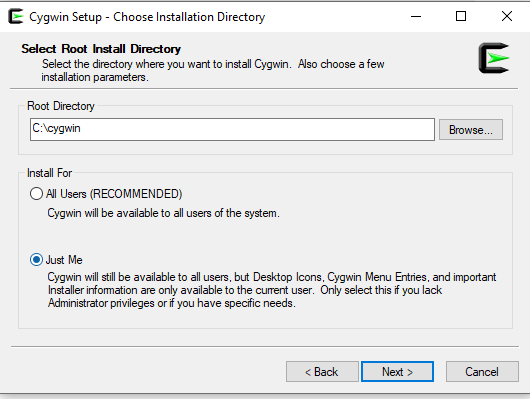


**Step 2**: Choose first option (install from internet) and click next to continue.

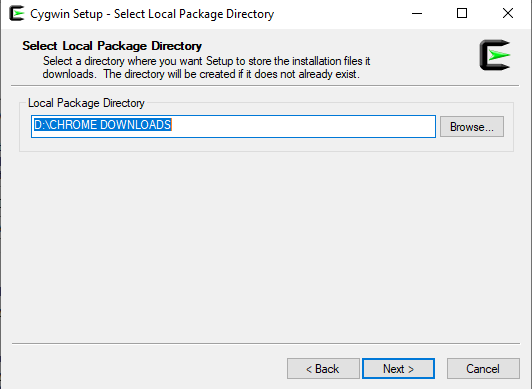


**Step 3**: Select the root directory. Change it according to need or keep it as default.

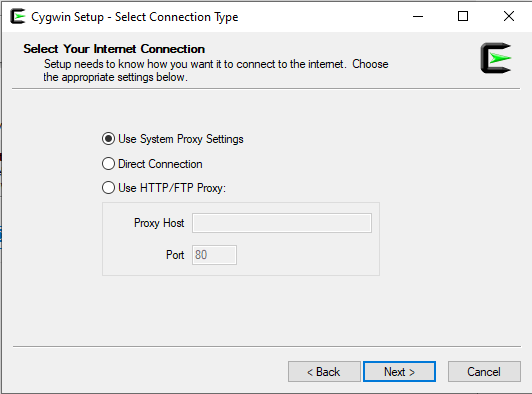
In install for, select just me and hit next.



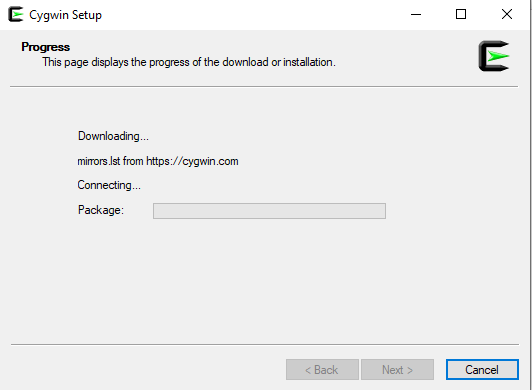
**Step 4:** Browse to change the path of a directory for local packages. After selecting appropriate directory hit next.



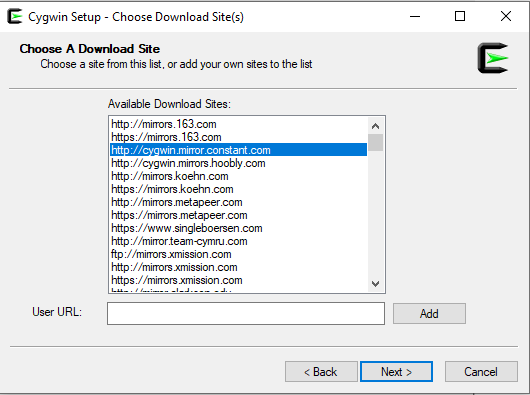
**Step 5**: Select connection type according to need or keeping it as default.



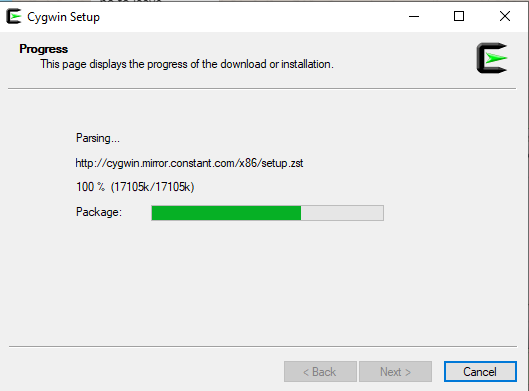
**Step 6**: After selecting connection type, Cygwin will automatically start connecting to the internet.

.

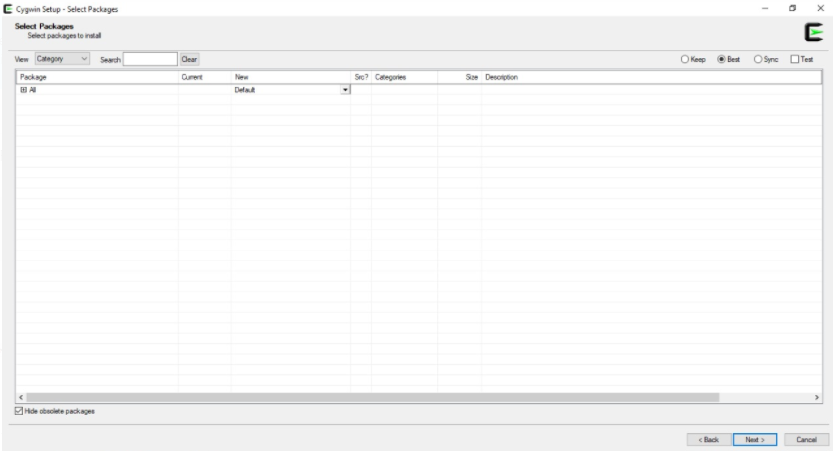
**Step 7**: Select site for downloading the packages. Choose the third site as in image and click next to continue.



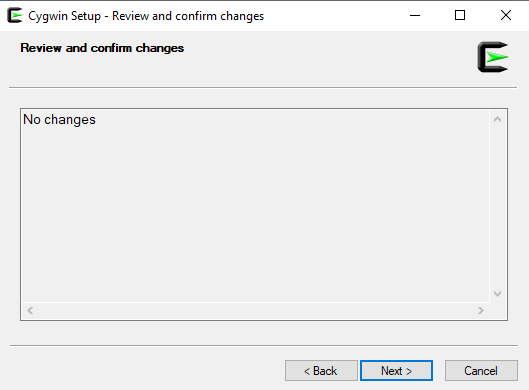
**Step 8**: Cygwin will start downloading the default packages from the site.



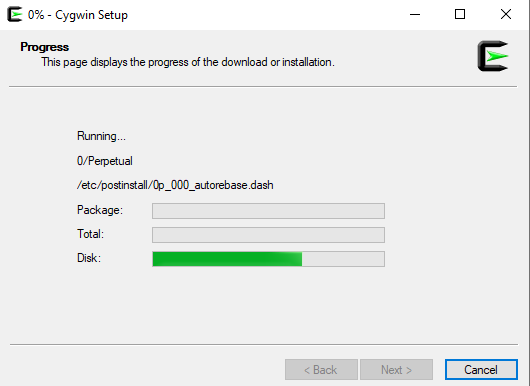
**Step 9**: A screen will pop up. Add additional packages according to the need or make no changes and click on next.



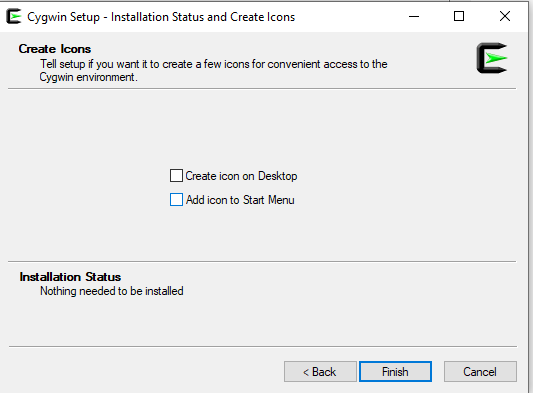
**Step 10**: If selected some additional packages, the following packages will show up else it will show no changes.



**Step 11**: Installation will begin, it will take around 2 minutes. Progress is shown at the top bar where it is showing 0% - Cygwin setup



**Step 12**: To create an icon on Desktop or Add icon to Start Menu, click to select the required option. At last click finish and installation is done.



OS LINUX COMMANDS FILE

Mkdir and Cat Commands

Mkdir name : makes a folder of the name in a speciifed area.

Cat > filename : this cat commands makes new file of the specified name

Cat Filename : This cat command shows the contents of a following file name

Cat filename1 > Filename2 : This cat command copies the content of filename 1 into filename 2

Cat filename1 >> Filename2 : This cat command appends the content of filename 1 into filename 2

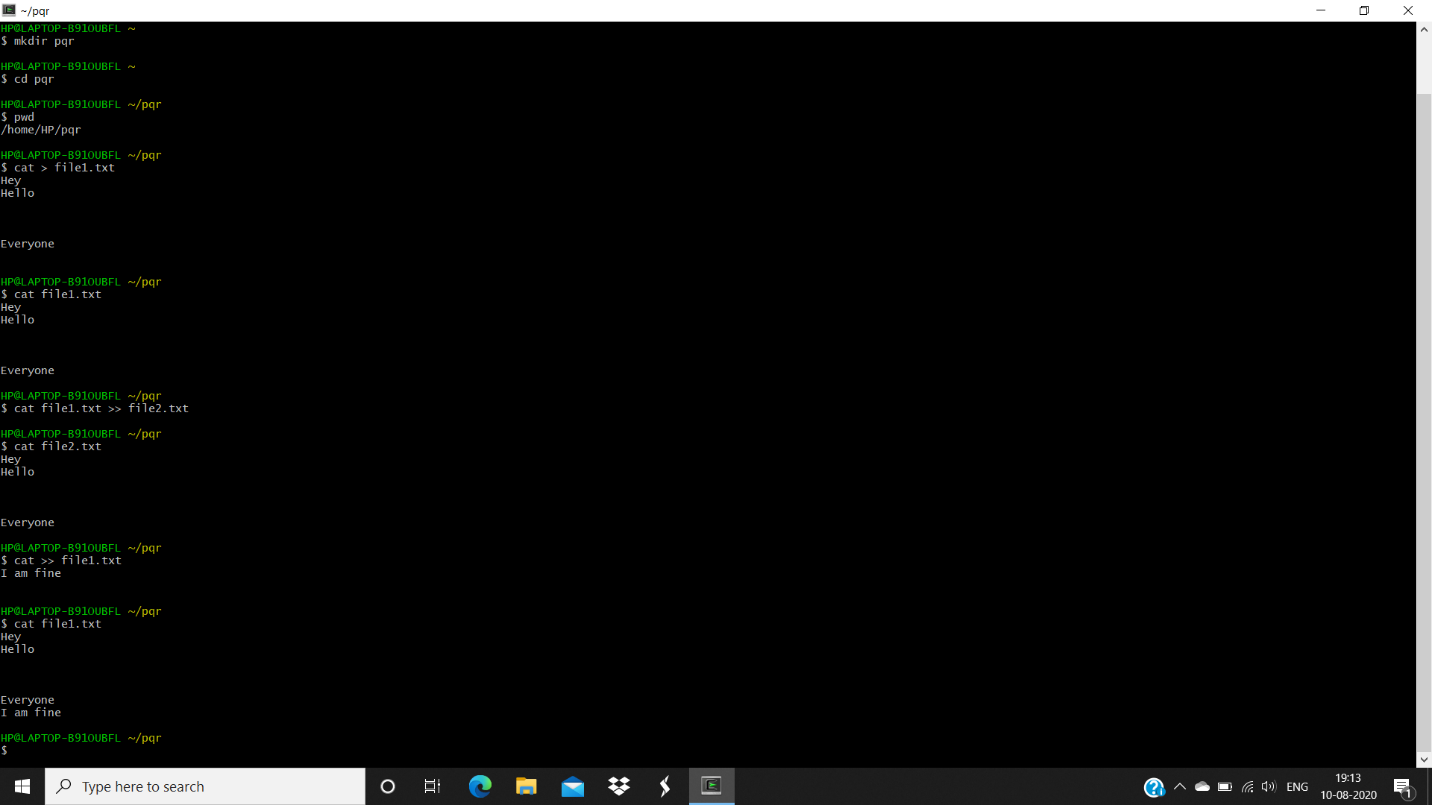
Cat Filename >> : This cat command helps you to edit a file in the cmd

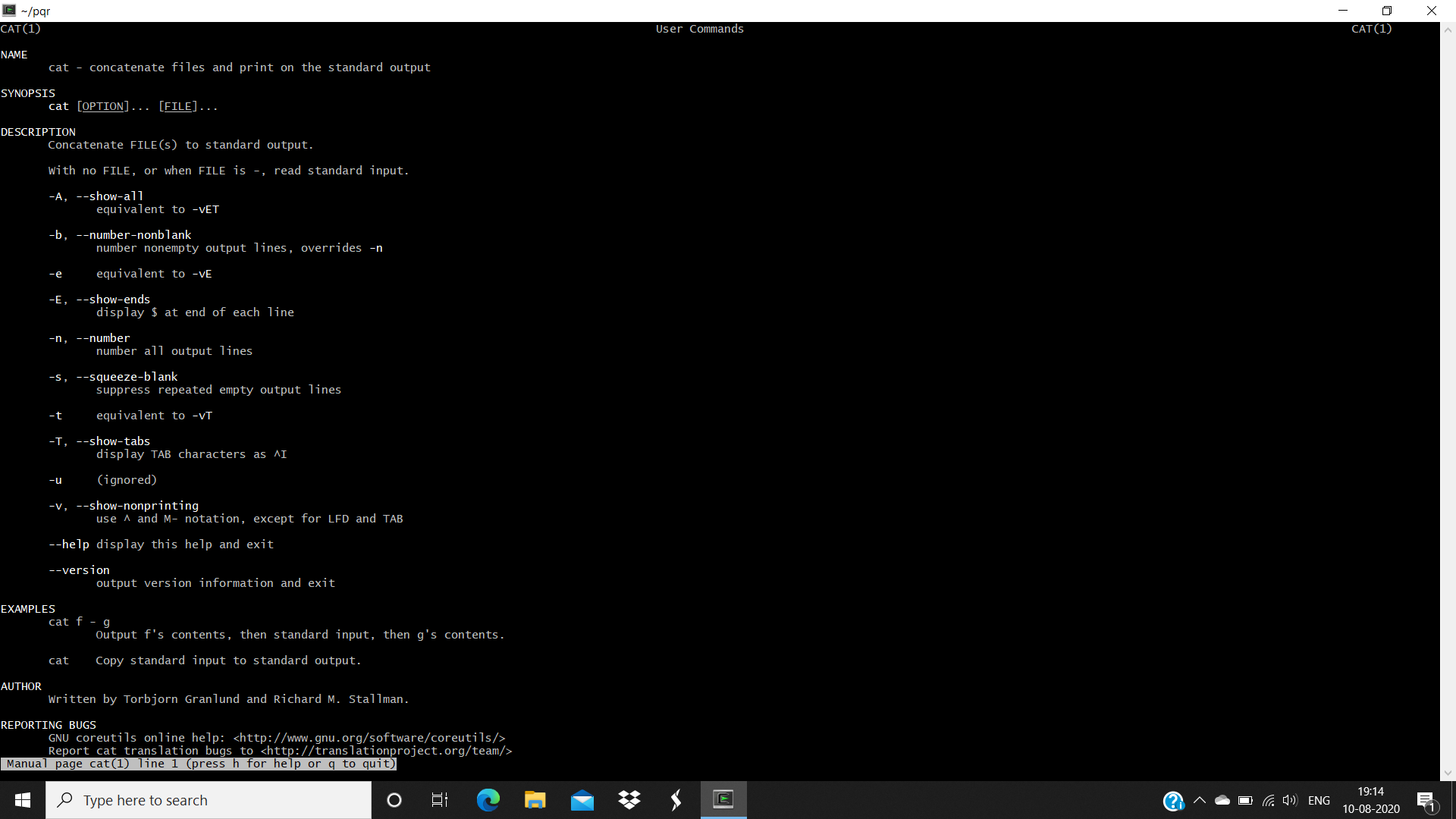
Cat -n filename : This cat command shows the content of the file with amount of contents in the each row

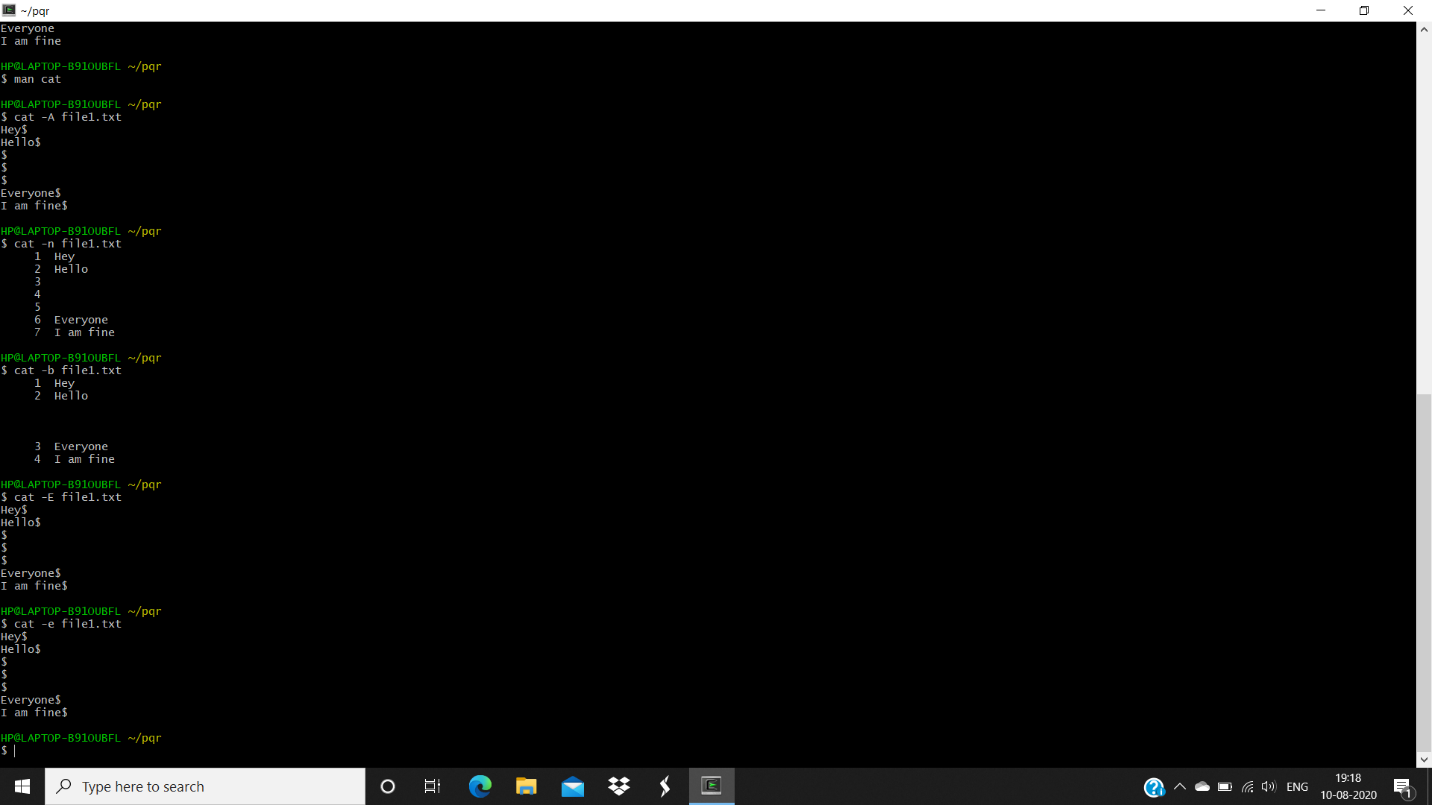
Cat -e filename: This cat command applies dollars to every end of line.

