

FAZAIA BILQUIS COLLEGE OF EDUCATION
FOR WOMEN PAF BASE NUR KHAN



COMPUTER SCIENCE DEPARTMENT

OPERATING SYSTEM

LAB MANUAL

WEEK # 1	1
1. Introduction to Operating System.....	1
1.2. Introduction to Ubuntu	1
1.3. Installation of Ubuntu.....	1
Steps to Install Ubuntu on Virtual Box.....	1
1.3.1. Before we begin with the installation process, we need to Download ISO for Ubuntu. For that, all the versions of Ubuntu are available on the Official Site.....	1
1.3.2. Open Virtual Box and click on the New button.	2
1.3.3. Give a Name to your Virtual Machine and select the Location for it to install... 	2
1.3.4. Assign RAM Size to your Virtual Machine.	3
1.3.5. Create a Virtual Hard Disk for the machine to store files.	3
1.3.6. Select the type of Hard disk. Using VDI type is recommended.	4
1.3.7. Either of the Physical Storage types can be selected. Using a Dynamically Allocated Disk is by default recommended.....	4
1.3.8. Select Disk Size and provide the Destination Folder to install.....	5
1.3.9. After the Disk creation is done, boot the Virtual Machine and begin installing Ubuntu.....	5
1.3.10. If the installation disk is not automatically detected. Browse the file location and select the ISO file for Ubuntu.	6
1.3.11. Proceed with the installation file and wait for further options.	6
1.3.12. Click on the Install Ubuntu option, this might look different for other Ubuntu versions. 7	
1.3.13. Select Keyboard Layout, if the defaults are compatible, just click on the Continue button and proceed.	7
1.3.14. Wait for the installation process to complete.	8
1.3.15. Once the installation process is over, reboot your Virtual Machine.	8
1.3.16. You're finished with the installation process. Now you can use Ubuntu along with Windows, without creating a dual boot.	9
2. COMMAND PROMPT	10
2.1. COMMAND'S EXECUTION.....	10
Files Command.....	10
2.1.1. Title aafreen.....	10
2.1.2. date	11
2.1.3. Time.....	11
2.1.4. Hostname	11
2.1.5. cls	12
2.1.6. Color fb	12
2.1.7. Color 0a.....	12

2.1.8.	Ver	13
2.1.9.	Cd	13
2.1.10.	dir	13
2.1.11.	find/?.....	14
2.1.12.	runas.....	14
2.1.13.	cd download.....	14
2.1.14.	Sort	15
2.1.15.	Taskkill.....	15
2.1.16.	Tasklist	15
2.1.17.	w32tm	16
2.1.18.	Start.....	16
2.1.19.	Timeout	16
2.1.20.	Pause.....	17
2.1.21.	Exit	17
2.1.22.	shutdown.....	17
2.1.23.	setfont	18
2.1.24.	ftp.....	18
2.1.25.	ftype.....	18
2.1.26.	getmac	19
2.1.27.	Ipconfig	19
2.1.28.	Ping.....	20
2.1.29.	route	20
2.1.30.	systeminfo	21
2.1.31.	netstat	22
2.1.32.	telnet	22
2.1.33.	netview	22
2.1.34.	Arp.....	23
2.1.35.	nslookup	23
2.1.36.	echo.....	24
2.1.37.	attrib.....	24
2.1.38.	compact	25
2.1.39.	copy.....	25
2.1.40.	mkdir (or md).....	26
2.1.41.	rename (or ren).....	26
2.1.42.	rmdir (or rd).....	27
2.1.43.	tree.....	28

2.1.44.	type	29
2.1.45.	chkdsk	29
2.1.46.	chkntfs	29
2.1.47.	defrag	29
2.1.48.	diskpart	30
2.1.49.	format	30
2.1.50.	label	30
2.1.51.	mode	30
2.1.52.	mountvol	31
2.1.53.	verify	31
2.1.54.	vol	31
2.1.55.	for	31
2.1.56.	gpupdate	32
2.1.57.	gpresult	32
2.1.58.	prompt	33
2.1.59.	tracert	34
WEEK # 2	35
3.	UBUNTU	35
3.1.	Files Command	36
3.1.1.	ls	36
3.1.2.	mkdir	36
3.1.3.	rmdir	36
3.1.4.	cd	37
3.1.5.	sudo -i	37
3.1.6.	sudo apt install ncal	37
3.1.7.	cal	38
3.1.8.	clear	40
3.1.9.	ncal -M	40
3.1.10.	cal 9 1752	40
3.1.11.	time	40
3.1.12.	date	41
3.1.13.	hostname	41
3.1.14.	sudo snap install chromium	41
3.1.15.	who	41
3.1.16.	History	42
3.1.17.	sudo snap install vlc	42

3.1.18. cd /	43
3.1.19. cat >> bar.txt	43
3.1.20. Permissions to make a file readable,writeable ,executable.....	43
3.1.21. File	44
3.1.22. Head	45
3.1.23. Which	45
3.1.24. Whereis	45
3.1.25. Locate	45
3.1.26. Find.....	46
3.1.27. Ps.....	46
3.1.28. Tty.....	46
3.1.29. ps aux.....	47
3.1.30. Kill.....	47
3.1.31. Echo.....	48
3.1.32. Rm	48
3.1.33. Tail.....	48
3.1.34. Sort	49
3.1.35. Grep.....	49
3.1.36. Less	49
3.1.37. Man.....	50
3.1.38. Output Redirection	50
3.1.39. ls -l.....	51
3.1.40. Pwd	51
3.1.41. touch hello.cpp.....	51
3.1.42. g++ hello.cpp -o test	52
3.1.43. ./test	52
WEEK # 3.....	53
4. CODING IN UBUNTU WITH C++ PROGRAMMING LANGUAGE.....	53
4.1. SCENERIOS:	53
4.1.1. You are tasked with developing an interactive calculator that evaluates mathematical expressions while adhering to the BODMAS rule.	53
4.1.2. You are a teacher who wants to analyze the scores of students in a recent exam. You have collected the scores from all students, and now you want to find out: The highest score in the class (maximum).The lowest score in the class (minimum).	54
4.1.3. You are a data analyst working with a list of integers representing the daily sales of a product over a month. You