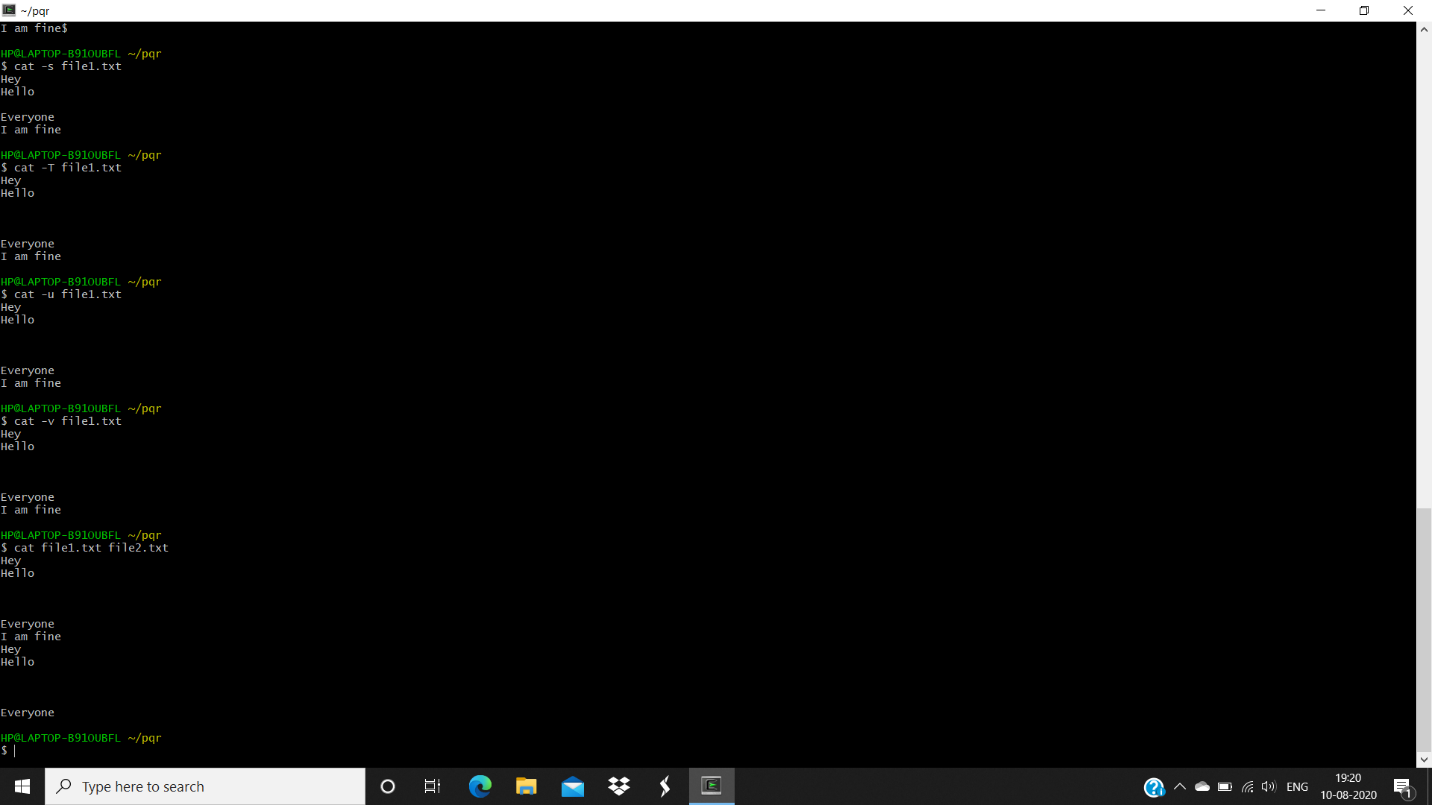
Cat -T filename : Every tab space is covered with ‘**^I’ sign**

**Cat -b filename : removes all the blank Lines**

****

****

****

****

LS COMMANDS

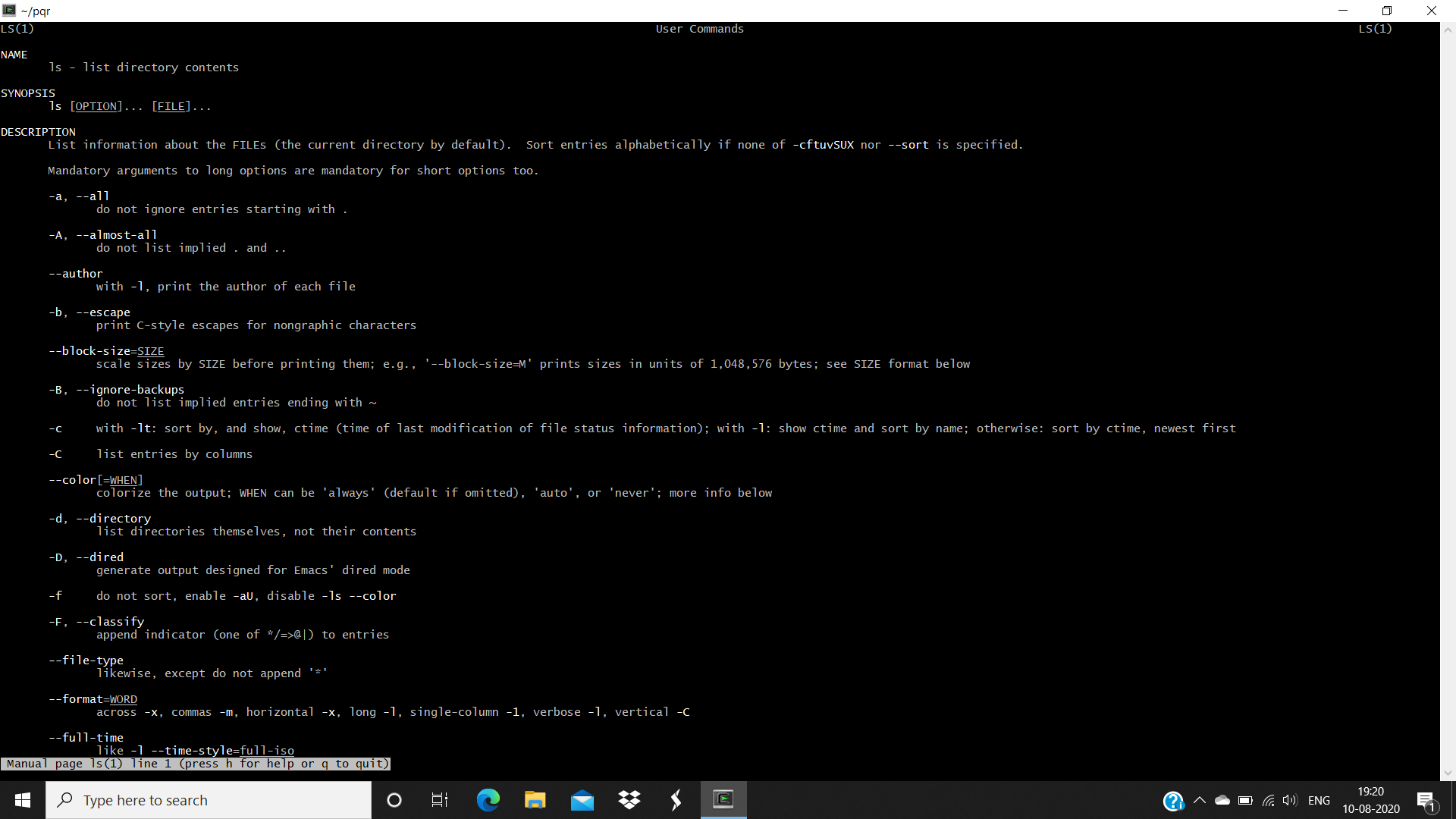
Ls -a: list all the files including the hiddens ones too

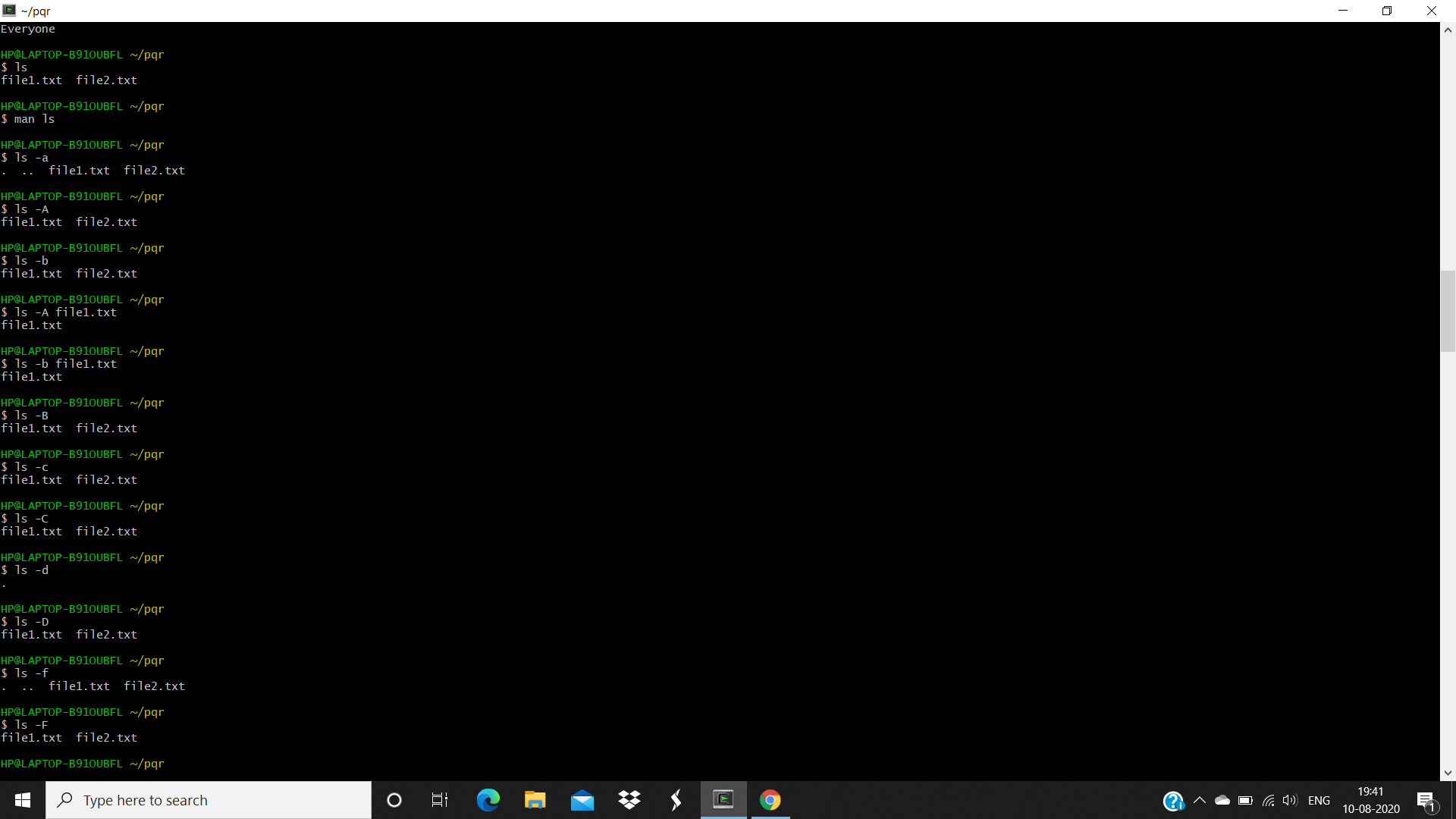
Ls -l : it gives us the information about our different files. Such as telling us how much permissions one file has.

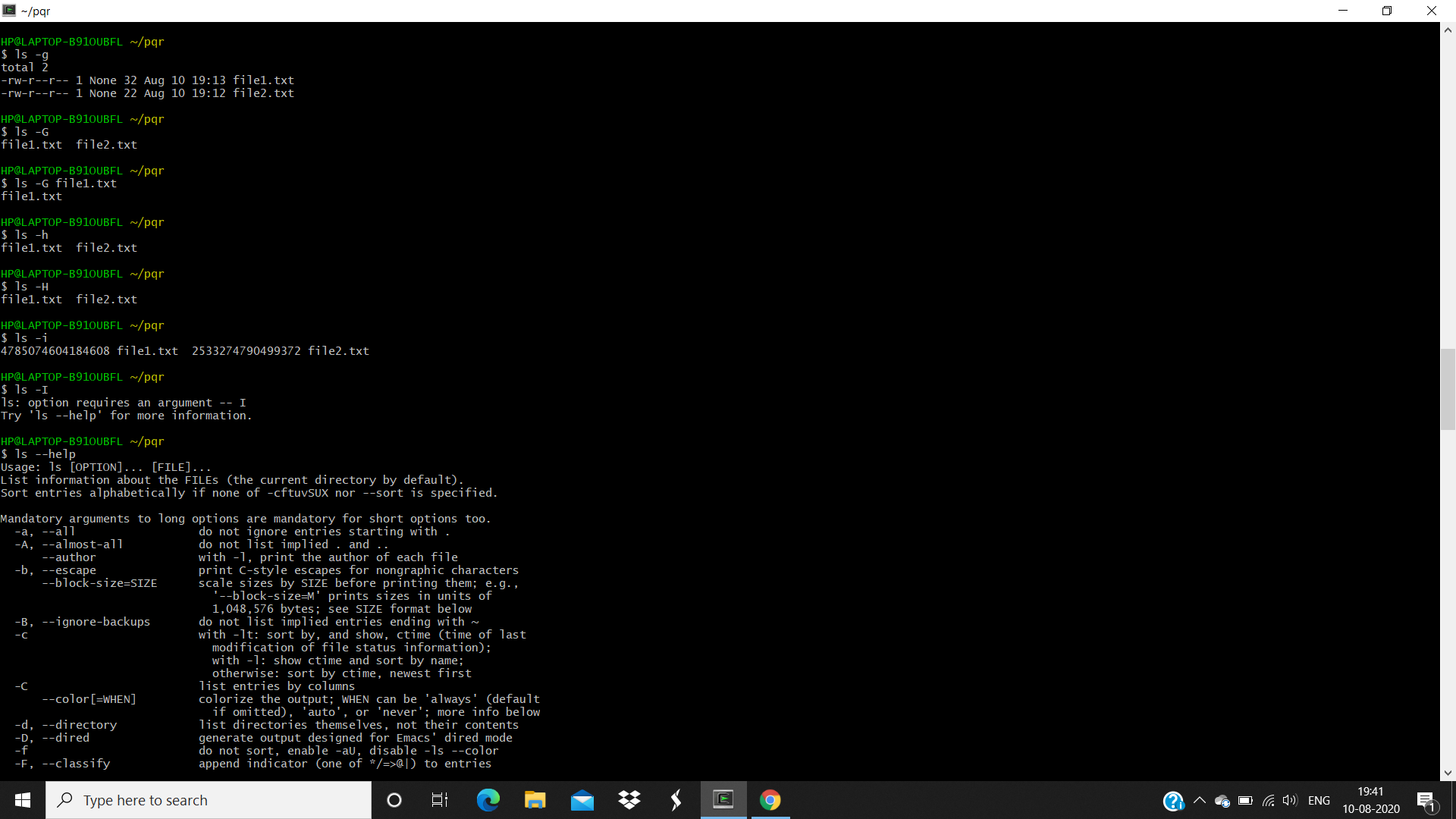
Ls -t : files are listed according to the modification date. The latest modified files are showed first

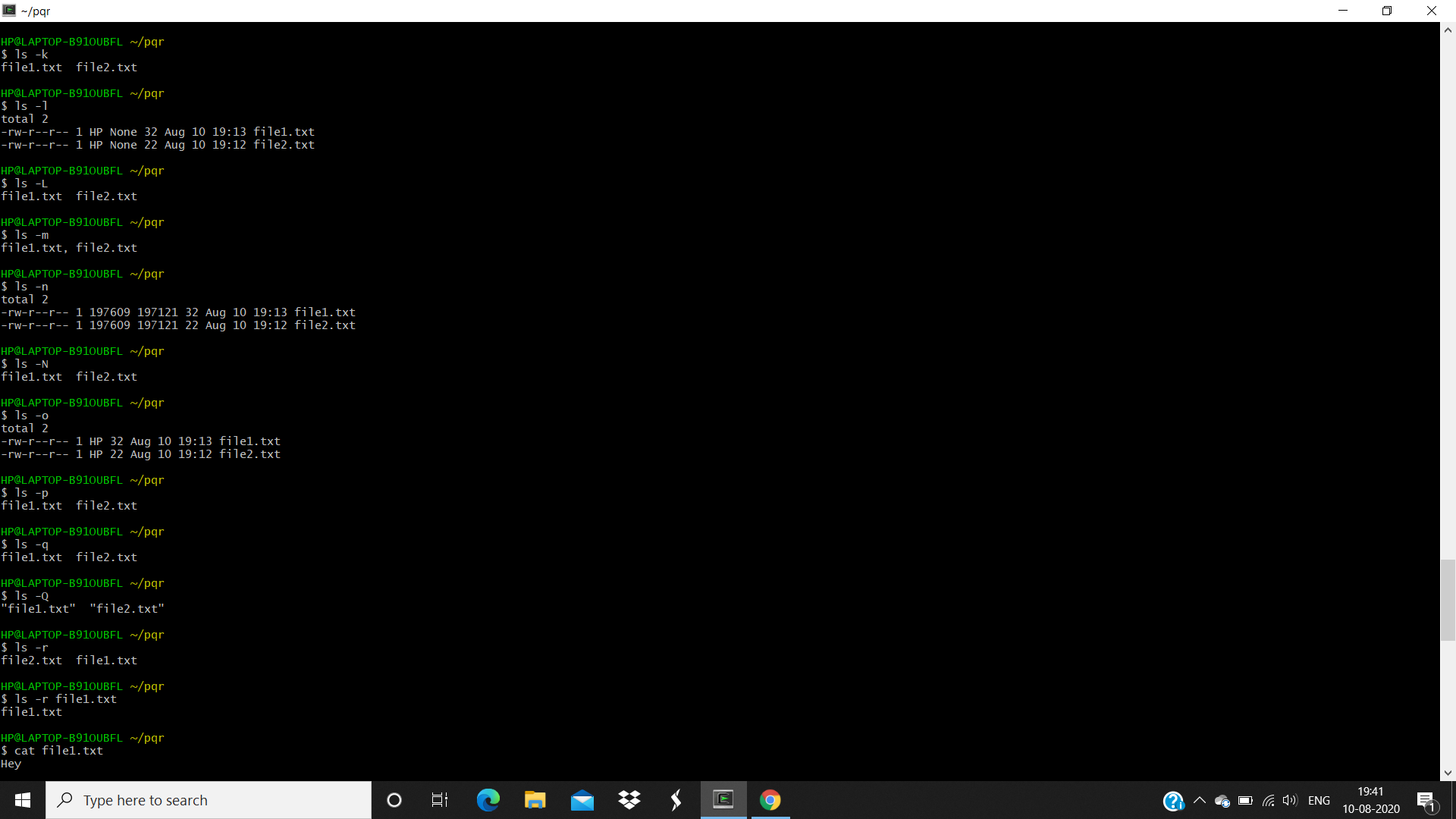
Ls -r : Files are showed in reverse order

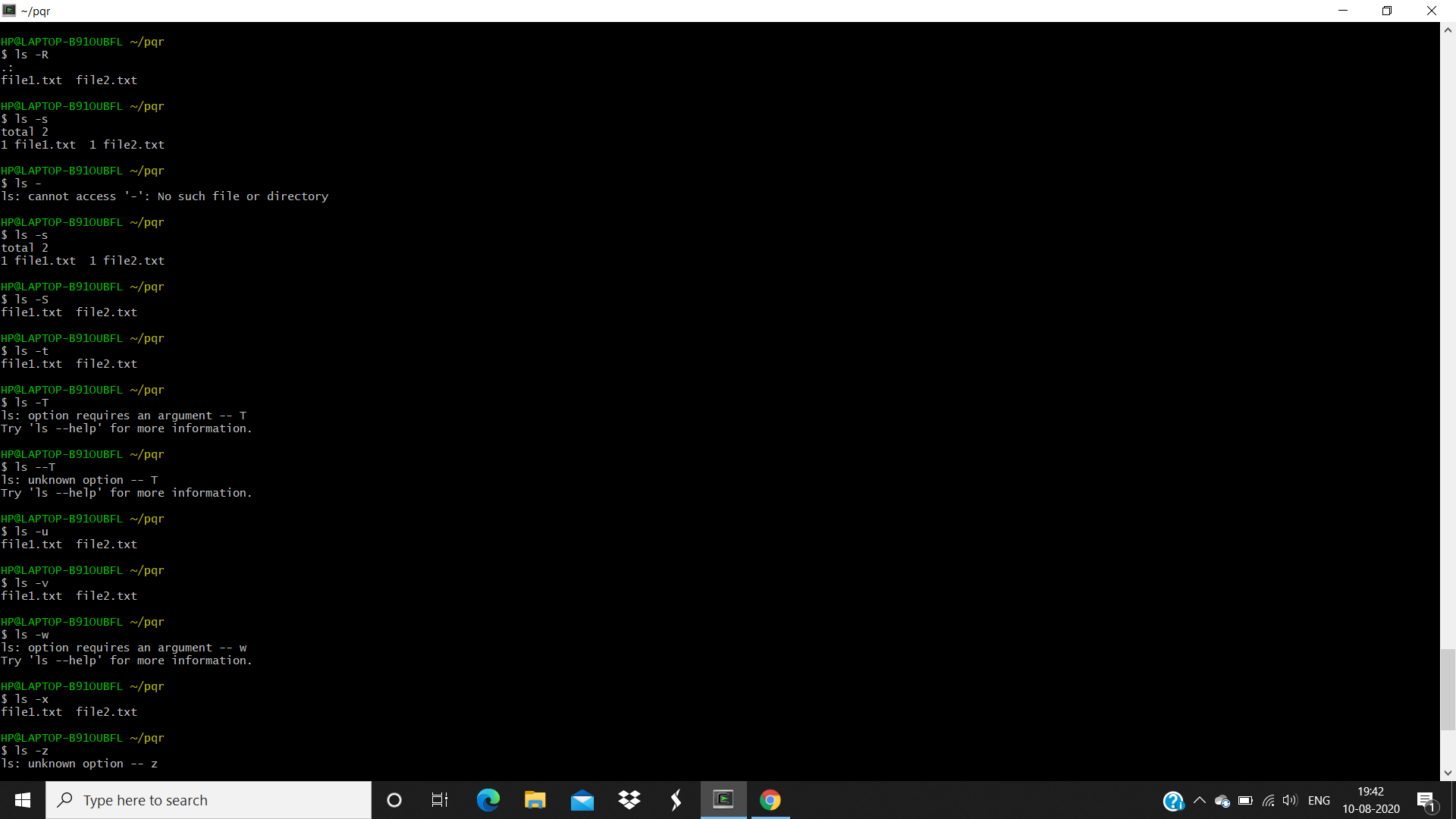
Ls -I : prints the index number of every file present in the directory

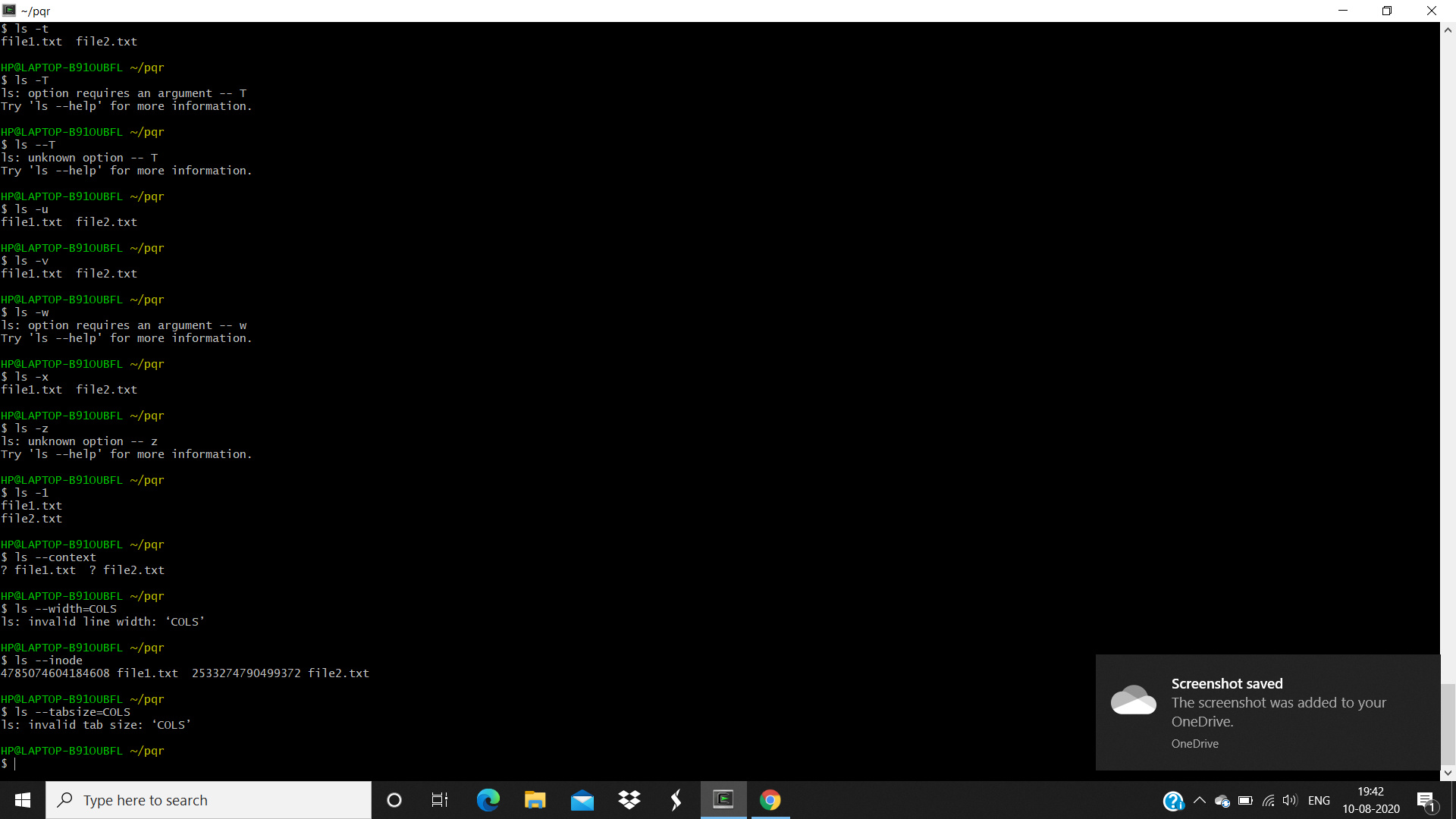
****

****

****

****

****

****

Alias Command

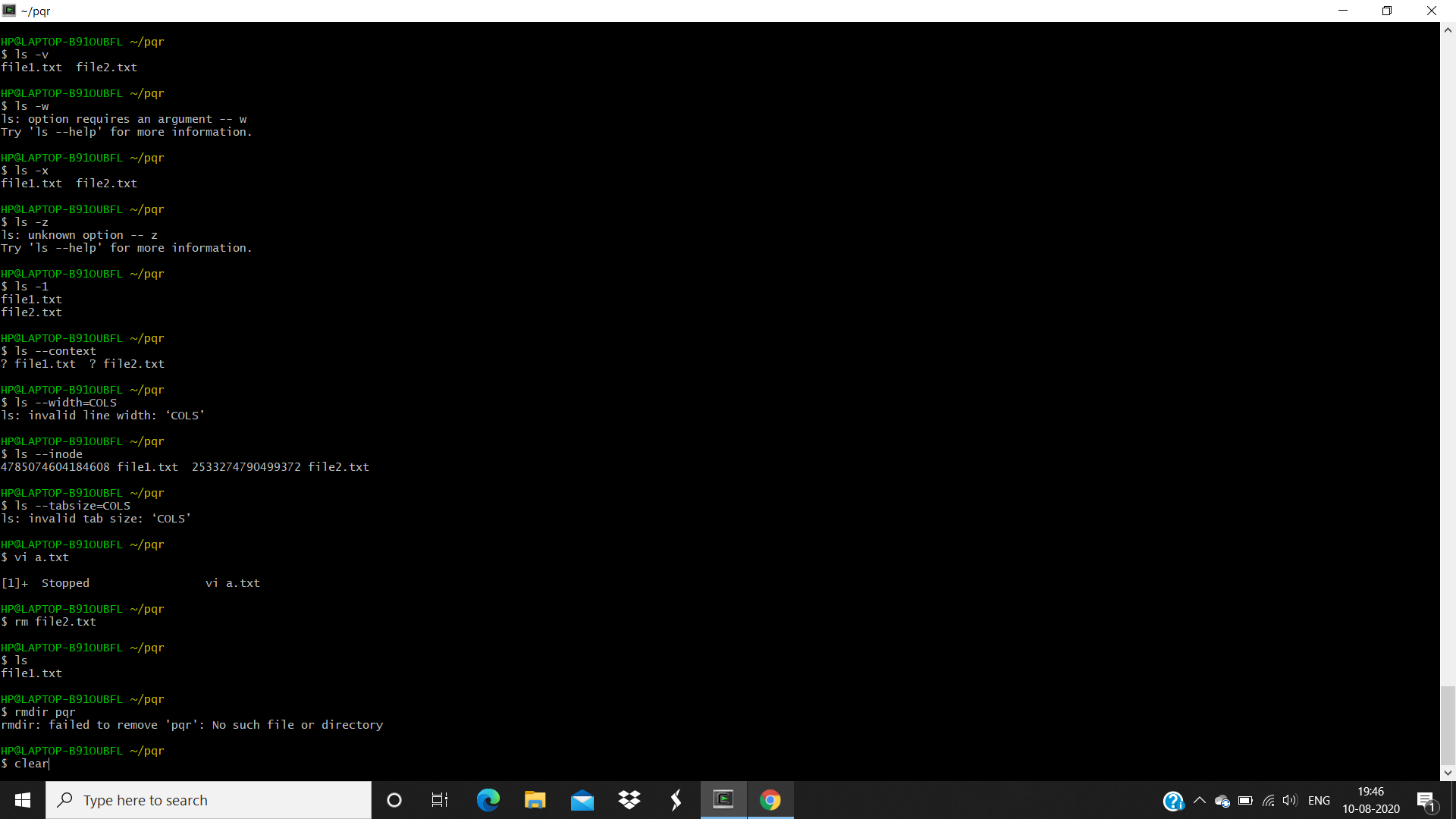
Alias newcommand name = “Old name” : To rename any command to a name you want

Remove/Delete Command

Use clear command to clear the sheet of cygwin terminal.

rm filename if we want to delete the file.

rm dir pqr to delete the directory.

****

Chmod Commands:

Chmod u+wx/u+rwx : This command gives the read or write or view access to a user for a particular file.

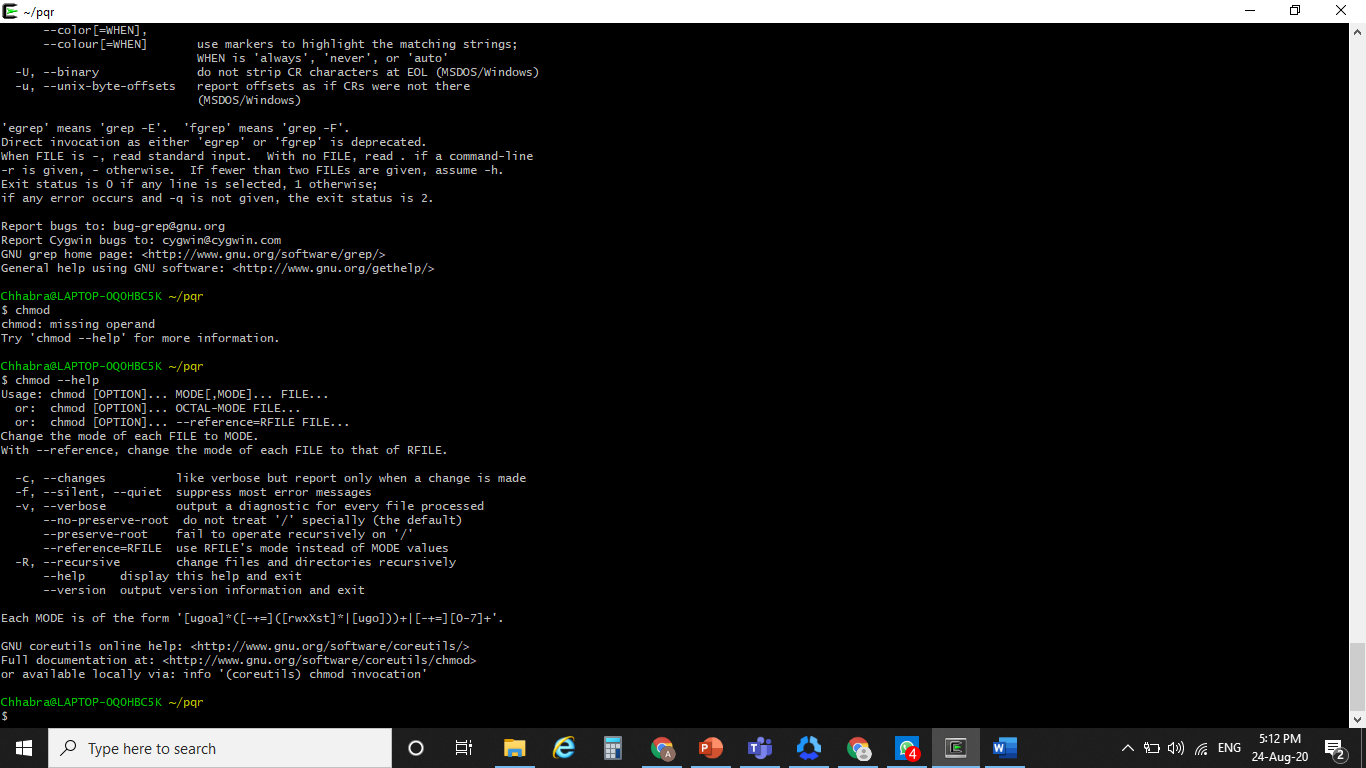
Chmod o+wx/o+rwx : This command gives the read or write or view access to others for a particular file

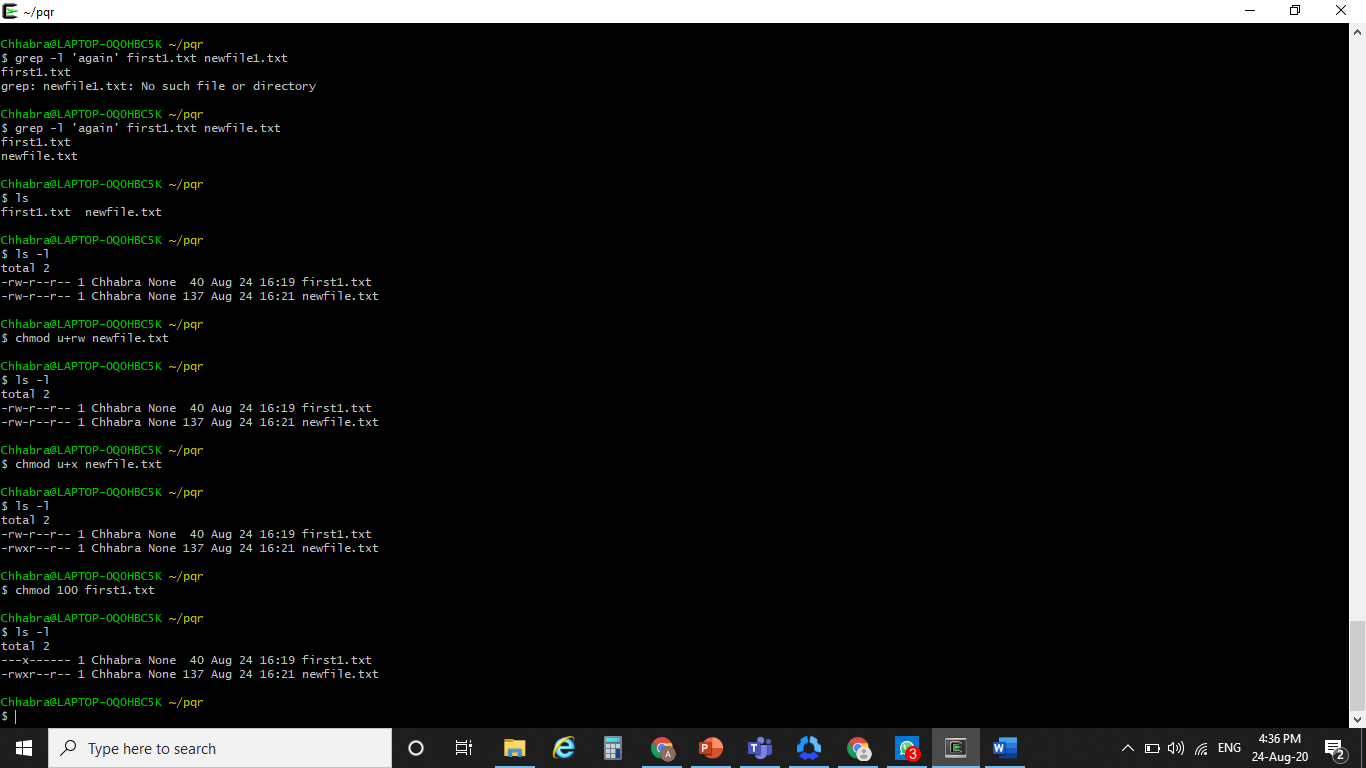
Chmod u-wx/u-rwx : This command removes the read or write or view access from a user for a particular file

Chmod o-wx/o-rwx : This command removes the read or write or view acces from others for a particular file.

Chmod 100 filename : gives only read permission to a file to all the users, root and others.

Chmod 777 filename: Gives all the permission to a file to all the users, root and others.





Grep Commands:

Grep “particular name” filename : This command is used to search a particular word or line in a file.

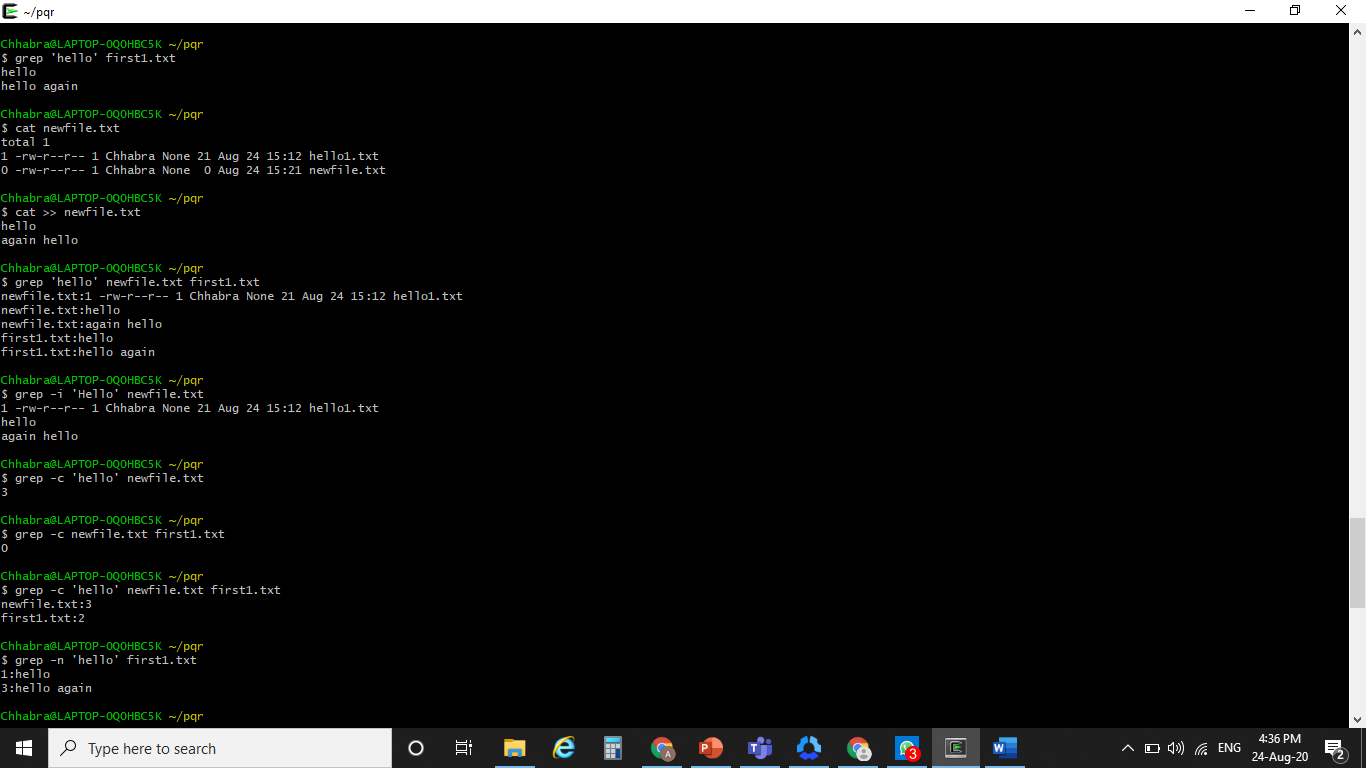
Grep “particular name” filename1 filename2 : This command is used to search a particular word or line in multiple lines.

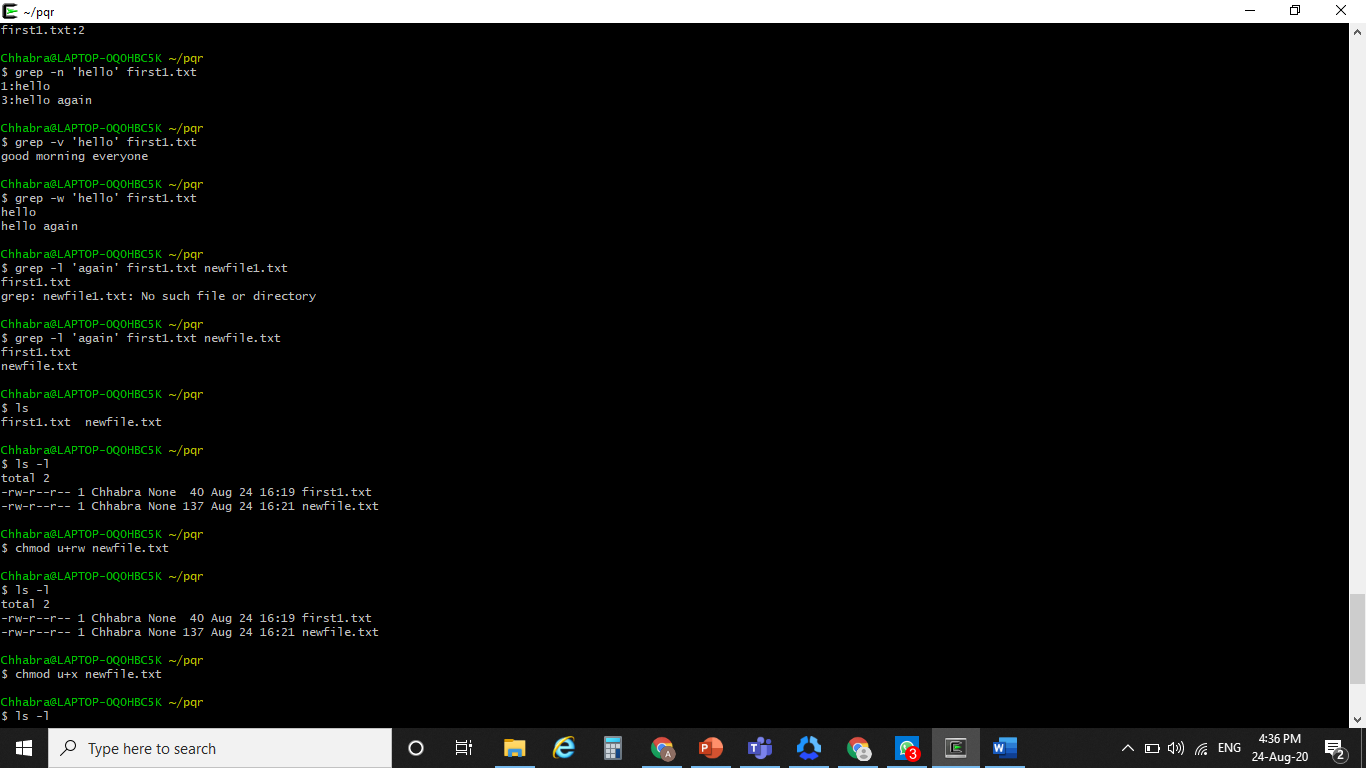
Grep -i “particular name” filename : This command is used to search a particular word or line in a file.

Grep -c “particular name” filename : This command is tells the count of a particular word or line in a file.

Grep -n “particular name” filename : This command is tells at which line number a particular word or line is present that file.

Grep -v “particular name” filename : This command prints all the other words present in that file except the one given n the commands.





**Experiment No: 1**

Student Name and Roll Number: Chayan 18csu054

Semester /Section: 5/A

Link to Code: https://github.com/chayangulati321/Operating-System

Date: 7/09/20

Faculty Signature:

Remarks:

**Objective**

Write a shell program to find factorial of a number.

**Program Outcome**

* The students will understand the shell program to find factorial of a number.

**Problem Statement**

Write a shell program to find factorial of a number.

**Background Study:**

A factorial is a function that multiplies a number by every number below it. For example 5!= 5\*4\*3\*2\*1=120. The function is used, among other things, to find the number of way “n” objects can be arranged.

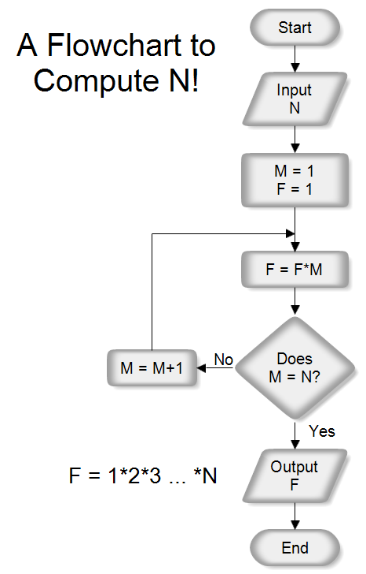
**Algorithm/ flowchart**

1. Get a number

2. Use for loop or while loop to compute the factorial by using the below formula

3. fact(n) = n \* n-1 \* n-2 \* …. 1

4. Display the result.



**Code**

echo "Enter a no.: "

read n

f=1

for ((i=2;i<=n;i++))

{

F=$((f\*i))

}

echo $f

**Output: Screenshots**



**Experiment No: 2**

Student Name and Roll Number: Chayan 18csu054

Semester /Section: 5/A

Link to Code: https://github.com/chayangulati321/Operating-System

Date: 7/09/20

Faculty Signature:

Remarks:

**Objective**

Write a shell program to find gross salary of an employee.

**Program Outcome**

The students will understand Write a shell program to find gross salary of an employee.

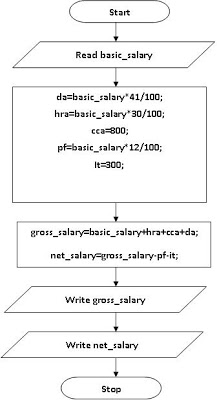
**Problem Statement**

Write a shell program to find gross salary of an employee.

**Background Study:**

To find the gross salary we have to take all the value of allowances and other things that we have to reduce or to increment the things accordingly.

**Algorithm/ flowchart**



**Code**

echo "Enter the basic salary : "

read s

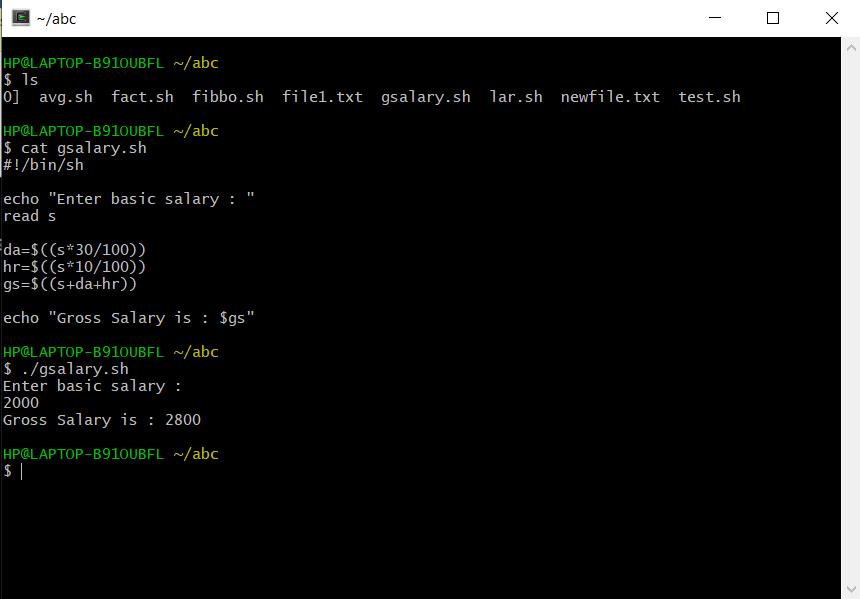
da = $((s\*30/100))

hr = $((s\*10/100))

gs = $((s+da+hr))

echo “Gross Salary is : $gs”

**Output: Screenshots**



**Experiment No: 3**

Student Name and Roll Number: Chayan 18csu054

Semester /Section: 5/A

Link to Code: https://github.com/chayangulati321/Operating-System

Date: 7/09/20

Faculty Signature:

Remarks:

**Objective**

Write a shell program to display the menu and execute instructions accordingly

(i)List of file (ii)Process Status (iii) Date (iv) users in program (v) Quit

**Program Outcome**

The students will understand Write a shell program to display the menu and execute instructions accordingly

(i)List of file (ii)Process Status (iii) Date (iv) users in program (v) Quit

**Problem Statement**

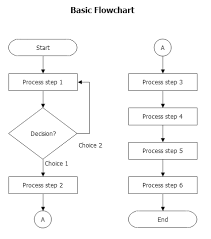
Write a shell program to display the menu and execute instructions accordingly

(i)List of file (ii)Process Status (iii) Date (iv) users in program (v) Quit

**Background Study:**

By this program we can execute the file list that we have and to see the process status, date for that and user in program

**Algorithm/ flowchart**

****

**Code**

Echo “ select option”

Echo “list of files”

Echo “ process status”

Echo “ date”

Echo “ users in program”

Echo “quit”

Read n

If [$n -eq 1]

Then

Ls

Elif [$n -eq 2]

Then

Ps -aux

Elif [ $n -eq 3]

Then

Date + “%m-%d-%y”

Elif [$n -eq 4]

Then

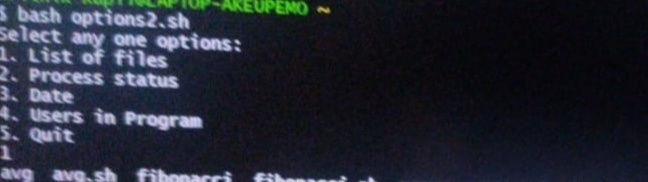
Users

Else

Echo “exiting..”

fi

**Output: Screenshots**



**Experiment No: 4**

Student Name and Roll Number: Chayan 18csu054

Semester /Section: 5/A

Link to Code: https://github.com/chayangulati321/Operating-System

Date: 7/09/20

Faculty Signature:

Remarks:

**Objective**

Write a shell program to find Fibonacci series.

**Program Outcome**

* The students will understand a shell program to find Fibonacci series.

**Problem Statement**

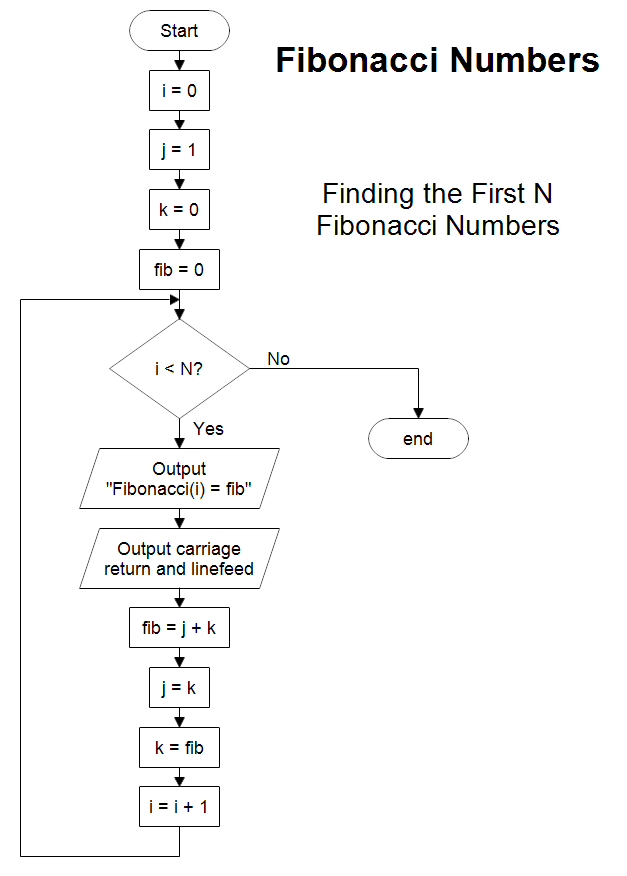
Write a shell program to find Fibonacci series.

**Background Study:**

The Fibonacci sequence is one of the most famous formulas in mathematics.

Each number in the sequence is the sum of the two numbers that precede it. So, the sequence goes: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, and so on. The mathematical equation describing it is *Xn+2= Xn+1 + Xn*

**Algorithm/ flowchart**



**Code**

echo "Enter a no.: "

read n

a=0

b=1

echo “The Fibonacci series is : ”

for ((i=0;i<n,i++))

{

echo “$a”

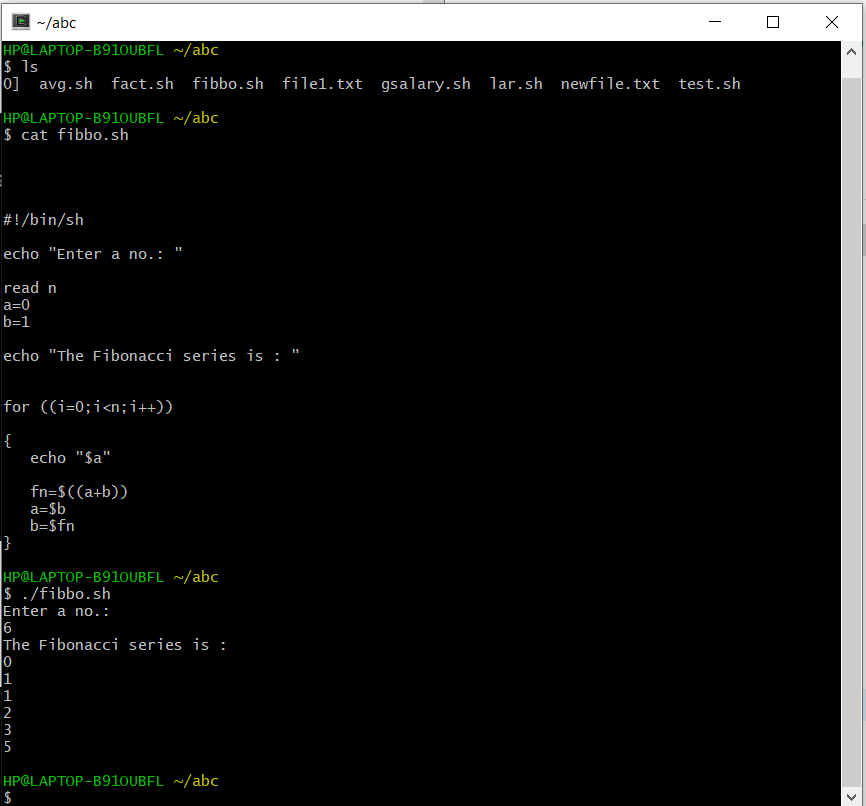
fn = $((a+b))

a=$b

b=$fn

}

**Output: Screenshots**



**Experiment No: 5**

Student Name and Roll Number: Chayan 18csu054

Semester /Section: 5/A

Link to Code: https://github.com/chayangulati321/Operating-System

Date: 7/09/20

Faculty Signature:

Remarks:

**Objective**

Write a shell program to find largest of three numbers.

**Program Outcome**

Write a shell program to find largest of three numbers.

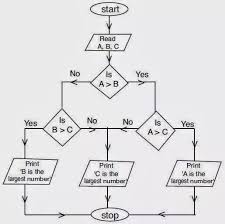
**Problem Statement**

Write a shell program to find largest of three numbers.

**Background Study**:

To find the largest of three number we can check if a particular number is greater then other else check for other numbers also if found the greatest then report for the answer .

**Algorithm/ flowchart**

****

1. Get three numbers. Say num1, num2, num2

2. If (num1 > num2) and (num1 > num3)

     echo value of num1

3. elif(num2 > num1) and (num2 > num3)

     echo value of num2

4. Otherwise,

     echo value of num3

**Code**

echo “Enter a: ”

read a

echo “Enter b: “

read b

echo “Enter c: ”

read c

if [ $a -gt $b ] && [ $a -gt $c ]

then

echo “Largest no. is : ,$a”

elif [ $b -gt $a ] && [ $b -gt $c ]

then

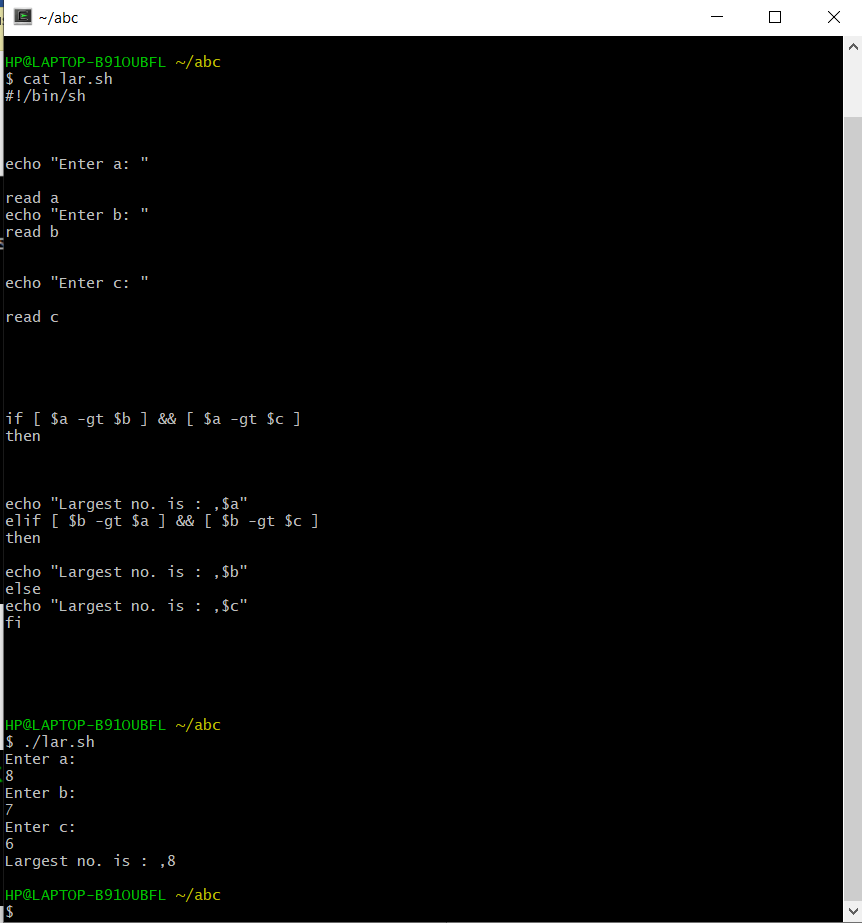
echo “Largest no. is : ,$b”

else

echo “Largest no. is: ,$c”

fi

**Output: Screenshots**



**Experiment No: 6**

Student Name and Roll Number: Chayan 18csu054

Semester /Section: 5/A

Link to Code: https://github.com/chayangulati321/Operating-System

Date: 7/09/20

Faculty Signature:

Remarks:

**Objective**

Write a shell program to find average of N numbers

**Program Outcome**

Student will be able to understand Write a shell program to find average of N numbers

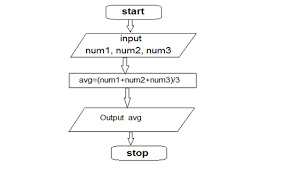
**Problem Statement**

Write a shell program to find average of N numbers

**Background Study**:

To find the average of N number we can check if a enter values that we have in our data and then check for the total value entered in the data then we can do the sigma of value divide by the total values .

**Algorithm/ flowchart**

****

**Code**

echo “Enter no. of terms : ”

read n

for ((i=0;i<n;i++))

{

echo “Enter each no. : “

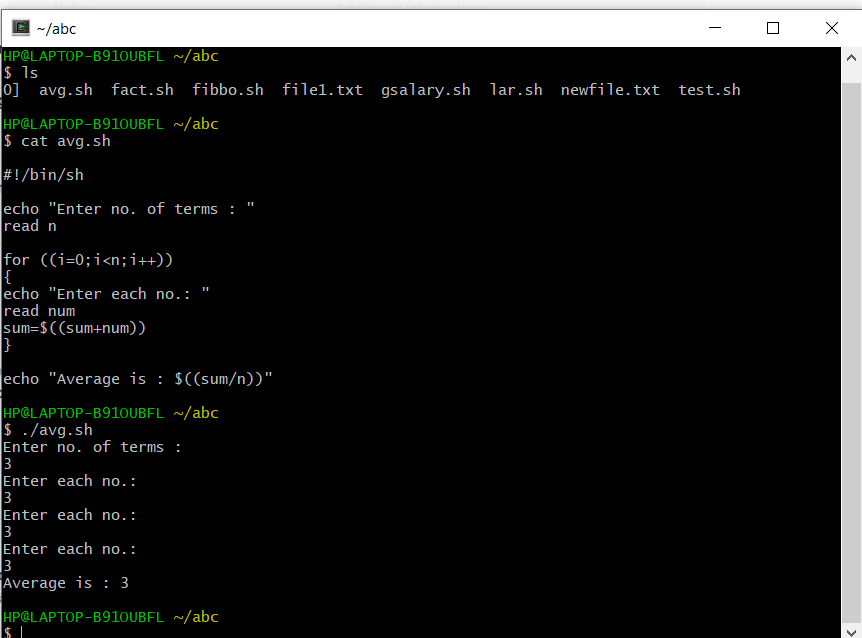
read num

sum=$((sum+num))

}

Echo “Average is : $((sum/n))”

**Output: Screenshots**



**Experiment No: 7**

Student Name and Roll Number: Chayan 18csu054

Semester /Section: 5/A

Link to Code: https://github.com/chayangulati321/Operating-System

Date: 12/09/20

Faculty Signature:

Remarks:

**Objective**

To familiarize the students about CPU scheduling Algorithms.

**Program Outcome**

* The students will understand the First-cum-first-serve and shortest job first algorithm.

**Problem Statement**

Implement the following CPU scheduling Algorithms.

1. FCFS
2. Shortest Job First

**Background Study:**

**FCFS**

* The simplest CPU-scheduling algorithm is the first-come, first-served (FCFS) scheduling algorithm. With this algorithm, processes are assigned the CPU in the order they request it.
* There is a single queue of ready processes.
* The implementation of the FCFS policy is easily managed with a FIFO queue. When a process enters the ready queue, its PCB is linked onto the tail of the queue.
* The average waiting time under the FCFS policy, however, is often quite long.

**Algorithm/ flowchart**

Input the processes along with their burst time

Find waiting time for all processes

As first process that comes need not to wait

Wt[i] = bt[i-1] + wt[i-1]

Turn around time = waiting time +burst time

Average waiting time = total waiting time / no. of processes

Average turn around time = total turn around time / no. of processes

**Code without Arrival Time**

def findWaitingTime(processes, n, bt, wt):

wt[0] = 0

for i in range(1, n ):

wt[i] = bt[i - 1] + wt[i - 1]

def findTurnAroundTime(processes, n, bt, wt, tat):

for i in range(n):

tat[i] = bt[i] + wt[i]

def findavgTime( processes, n, bt):

wt = [0] \* n

tat = [0] \* n

total\_wt = 0

total\_tat = 0

findWaitingTime(processes, n, bt, wt)

findTurnAroundTime(processes, n, bt, wt, tat)

print( "Processes Burst time " +

" Waiting time " +

" Turn around time")

for i in range(n):

total\_wt = total\_wt + wt[i]

total\_tat = total\_tat + tat[i]

print(" " + str(i + 1) + "\t\t" +

str(bt[i]) + "\t " +

str(wt[i]) + "\t\t " +

str(tat[i]))

print( "Average waiting time = "+

str(total\_wt / n))

print("Average turn around time = "+

str(total\_tat / n))

if \_\_name\_\_ =="\_\_main\_\_":

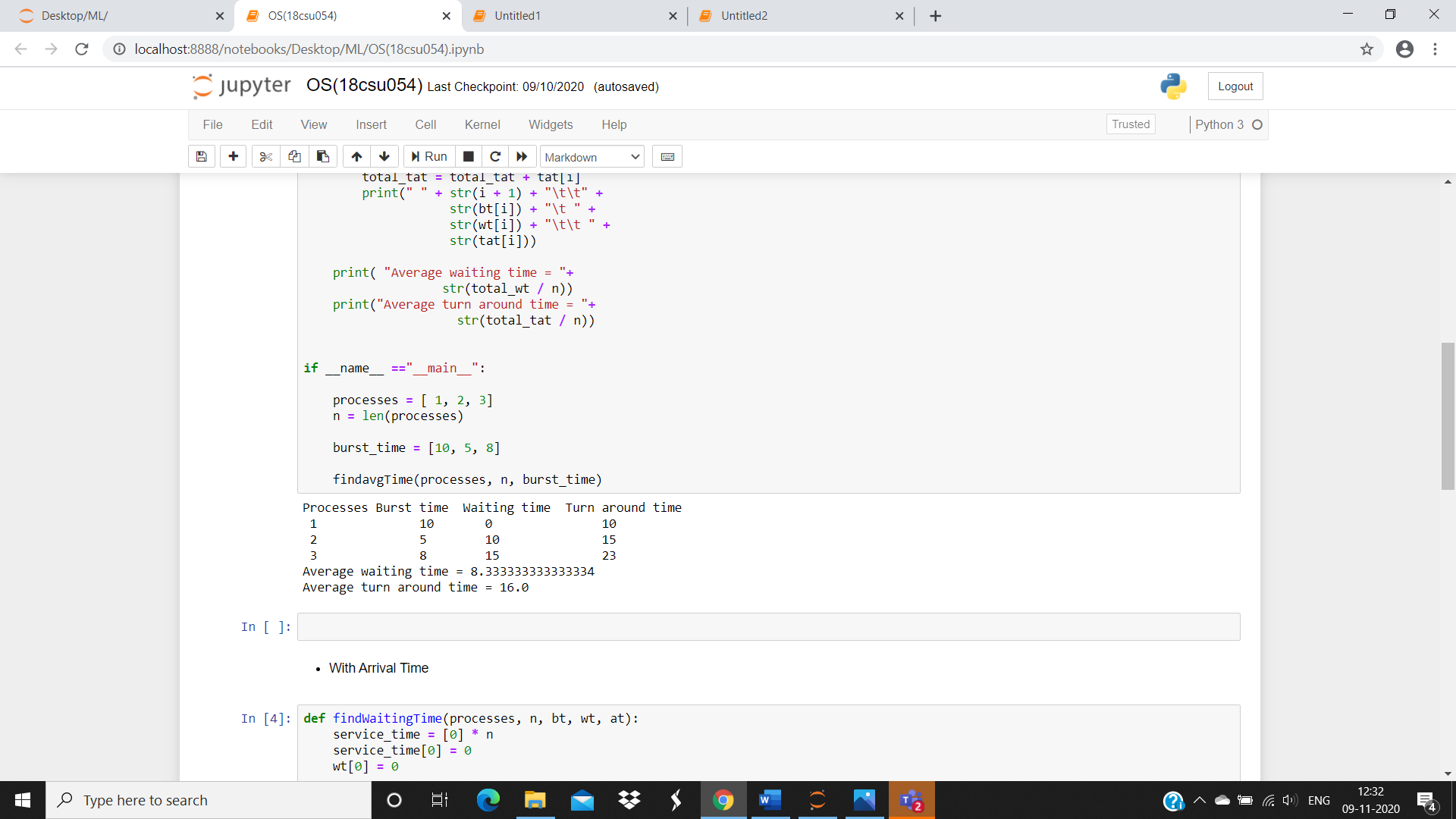
processes = [ 1, 2, 3]

n = len(processes)

burst\_time = [10, 5, 8]

findavgTime(processes, n, burst\_time)

**Output: Screenshots**



**Code with Arrival Time**

def findWaitingTime(processes, n, bt, wt, at):

service\_time = [0] \* n

service\_time[0] = 0

wt[0] = 0

for i in range(1, n):

service\_time[i] = (service\_time[i - 1] +

bt[i - 1])

wt[i] = service\_time[i] - at[i]

if (wt[i] < 0):

wt[i] = 0

def findTurnAroundTime(processes, n, bt, wt, tat):

for i in range(n):

tat[i] = bt[i] + wt[i]

def findavgTime(processes, n, bt, at):

wt = [0] \* n

tat = [0] \* n

findWaitingTime(processes, n, bt, wt, at)

findTurnAroundTime(processes, n, bt, wt, tat)

print("Processes Burst Time Arrival Time Waiting",

"Time Turn-Around Time Completion Time \n")

total\_wt = 0

total\_tat = 0

for i in range(n):

total\_wt = total\_wt + wt[i]

total\_tat = total\_tat + tat[i]

compl\_time = tat[i] + at[i]

print(" ", i + 1, "\t\t", bt[i], "\t\t", at[i],

"\t\t", wt[i], "\t\t ", tat[i], "\t\t ", compl\_time)

print("Average waiting time = %.5f "%(total\_wt /n))

print("\nAverage turn around time = ", total\_tat / n)

if \_\_name\_\_ =="\_\_main\_\_":

processes = [1, 2, 3]

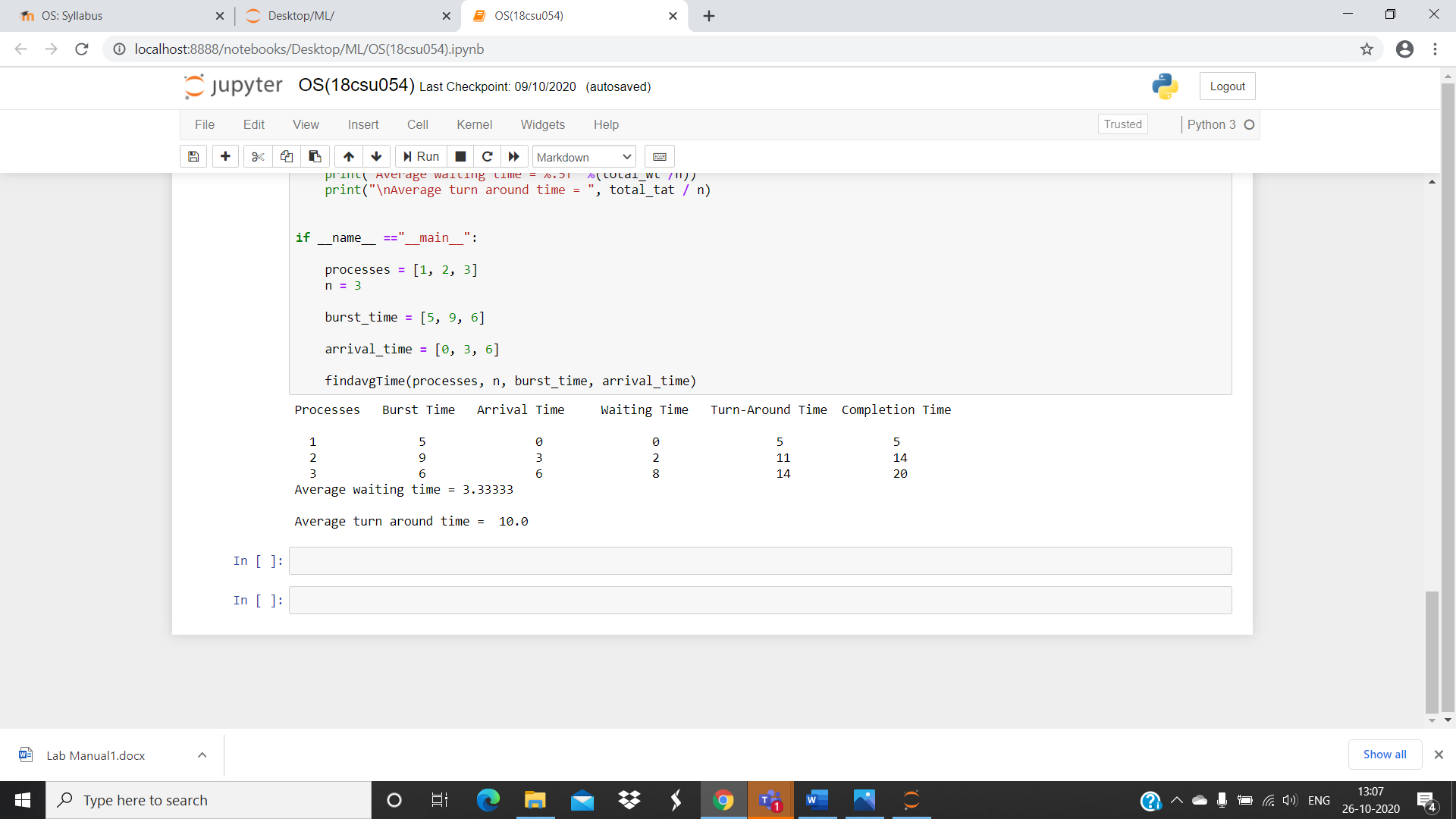
n = 3

burst\_time = [5, 9, 6]

arrival\_time = [0, 3, 6]

findavgTime(processes, n, burst\_time, arrival\_time)

**Output: Screenshots**



**SHORTEST JOB FIRST**

* This algorithm associates with each process the length of the process's next CPU burst.
* When the CPU is available, it is assigned to the process that has the smallest next CPU burst. If the next CPU bursts of two processes are the same, FCFS scheduling is used.
* The SJF scheduling algorithm gives the minimum average waiting time for a given set of processes
* The real difficulty with the SJF algorithm knows the length of the next CPU request.
* Shortest Job first has the advantage of having minimum average waiting time among all scheduling algorithms.
* It is a Greedy Algorithm.
* It may cause starvation if shorter processes keep coming. This problem can be solved using the concept of aging.

**Algorithm/ flowchart**

Sort all the process according to the arrival time.

Then select that process which has minimum arrival time and minimum Burst time.

After completion of process make a pool of process which after till the completion of previous process and select that process among the pool which is having minimum Burst time.

**Code for SJF** (with preemption)

n = int(input('Enter no of processes: '))

bt = [0] \* (n + 1)

at = [0] \* (n + 1)

abt = [0] \* (n + 1)

for i in range(n):

abt[i] = int(input('Enter the burst time for process {} : '.format(i + 1)))

at[i] = int(input('Enter the arrival time for process {} : '.format(i + 1)))

bt[i] = [abt[i], at[i], i]

bt.pop(-1)

#print(abt)

print(bt)

sumbt = 0

i = 0

ll = []

for i in range(0, sum(abt)):

l = [j for j in bt if j[1] <= i]

l.sort(key=lambda x: x[0])

#print(l, l[0][2])

bt[bt.index(l[0])][0] -= 1

for k in bt:

if k[0] == 0:

t = bt.pop(bt.index(k))

ll.append([k, i + 1])

#print(ll)

ct = [0] \* (n + 1)

tat = [0] \* (n + 1)

wt = [0] \* (n + 1)

for i in ll:

#print(i, i[0], i[1], i[0][2])

ct[i[0][2]] = i[1]

#abt[i[0][3]] = i[0][2]

for i in range(len(ct)):

tat[i] = ct[i] - at[i]

wt[i] = tat[i] - abt[i]

ct.pop(-1)

wt.pop(-1)

tat.pop(-1)

abt.pop(-1)

at.pop(-1)

print('\nBT\tAT\tCT\tTAT\tWT')

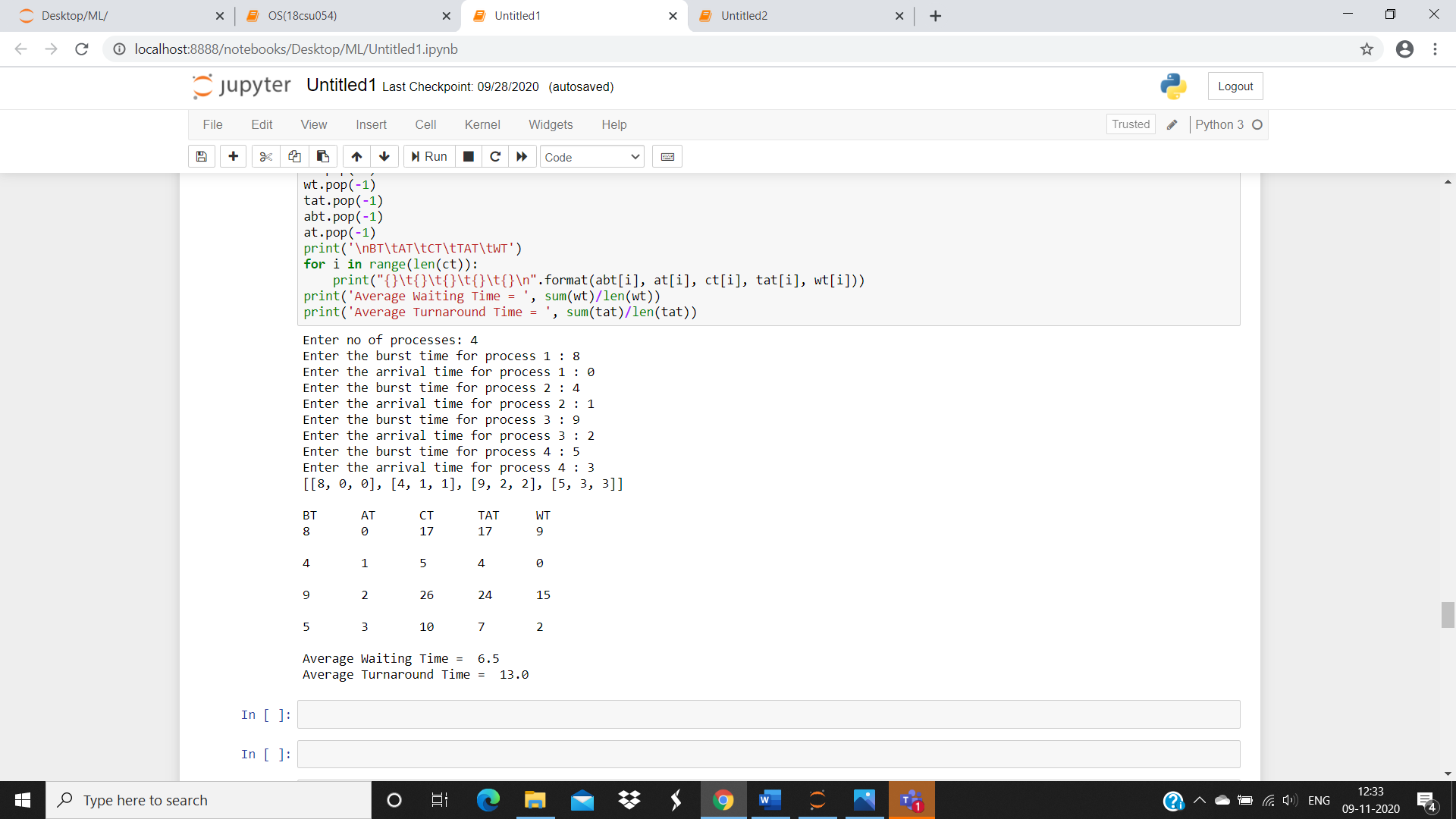
for i in range(len(ct)):

print("{}\t{}\t{}\t{}\t{}\n".format(abt[i], at[i], ct[i], tat[i], wt[i]))

print('Average Waiting Time = ', sum(wt)/len(wt))

print('Average Turnaround Time = ', sum(tat)/len(tat))

**Output: Screenshots**



**Code for SJF** (without preemption)

bt=[]

print("Enter the number of process: ")

n=int(input())

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 Turn Around Time")

for i in range(0,n):

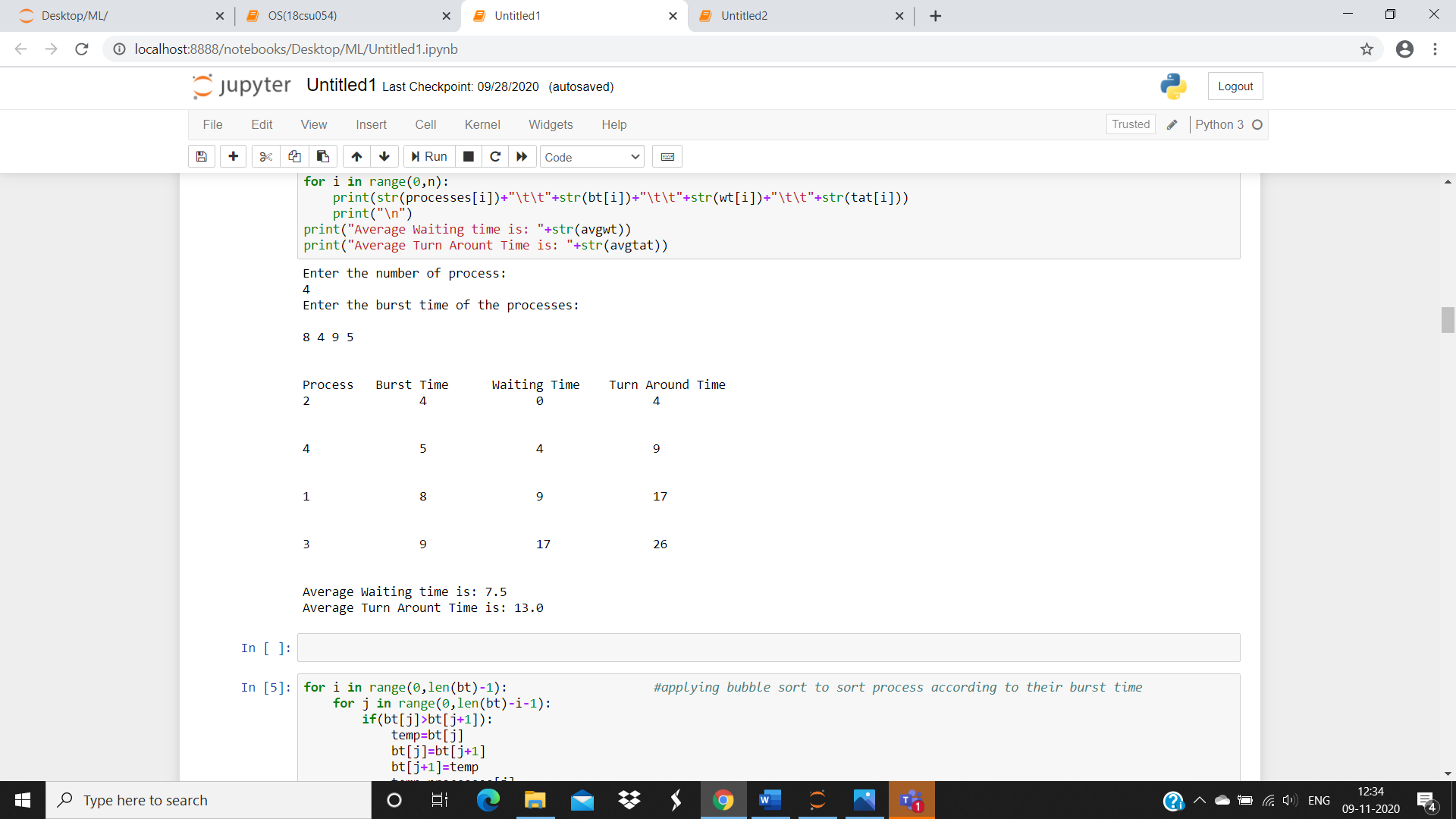
print(str(processes[i])+"\t\t"+str(bt[i])+"\t\t"+str(wt[i])+"\t\t"+str(tat[i]))

print("\n")

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

print("Average Turn Arount Time is: "+str(avgtat))

**Output: Screenshots**



**Experiment No: 8 (i)**

Student Name and Roll Number: Chayan 18csu054

Semester /Section: 5/A

Link to Code: https://github.com/chayangulati321/Operating-System

Date: 26/09/20

Faculty Signature:

Remarks:

**Objective**

To familiarize the students about CPU scheduling Algorithms.

**Program Outcome**

* The students will understand the Priority Scheduling with Arrival and without Arrival Time

**Problem Statement**

Implement the following CPU scheduling Algorithms.

Priority Scheduling

**Background Study:**

**Priority**

* Priority scheduling is a non-preemptive algorithm and one of the most common scheduling algorithms in batch systems.
* Each process is assigned a priority. Process with highest priority is to be executed first and so on.
* Processes with same priority are executed on first come first served basis.
* Priority can be decided based on memory requirements, time requirements or any other resource requirement.

**Algorithm/ flowchart**

First input the processes with their burst time

And priority.

Sort the processes, burst time and priority according to the priority.

Now simply apply FCFS algorithm

**Code for Priority Without Arrival Time**

print("Enter the number of processess: ")

n=int(input())

processes=[]

for i in range(0,n):

processes.insert(i,i+1)

#Input Burst time of every process

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

bt=list(map(int, input().split()))

#Input Priority of every process

print("\nEnter the priority of the processes: \n")

priority=list(map(int, input().split()))

tat=[]

wt=[]

#Sorting processes burst time, on the basis of their priority

for i in range(0,len(priority)-1):

for j in range(0,len(priority)-i-1):

if(priority[j]>priority[j+1]):

swap=priority[j]

priority[j]=priority[j+1]

priority[j+1]=swap

swap=bt[j]

bt[j]=bt[j+1]

bt[j+1]=swap

swap=processes[j]

processes[j]=processes[j+1]

processes[j+1]=swap

wt.insert(0,0)

tat.insert(0,bt[0])

#Calculating of waiting time and Turn Around Time of each process

for i in range(1,len(processes)):

wt.insert(i,wt[i-1]+bt[i-1])

tat.insert(i,wt[i]+bt[i])

#calculating average waiting time and average turn around time

avgtat=0

avgwt=0

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

avgwt=avgwt+wt[i]

avgtat=avgtat+tat[i]

avgwt=float(avgwt)/n

avgwt=float(avgtat)/n

print("\n")

print("Process\t Burst Time\t Waiting Time\t Turn Around Time")

for i in range(0,n):

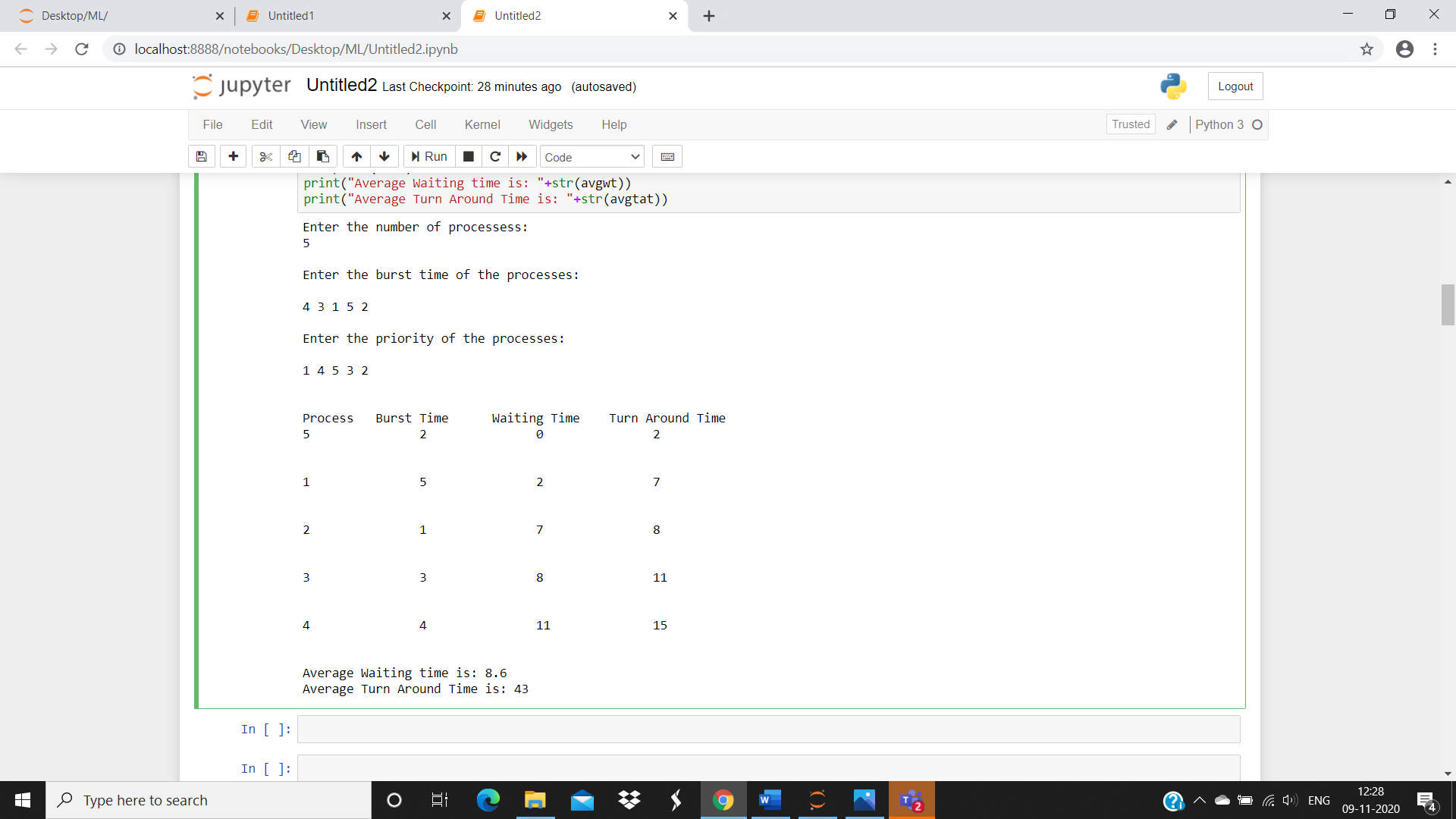
print(str(processes[i])+"\t\t"+str(bt[i])+"\t\t"+str(wt[i])+"\t\t"+str(tat[i]))

print("\n")

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

print("Average Turn Around Time is: "+str(avgtat))

**Output: Screenshots**



**Code for Priority With Arrival Time**

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

# Initilising 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])

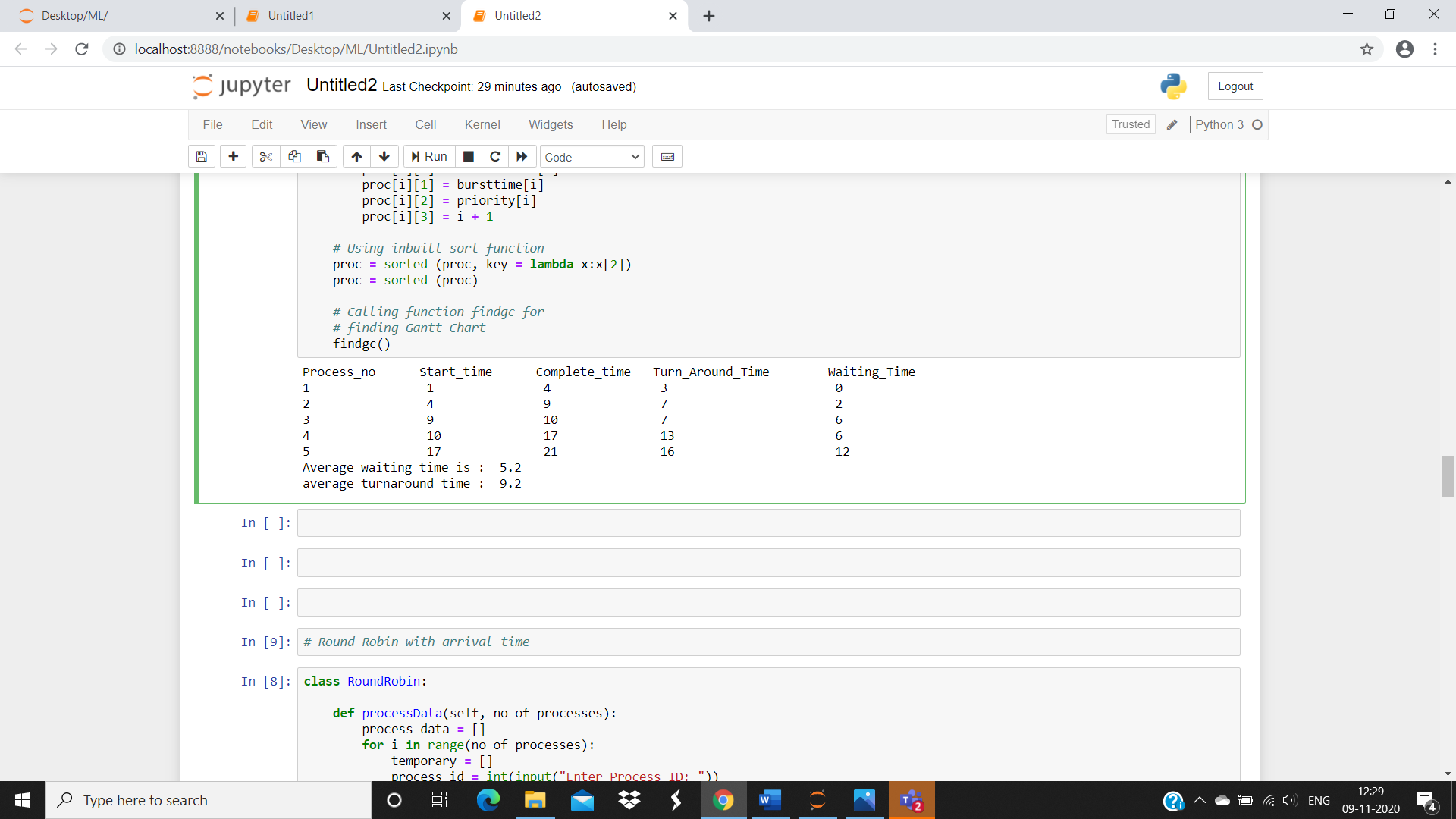
proc = sorted (proc)

# Calling function findgc for

# finding Gantt Chart

findgc()

**Output: Screenshots**



**Experiment No: 8 (ii)**

Student Name and Roll Number: Chayan 18csu054

Semester /Section: 5/A

Link to Code: https://github.com/chayangulati321/Operating-System

Date: 1/10/20

Faculty Signature:

Remarks:

**Objective**

To familiarize the students about CPU scheduling Algorithms.

**Program Outcome**

* The students will understand the Round robin Scheduling with Arrival and without Arrival Time

**Problem Statement**

Implement the following CPU scheduling Algorithms.

**Round robin Scheduling**

**Background Study:**

**Round robin**

* Round Robin is the preemptive process scheduling algorithm.
* Each process is provided a fix time to execute, it is called a quantum.

**Algorithm/ flowchart**

Create an array rem bt[] to keep track of remaining

Burst time of processes this array is initially a

Copy of bt[]

Create another array wt[] to store waiting times

Of processes . Initialise this array to 0.

Initialise time : t =0

Keep traversing the all processes while all processes

Are not done . Do following for i’th process if it is

Not done yet .

If rem bt[i] > quantum

T = t +quantum

Bt rem[i] = quantum

Else // Last cycle for this process

T = t + bt rem[i]

Wt[i] = t- bt[i]

Bt rem[i] = 0;

**Code for Round robin Without Arrival Time**

def findWaitingTime(processes, n, bt, wt, q):

rem\_bt = [0] \* n

for i in range(n):

rem\_bt[i] = bt[i]

t = 0

while(1):

flag = True

for i in range(n):

if (rem\_bt[i] > 0) :

flag = False

if (rem\_bt[i] > q) :

t += q

rem\_bt[i] -= q

else:

t = t + rem\_bt[i]

wt[i] = t - bt[i]

rem\_bt[i] = 0

if (flag == True):

break

def findTurnAroundTime(processes, n, bt, wt, tat):

for i in range(n):

tat[i] = bt[i] + wt[i]

def findavgTime(processes, n, bt, q):

wt = [0] \* n

tat = [0] \* n

findWaitingTime(processes, n, bt, wt, q)

findTurnAroundTime(processes, n, bt, wt, tat)

print("Processes Burst Time Waiting Time Turn-Around Time")

total\_wt = 0

total\_tat = 0

for i in range(n):

total\_wt = total\_wt + wt[i]

total\_tat = total\_tat + tat[i]

print(" ", i + 1, " \t\t", bt[i], " \t\t", wt[i], " \t\t", tat[i])

print("Average waiting time = %.5f "%(total\_wt /n) )

print("Average turn around time = %.5f "% (total\_tat / n))

if \_\_name\_\_ =="\_\_main\_\_":

proc = [1, 2, 3]

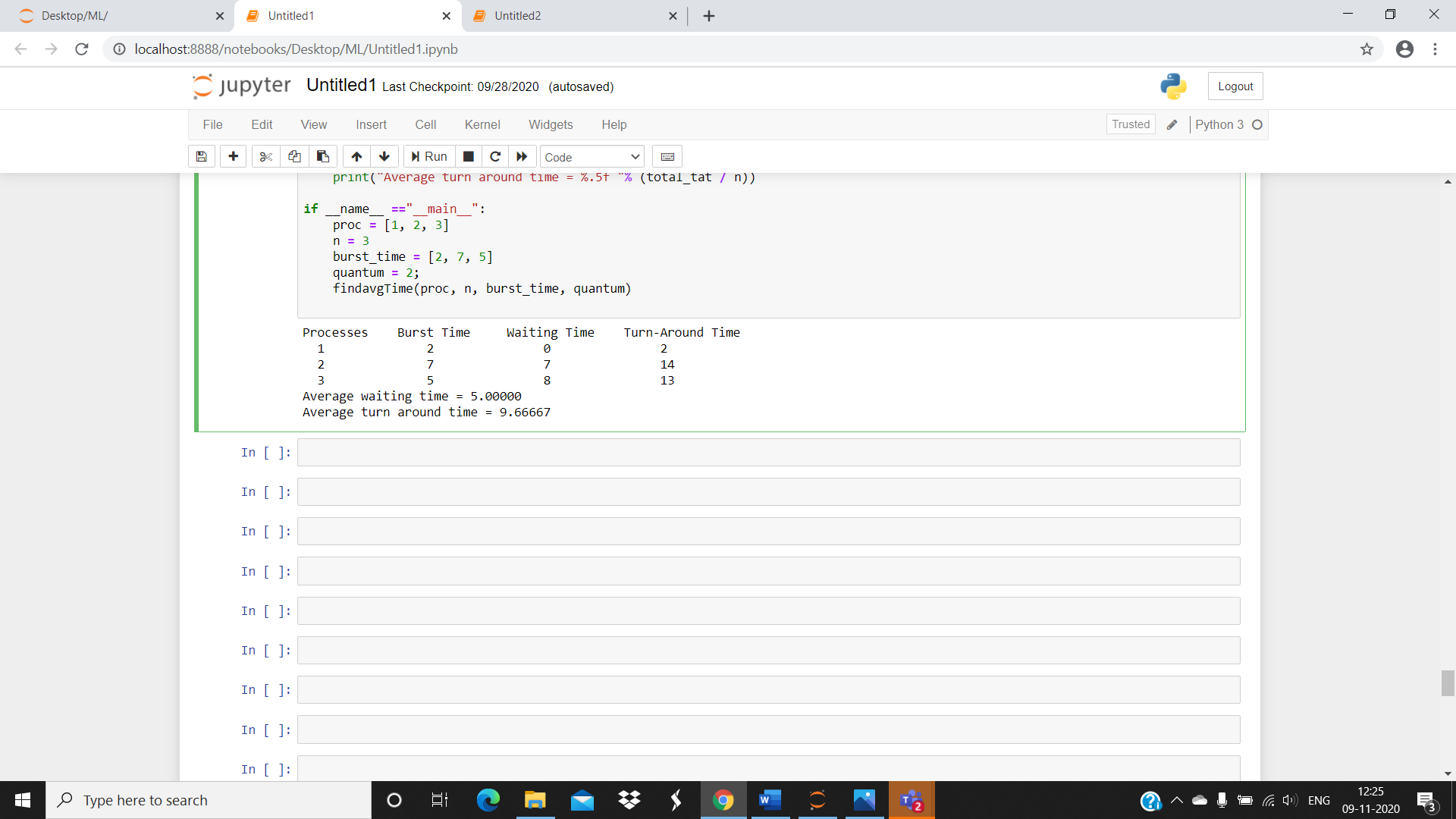
n = 3

burst\_time = [2, 7, 5]

quantum = 2;

findavgTime(proc, n, burst\_time, quantum)

**Output: Screenshots**



**Code for Round robin With Arrival Time**

class RoundRobin:

def processData(self, no\_of\_processes):

process\_data = []

for i in range(no\_of\_processes):

temporary = []

process\_id = int(input("Enter Process ID: "))

arrival\_time = int(input(f"Enter Arrival Time for Process {process\_id}: "))

burst\_time = int(input(f"Enter Burst Time for Process {process\_id}: "))

temporary.extend([process\_id, arrival\_time, burst\_time, 0, burst\_time])

'''

'0' is the state of the process. 0 means not executed and 1 means execution complete

'''

process\_data.append(temporary)

time\_slice = int(input("Enter Time Slice: "))

RoundRobin.schedulingProcess(self, process\_data, time\_slice)

def schedulingProcess(self, process\_data, time\_slice):

start\_time = []

exit\_time = []

executed\_process = []

ready\_queue = []

s\_time = 0

process\_data.sort(key=lambda x: x[1])

'''

Sort processes according to the Arrival Time

'''

while 1:

normal\_queue = []

temp = []

for i in range(len(process\_data)):

if process\_data[i][1] <= s\_time and process\_data[i][3] == 0:

present = 0

if len(ready\_queue) != 0:

for k in range(len(ready\_queue)):

if process\_data[i][0] == ready\_queue[k][0]:

present = 1

'''

The above if loop checks that the next process is not a part of ready\_queue

'''

if present == 0:

temp.extend([process\_data[i][0], process\_data[i][1], process\_data[i][2], process\_data[i][4]])

ready\_queue.append(temp)

temp = []

'''

The above if loop adds a process to the ready\_queue only if it is not already present in it

'''

if len(ready\_queue) != 0 and len(executed\_process) != 0:

for k in range(len(ready\_queue)):

if ready\_queue[k][0] == executed\_process[len(executed\_process) - 1]:

ready\_queue.insert((len(ready\_queue) - 1), ready\_queue.pop(k))

'''

The above if loop makes sure that the recently executed process is appended at the end of ready\_queue

'''

elif process\_data[i][3] == 0:

temp.extend([process\_data[i][0], process\_data[i][1], process\_data[i][2], process\_data[i][4]])

normal\_queue.append(temp)

temp = []

if len(ready\_queue) == 0 and len(normal\_queue) == 0:

break

if len(ready\_queue) != 0:

if ready\_queue[0][2] > time\_slice:

'''

If process has remaining burst time greater than the time slice, it will execute for a time period equal to time slice and then switch

'''

start\_time.append(s\_time)

s\_time = s\_time + time\_slice

e\_time = s\_time

exit\_time.append(e\_time)

executed\_process.append(ready\_queue[0][0])

for j in range(len(process\_data)):

if process\_data[j][0] == ready\_queue[0][0]:

break

process\_data[j][2] = process\_data[j][2] - time\_slice

ready\_queue.pop(0)

elif ready\_queue[0][2] <= time\_slice:

'''

If a process has a remaining burst time less than or equal to time slice, it will complete its execution

'''

start\_time.append(s\_time)

s\_time = s\_time + ready\_queue[0][2]

e\_time = s\_time

exit\_time.append(e\_time)

executed\_process.append(ready\_queue[0][0])

for j in range(len(process\_data)):

if process\_data[j][0] == ready\_queue[0][0]:

break

process\_data[j][2] = 0

process\_data[j][3] = 1

process\_data[j].append(e\_time)

ready\_queue.pop(0)

elif len(ready\_queue) == 0:

if s\_time < normal\_queue[0][1]:

s\_time = normal\_queue[0][1]

if normal\_queue[0][2] > time\_slice:

'''

If process has remaining burst time greater than the time slice, it will execute for a time period equal to time slice and then switch

'''

start\_time.append(s\_time)

s\_time = s\_time + time\_slice

e\_time = s\_time

exit\_time.append(e\_time)

executed\_process.append(normal\_queue[0][0])

for j in range(len(process\_data)):

if process\_data[j][0] == normal\_queue[0][0]:

break

process\_data[j][2] = process\_data[j][2] - time\_slice

elif normal\_queue[0][2] <= time\_slice:

'''

If a process has a remaining burst time less than or equal to time slice, it will complete its execution

'''

start\_time.append(s\_time)

s\_time = s\_time + normal\_queue[0][2]

e\_time = s\_time

exit\_time.append(e\_time)

executed\_process.append(normal\_queue[0][0])

for j in range(len(process\_data)):

if process\_data[j][0] == normal\_queue[0][0]:

break

process\_data[j][2] = 0

process\_data[j][3] = 1

process\_data[j].append(e\_time)

t\_time = RoundRobin.calculateTurnaroundTime(self, process\_data)

w\_time = RoundRobin.calculateWaitingTime(self, process\_data)

RoundRobin.printData(self, process\_data, t\_time, w\_time, executed\_process)

def calculateTurnaroundTime(self, process\_data):

total\_turnaround\_time = 0

for i in range(len(process\_data)):

turnaround\_time = process\_data[i][5] - process\_data[i][1]

'''

turnaround\_time = completion\_time - arrival\_time

'''

total\_turnaround\_time = total\_turnaround\_time + turnaround\_time

process\_data[i].append(turnaround\_time)

average\_turnaround\_time = total\_turnaround\_time / len(process\_data)

'''

average\_turnaround\_time = total\_turnaround\_time / no\_of\_processes

'''

return average\_turnaround\_time

def calculateWaitingTime(self, process\_data):

total\_waiting\_time = 0

for i in range(len(process\_data)):

waiting\_time = process\_data[i][6] - process\_data[i][4]

'''

waiting\_time = turnaround\_time - burst\_time

'''

total\_waiting\_time = total\_waiting\_time + waiting\_time

process\_data[i].append(waiting\_time)

average\_waiting\_time = total\_waiting\_time / len(process\_data)

'''

average\_waiting\_time = total\_waiting\_time / no\_of\_processes

'''

return average\_waiting\_time

def printData(self, process\_data, average\_turnaround\_time, average\_waiting\_time, executed\_process):

process\_data.sort(key=lambda x: x[0])

'''

Sort processes according to the Process ID

'''

print("Process\_ID Arrival\_Time Rem\_Burst\_Time Completed Original\_Burst\_Time Completion\_Time Turnaround\_Time Waiting\_Time")

for i in range(len(process\_data)):

for j in range(len(process\_data[i])):

print(process\_data[i][j], end=" ")

print()

print(f'Average Turnaround Time: {average\_turnaround\_time}')

print(f'Average Waiting Time: {average\_waiting\_time}')

print(f'Sequence of Processes: {executed\_process}')

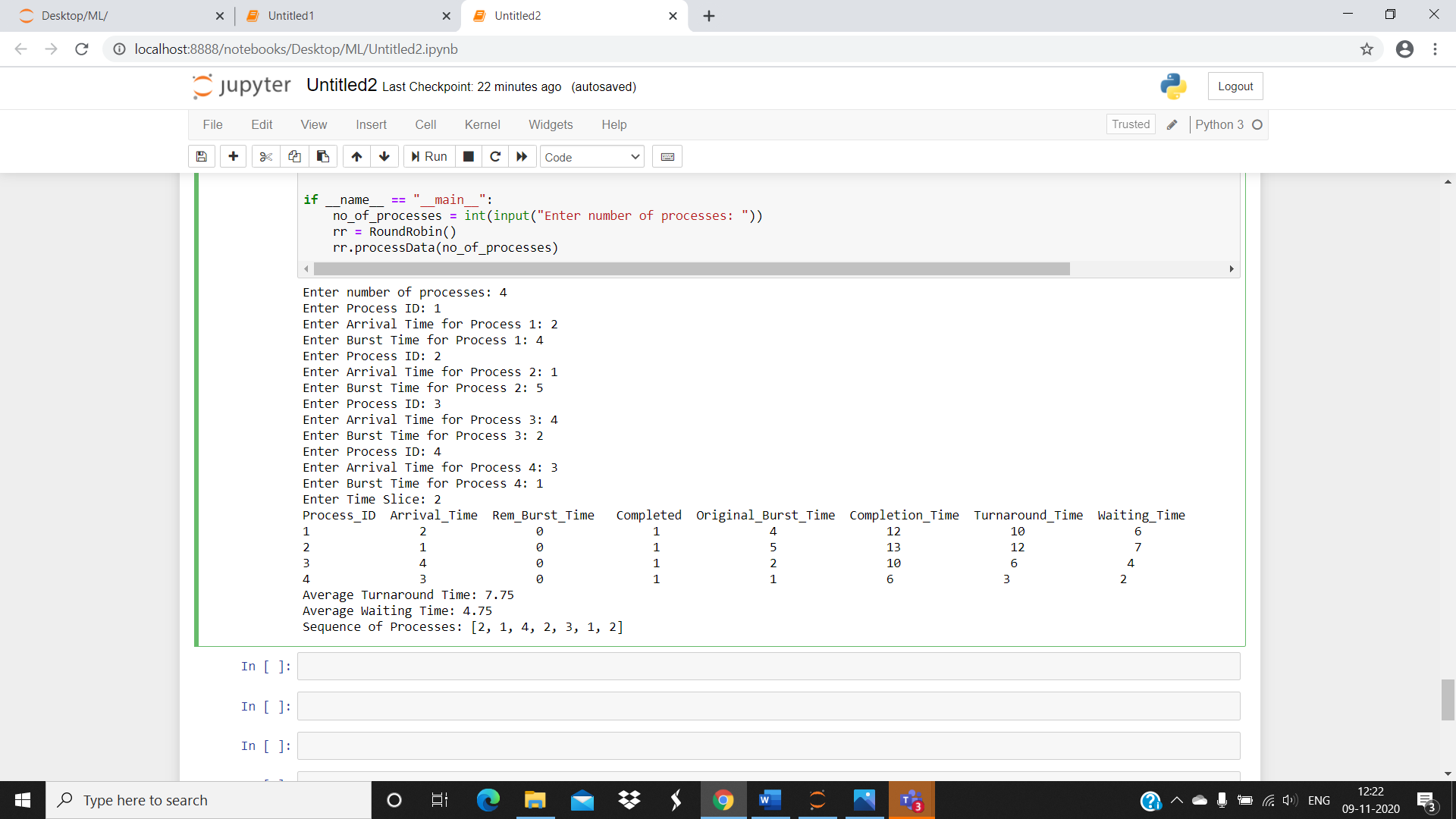
if \_\_name\_\_ == "\_\_main\_\_":

no\_of\_processes = int(input("Enter number of processes: "))

rr = RoundRobin()

rr.processData(no\_of\_processes)

**Output: Screenshots**



**Case Study of Dos, Windows and linux**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Feature** | **Windows** | **Linux** | **DOS** | **Linux** |
| **Ability to** | You can change the | You can rewrite the | You can download | Overall Linux is the most |
| **Customize** | background and general | code for Linux so | expansions for DOS | customisable, because it is |
|  | colour scheme and | that you can | so that you can | open source and you can |
|  | fonts for your windows | customise it further, | customise it. | alter the codes for it. DOS |
|  | desktop. | meaning that you |  | can also be open source, |
|  |  | can have a |  | but there is no GUI to |
|  |  | completely different |  | customise. Windows is |
|  |  | layout than everyone |  | relatively customisable, but |
|  |  | else. |  | not fully. You can mainly |
|  |  |  |  | just change the colours and |
|  |  |  |  | looks of things slightly. |
| **Support for** | Windows supports | Linux doesn’t have | DOS doesn’t | With Windows being the |
| **connectivity for** | connectivity for | as wide a range of | provide much | most popular OS out of  the |
| **portable media** | different media, | support for | connectivity, from | 3, it has more  connectivity |
|  | including USB drives, | connectivity but you | the fixed HDD only. | available, and expansions |
|  | CD/DVD, Syncing, | can download a |  | available to enable more |
|  | Bluetooth and Wi-Fi. | Software |  | connectivity. You can also |
|  |  | development kit to |  | download expansions to |
|  |  | improve its |  | make Linux more |
|  |  | connectivity. |  | supportive for connectivity, |
|  |  |  |  | but there are not as many |
|  |  |  |  | things available as for |
|  |  |  |  | windows. As DOS is an old |
|  |  |  |  | OS and isn’t really used |
|  |  |  |  | anymore, it doesn’t really |
|  |  |  |  | have any support for |
|  |  |  |  | connecting media. Newer |
|  |  |  |  | versions of DOS might, |
|  |  |  |  | because they are built with |
|  |  |  |  | Windows on top of them. |
| **Security** | Windows has an | For Linux you have | DOS does not | The most secure would be |
|  | administrative user that | to enter the | support multiple | Linux, because of its ability |
|  | can make changes to | administrator’s | users so there are | to customise, viruses |
|  | the computer, but it has | password in order to | no threats from | wouldn’t be able to access |
|  | other users that can | make changes or | other user on the | the computer because it |
|  | perform small tasks. | download things, so | computer. DOS | wouldn’t be specific |
|  | You need the | it is harder for | didn’t really have | enough. Because of |
|  | administrator’s | harmful programs to | much security | Windows’ popularity, it |
|  | password to change | gain access to your | when it first came | makes it the most |
|  | anything if on another | computer. You can | out, because of the | vulnerable to attacks. This |
|  | user. | also change the | low threat. | is also because you can’t |
|  |  | codes to make it |  | edit the code. DOS has the |
|  |  | more secure. |  | least security because it | |
|  |  |  |  | was made before viruses | |
|  |  |  |  | existed. | |
| **Stability** | Windows is relatively | Linux operating | DOS doesn’t have a | Overall I think that Linux is | |
|  | stable, but vista is more | systems are stable, | user interface, so it | the most stable, because it | |
|  | unstable than other | because you can | is much more | doesn’t have as much | |
|  | versions. Vista didn’t | rewrite the codes for | stable because it | software and other things | |
|  | sell because it crashed when it was launched, and Microsoft released windows 7 soon after it. | it. | doesn’t have to produce graphics. | running as Windows, and it  is more popular than DOS, so it is better tested and improved.  Windows crashes a lot and needs rebooting a lot because of the  amount of programs running on it. | |
| **Reliability** | Windows is not that reliable, with it crashing/needing to be restarted all of the time. Windows Vista was particularly unreliable, which is why it didn’t sell very well. | Linux is very reliable and is known for being run for months or even years without needing to be rebooted. | DOS is not very reliable because it crashes easily if a program doesn’t open properly or a wrong command is put in. | Linux is the most reliable out of the 3 because it doesn’t need rebooting or crash all of the time. It can run for months or even years without  need to be rebooted. If programs  don’t open correctly in DOS  then the system can crash,  because it was designed for the correct commands  to be entered and nothing else. | |
| **Ease of management** | Because Windows is so popular, people are more familiar with the way it works so it is fairly easy for everyone to use. | Later versions of Linux have greatly improved in their ease of management side, but because it is open source you have to know how to write programming language to alter things. | DOS isn’t very easy to manage unless you know the commands to perform different tasks. | Windows is probably the easiest to manage because of tutorials available to teach users and how most people are taught to use Windows over any other OS. Linux is simple as well to  an extent, but you have to know the commands to manage things past a certain point. DOS is purely command driven so it is easy to manage things  if you know the codes. | |
| **Associated utilities** | Windows is the most popular OS, so it has a much wider range of utilities. | Linux has a wide range of utilities available to download. | DOS doesn’t have many associated utilities, because it isn’t very popular any more. There aren’t many utilities available because it doesn’t have a GUI as well. | Windows has the most associated  utilities, developed by Microsoft and  many other companies. Linux has  a lot, but not as many, with many  people writing their own because it  is open source. DOS has the least  associated utilities, because it isn’t  used for applications because it  is just code. | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Cost** | The latest version of windows costsaround  £100. You also have to pay for other software you want onwindows. | Linux is free to download, so it doesn’t cost you anything. There are  software downloads that are free, such as open office etc | Because DOS isn’t actively developed any more, it is free of charge. | DOS is free of charge to download, and  so is Linux. This is better than Windows, which costs to download. Further  software for Windows also costs, and with the other operating systems of  software costs  a lot less or nothing  at all. |
| **Support for the** | Windows has its own | Linux has books | There are websites | There is most support |
| **User** | help section, and | available for help | and forums | available for windows, with |
|  | websites/forums where | and also online | available for help | a help section actually in |
|  | you can talk to people | forums. | with DOS, but they | the OS and with many |
|  | and gain information |  | provide more basic | websites and books and |
|  | about windows. There |  | Information | other sources. Linux has a |
|  | are also books available |  | because DOS isn’t | lot of support as well, but |
|  | for each version of |  | very popular. | not as much as Windows. |
|  | windows to help you. |  |  | DOS has the least amount |
|  |  |  |  | of support available. |
|  |  |  |  | Windows users need the |
|  |  |  |  | most support because of |
|  |  |  |  | its stability. |

**Experiment No: 9**

Student Name and Roll Number: Chayan 18csu054

Semester /Section: 5/A

Link to Code: https://github.com/chayangulati321/Operating-System

Date: 28/10/20

Faculty Signature:

Remarks:

**Objective**

To familiarize the students to Reader Writer Problem.

**Program Outcome**

* The students will understand about the semaphores

**Problem Statement**

Implement the reader writer Algorithm

**Background Study:**

There is a shared data which can be accessed by multiple processes. There are two types of processes in the system i.e a reader and a writer. Any number of readers can read from the shared resource simultaneously, but only one writer can write to the shared resource. When a writer is writing data to the resource, no other process can access the resource. A writer cannot write to the resource if there are non zero number of readers accessing the resource at that time

**Algorithm:**

**Writer:**

wait(wrt);

. . .

writing is performed

. . .

signal(wrt);

**Reader:**

wait(mutex);

readcount := readcount + 1;

if readcount = 1 then

wait(wrt); //since reader is reading, prevent writing for writer

signal(mutex);

. . .

reading is performed

. . .

wait(mutex);

readcount := readcount - 1;

if readcount = 0 then signal(wrt); //since no reader is present allow

writer to write

signal(mutex);

**Code:**

import threading as thread

import random

global x

x = 0

lock = thread.Lock()

def Reader():

global x

print('Reader is Reading!')

lock.acquire() # Acquire the lock before Reading (mutex approach)

print('Shared Data:', x)

lock.release() # Release the lock after Reading

print()

def Writer():

global x

print('Writer is Writing!')

lock.acquire() # Acquire the lock before Writing

x += 1 # Write on the shared memory

print('Writer is Releasing the lock!')

lock.release() # Release the lock after Writing

print()

if \_\_name\_\_ == '\_\_main\_\_':

for i in range(0, 10):

randomNumber = random.randint(0, 100) # Generate a Random number between 0 to 100

if(randomNumber > 50):

Thread1 = thread.Thread(target = Reader)

Thread1.start()

else:

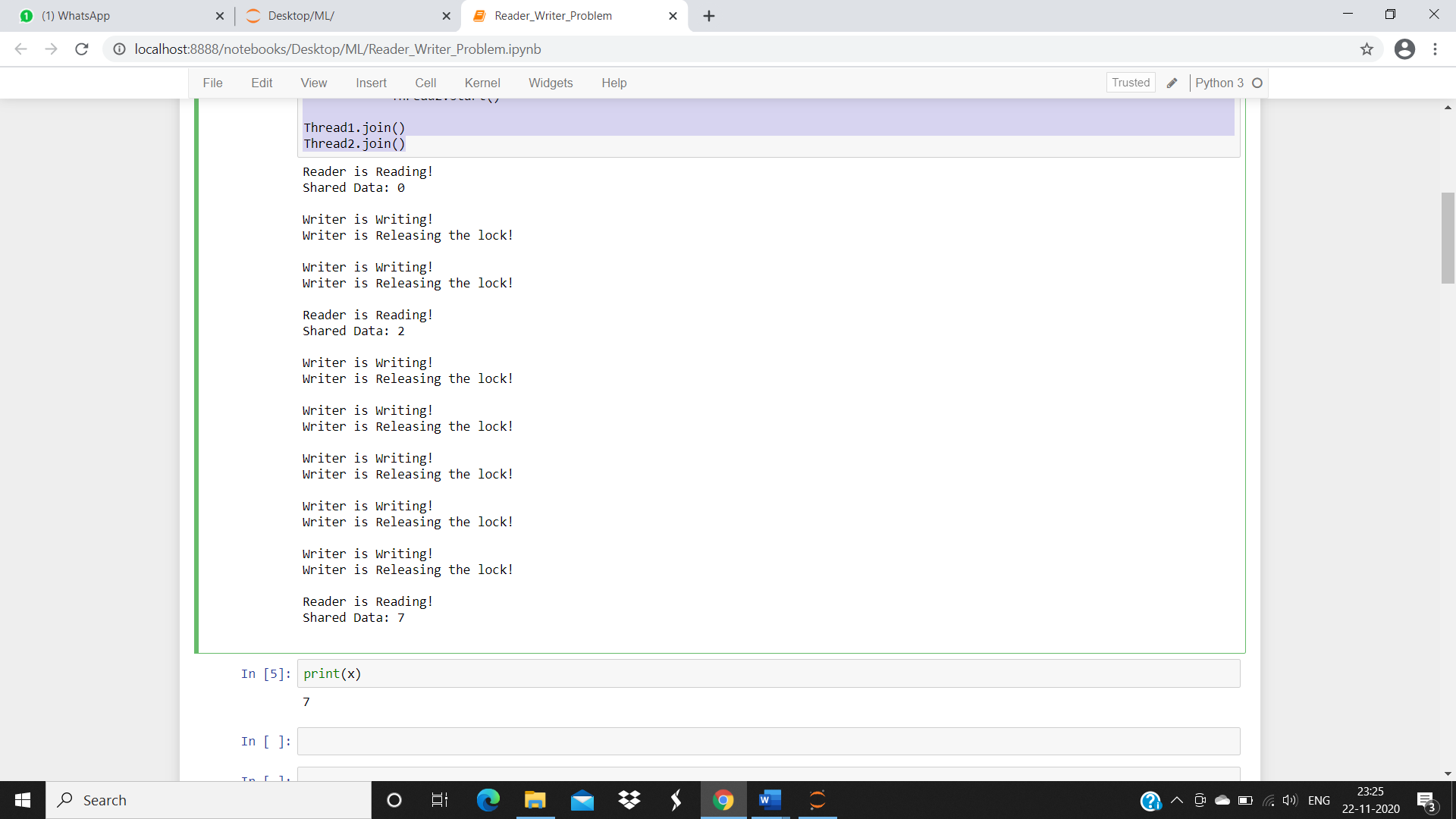
Thread2 = thread.Thread(target = Writer)

Thread2.start()

Thread1.join()

Thread2.join()

**Output: Screenshots**



**Experiment No: 10**

Student Name and Roll Number: Chayan 18csu054

Semester /Section: 5/A

Link to Code: https://github.com/chayangulati321/Operating-System

Date: 11/11/20

Faculty Signature:

Remarks:

**Objective**

To familiarize the students to Banker’s Algorithm.

**Program Outcome**

* The students will understand about the Banker’s Algorithm.

**Problem Statement**

Implement the Banker’s Algorithm

**Background Study:**

Banker's algorithm is a deadlock avoidance algorithm. It is named so because this algorithm is used in banking systems to determine whether a loan can be granted or not. Whenever a new process is created, it must exactly specify the maximum instances of each resource type that it needs. The banker's algorithm is a method used in deadlock avoidance technique in multiple instances of a resource type.

**Algorithm:**

Safety Algorithm:

1. Let Work and Finish be vectors of length ‘m’ and ‘n’ respectively.  
   Initialize: Work= Available  
   Finish [i]=false; for i=1,2,……,n
2. Find an i such that both  
   a) Finish [i]=false  
   b) Need\_i<=work  
     
   if no such i exists goto step (4)
3. Work=Work + Allocation\_i  
   Finish[i]= true  
   goto step(2)
4. If Finish[i]=true for all i,  
   then the system is in safe state.

**Code:**

# Number of processes

P = 5

# Number of resources

R = 3

# Function to find the need of each process

def calculateNeed(need, maxm, allot):

# Calculating Need of each P

for i in range(P):

for j in range(R):

# Need of instance = maxm instance -

# allocated instance

need[i][j] = maxm[i][j] - allot[i][j]

# Function to find the system is in

# safe state or not

def isSafe(processes, avail, maxm, allot):

need = []

for i in range(P):

l = []

for j in range(R):

l.append(0)

need.append(l)

# Function to calculate need matrix

calculateNeed(need, maxm, allot)

# Mark all processes as infinish

finish = [0] \* P

# To store safe sequence

safeSeq = [0] \* P

# Make a copy of available resources

work = [0] \* R

for i in range(R):

work[i] = avail[i]

# While all processes are not finished

# or system is not in safe state.

count = 0

while (count < P):

# Find a process which is not finish

# and whose needs can be satisfied

# with current work[] resources.

found = False

for p in range(P):

# First check if a process is finished,

# if no, go for next condition

if (finish[p] == 0):

# Check if for all resources

# of current P need is less

# than work

for j in range(R):

if (need[p][j] > work[j]):

break

# If all needs of p were satisfied.

if (j == R - 1):

# Add the allocated resources of

# current P to the available/work

# resources i.e.free the resources

for k in range(R):

work[k] += allot[p][k]

# Add this process to safe sequence.

safeSeq[count] = p

count += 1

# Mark this p as finished

finish[p] = 1

found = True

# If we could not find a next process

# in safe sequence.

if (found == False):

print("System is not in safe state")

return False

# If system is in safe state then

# safe sequence will be as below

print("System is in safe state.",

"\nSafe sequence is: ", end = " ")

print(\*safeSeq)

return True

# Driver code

if \_\_name\_\_ =="\_\_main\_\_":

processes = [0, 1, 2, 3, 4]

# Available instances of resources

avail = [3, 3, 2]

# Maximum R that can be allocated

# to processes

maxm = [[7, 5, 3], [3, 2, 2],

[9, 0, 2], [2, 2, 2],

[4, 3, 3]]

# Resources allocated to processes

allot = [[0, 1, 0], [2, 0, 0],

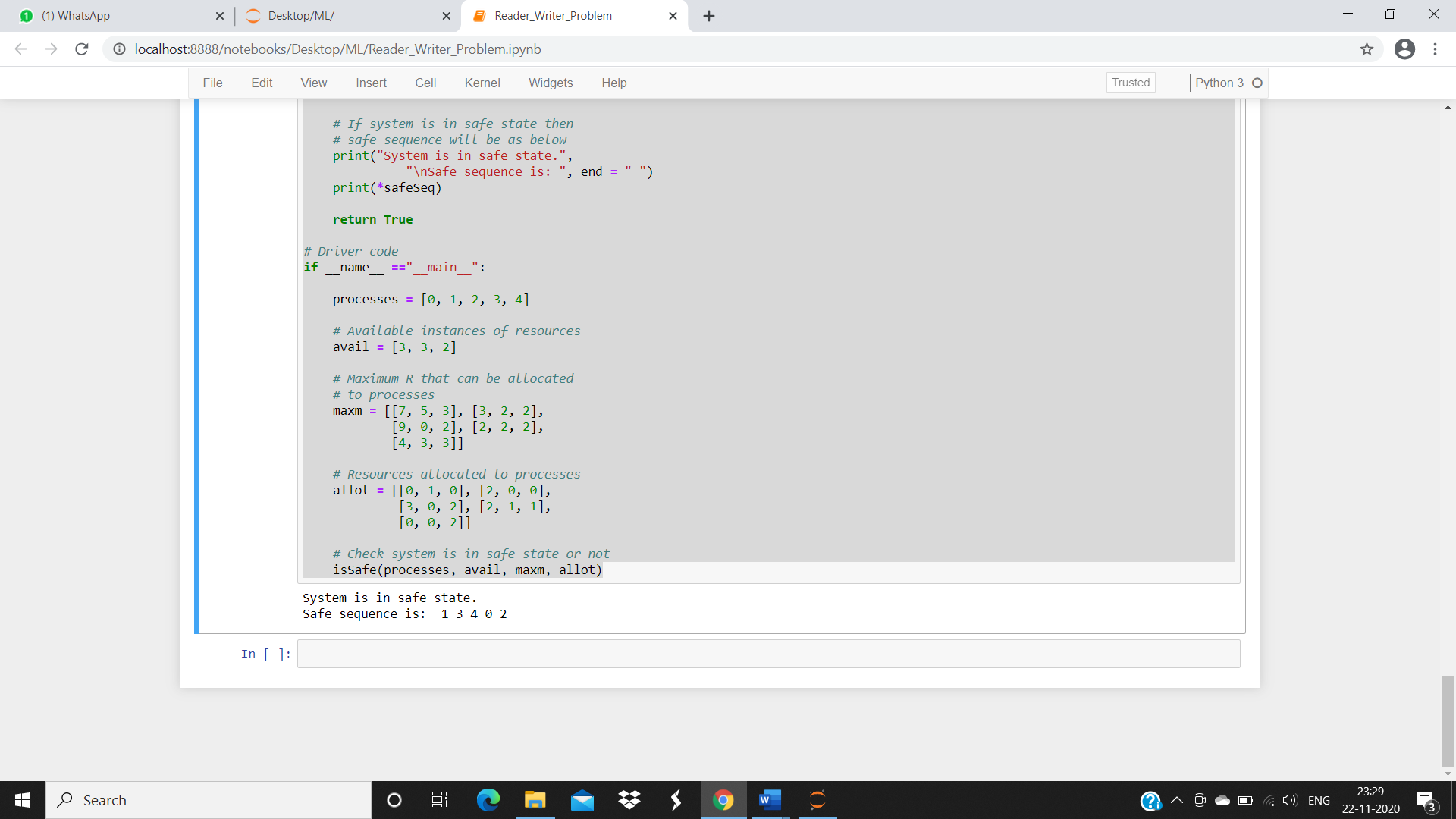
[3, 0, 2], [2, 1, 1],

[0, 0, 2]]

# Check system is in safe state or not

isSafe(processes, avail, maxm, allot)

**Output: Screenshots**



**ASSIGNMENT QUESTIONS**

**Assignment 1**

1. **How the OS ensure the non-interference between various processes stored in memory in case of multi-programming and time sharing systems.**

**Ans:** As in Time sharing systems each process is assigned with specific time and duration, which ensures that there is no interruption between other processes and talking about multi programming, in which job switching happens only when a job is completed so, the job scheduling is done in such a way that there is less to no interruption between other processes.

1. **What do you mean by Real time operating system? Give an example of it.**

**Ans:** Real Time systems are dedicated, embedded systems. In such systems, the OS must response to events within fixed periods of time to ensure correct performance.

* Real-time embedded systems most prevalent form of computers
* Vary considerable, special purpose, limited purpose OS, **real-time OS**
* Use expanding
* Many other special computing environments as well
* Some have OSes, some perform tasks without an OS

For example: flight control systems.

**Assignment 2**

1. **What is concept of Overlays? Explain with example.**

**Ans.** The concept of **overlays** is that whenever a process is running it will not use the complete program at the same time, it will use only some part of it.

The best example of overlays is assembler.Consider the assembler has 2 passes, 2 pass means at any time it will be doing only one thing, either the 1st pass or the 2nd pass.Which means it will finish 1st pass first and then 2nd pass.Let assume that available main memory size is 150KB and total code size is 200KB

As the total code size is 200KB and main memory size is 150KB, it is not possible to use 2 passes together.So, in this case, we should go with the overlays technique.According to the overlays concept at any time only one pass will be used and both the passes always need symbol table and common routine.Now the question is if overlays-driver\* is 10KB, then what is the minimum partition size required?For pass 1 total memory needed is = (70KB + 30KB + 20KB + 10KB) = 130KB and for pass 2 total memory needed is = (80KB + 30KB + 20KB + 10KB) = 140KB.So if we have minimum 140KB size partition then we can run this code very easily.

1. **What are the advantages and disadvantages of contiguous and non-contiguous memory allocation?**

**Ans. Advantages:**

1) This Memory allocation provides the direct and easy access.  
2) Number of disk required in type of memory allocation is reduced to minimum  
3) In case of contiguous memory allocation the good performance remains a positive factor.  
  
**Disadvantages:**1) For new files it is very difficult to find the spaces here.  
2)Further more you can't extend the file .  
3)The one big disadvantage is the difficulty about fragmentation.

**Assignment 3**

1. **Differentiate between Global and Local Page Replacement Algorithm.**

**Ans.** Global replacement allows a process to select a replacement frame from the set of all frames, even if that frame is currently allocated to some other process; that is, one process can take a frame from another. Local replacement requires that each process select from only its own set of allocated frames.

In local replacement strategy, the number of frames allocated to a process does not change. With global replacement, a process may happen to select only frames allocated to other processes, thus increasing the number of frames allocated to it (assuming that other processes do not choose its frames for replacement).

1. **Differentiate between demand paging and pure demand paging.**

**Ans.** Demand paging follows that pages should only be brought into memory if the executing process demands them. This is often referred to as lazy evaluation as only those pages demanded by the process are swapped from secondary storage to main memory. Contrast this to pure swapping, where all memory for a process is swapped from secondary storage to main memory during the process startup.

**Assignment 4**

1. **What is swapping and what is its purpose?**

**Ans.** Swapping is a mechanism in which a process can be swapped temporarily out of main memory (or move) to secondary storage (disk) and make that memory available to other processes.

Memory swapping makes use of storage capacity to create a swap file to extend available system resources. It is a computer technology that enables an operating system to provide more memory to a running application or process than is available in physical random access memory (RAM)

1. **Differentiate between Co-operating and independent process.**

**Ans.** A process is independent if it cannot affect other other process or be affected by it. Any process that does not share data with others is independent. Otherwise the process is cooperating. Cooperation is done to provide information sharing, computational speedups, modularity and convenience.

**Assignment 5**

* + - 1. **Distinguish between logical and physical address.**

**Ans.** The basic difference between Logical and physical address is that Logical address is generated by CPU in perspective of a program whereas the physical address is a location that exists in the memory unit. The logical address does not exist physically in the memory whereas physical address is a location in the memory that can be accessed physically.

**2**. **Explain multilevel and multilevel feedback queue scheduling?**

**Ans.**  A Multi-Level Queue scheduling algorithm partitions the ready queue into several separate queues. The processes are permanently assigned to one queue.

Multilevel Feedback Queue Scheduling. In a multilevel queue-scheduling algorithm, processes are permanently assigned to a queue on entry to the system. Multilevel feedback queue scheduling, however, allows a process to move between queues. The idea is to separate processes with different CPU-burst characteristics.

**Assignment 6**

1. **What do you mean by synchronization? What are the different tools used to achieve synchronization?**

**Ans.** Process Synchronization means sharing system resources by processes in a such a way that, Concurrent access to shared data is handled thereby minimizing the chance of inconsistent data. Maintaining data consistency demands mechanisms to ensure synchronized execution of cooperating processes.

1. **Differentiate between critical region, Monitor and Semaphore.**

**Ans.** Critical region is the code region wherein the shared variables can be accessed. Critical section contains shared variables or resources which are needed to be synchronized to maintain consistency of data variable.

A Semaphore is a lower-level object. A semaphore is a non-negative integer variable. The value of Semaphore indicates the number of shared resources available in the system. The value of semaphore can be modified only by two functions, namely wait() and signal() operations (apart from the initialization).

A Monitor type high-level synchronization construct. It is an abstract data type. The Monitor type contains shared variables and the set of procedures that operate on the shared variable.

**Assignment 8**

1. **Why rotational latency is usually not considered in disk scheduling?**

**Ans.**  Most disks do not export their rotational position information to the host. Even if they did, the time for this in order to reach the scheduler would be subject to imprecision and the time consumed by the scheduler is variable, so the rotational position in order would become incorrect. Further, the disk requests are usually given in terms of logical block numbers, and the mapping among logical blocks and physical locations is very difficult

1. **What is low level and high level formatting?**

**Ans.**

**High level formatting** can be performed on a hard disk only after the sectors and tracks have been formatted using a low level format. This type of formatting is where actual data, like the operating system is written onto the drive sectors to create file and system structures so you can use the drive to install other applications and store data and files.

**Low level formatting** is used to initiate a hard drive and prepare it for data by creating the actual sectors and tracks on the drive, as well as the control structures needed to read and write data on the drive. A low level format is often completed by the manufacturer of the hard drive, for instance, in cases where you purchased new hard drives to install into existing computers. In some cases, however, you may need to run a low level format after installing the drive into the computer before you can install an actual operating system.

**Suggested Question Bank**

**Preparatory Questions**

1. Which module gives control of the CPU to the process selected by the short-term scheduler?
   1. **dispatcher**
   2. interrupt
   3. scheduler
   4. none of the mentioned
2. The processes that are residing in main memory and are ready and waiting to execute are kept on a list called
   1. job queue
   2. **ready queue**
   3. execution queue
   4. process queue
3. The interval from the time of submission of a process to the time of completion is termed as
   1. waiting time
   2. **turnaround time**
   3. response time
   4. throughput
4. Which scheduling algorithm allocates the CPU first to the process that requests the CPU first?
   1. **first-come, first-served scheduling**
   2. shortest job scheduling
   3. priority scheduling
   4. none of the mentioned
5. In priority scheduling algorithm
   1. **CPU is allocated to the process with highest priority**
   2. CPU is allocated to the process with lowest priority
   3. equal priority processes can not be scheduled
   4. none of the mentioned
6. Process are classified into different groups in
   1. shortest job scheduling algorithm
   2. round robin scheduling algorithm
   3. priority scheduling algorithm
   4. **multilevel queue scheduling algorithm**
7. Which one of the following can not be scheduled by the kernel?
   1. kernel level thread
   2. **user level thread**
   3. process
   4. none of the mentioned
8. CPU scheduling is the basis of \_\_\_\_\_\_\_\_\_\_\_\_.
   1. multiprocessor systems
   2. **multiprogramming operating systems**
   3. larger memory sized systems
   4. None of these
9. With multiprogramming, \_\_\_\_\_\_ is used productively.

**a) time**  b) space c) money d) All of these

1. The two steps of a process execution are : (choose two)

**a) I/O Burst b) CPU Burst** c) Memory Burst d) OS Burst

1. An I/O bound program will typically have :
   1. a few very short CPU bursts
   2. many very short I/O bursts
   3. **many very short CPU bursts**
   4. a few very short I/O bursts
2. A process is selected from the \_\_\_\_\_\_ queue by the \_\_\_\_\_\_\_\_ scheduler, to be executed.

a) blocked, short term b) wait, long term **c) ready, short term** d) ready, long term

1. In the following cases non – preemptive scheduling occurs : (Choose two)
   1. When a process switches from the running state to the ready state
   2. **When a process goes from the running state to the waiting state**
   3. **When a process switches from the waiting state to the ready state**
   4. When a process terminates
2. Dispatch latency is :
   1. the speed of dispatching a process from running to the ready state
   2. the time of dispatching a process from running to ready state and keeping the CPU idle
   3. **the time to stop one process and start running another one**
   4. None of these
3. Scheduling is done so as to :
   1. **increase the throughput**
   2. decrease the throughput
   3. increase the duration of a specific amount of work
   4. None of these
4. Turnaround time is :
   1. the total waiting time for a process to finish execution
   2. the total time spent in the ready queue
   3. the total time spent in the running queue
   4. **the total time from the completion till the submission of a process**
5. Scheduling is done so as to :
   1. increase the turnaround time
   2. **decrease the turnaround time**
   3. keep the turnaround time same
   4. there is no relation between scheduling and turnaround time

**Experiment No: (Mini Project)**

Student Name and Roll Number:

Semester /Section:

Link to Code:

Date:

Faculty Signature:

Remarks:

**Project Title:**

**Description of Project:**

**Problem Statement:**

**Problem Analysis:**

**Program Design:**

**Programming Requirements:**

**Data/Input Output Description:**

**Algorithmic Approach/Algorithm/DFD/ER diagram/Program Steps**

**Implementation and Testing (stage/module wise)**

**Output (Screenshots)**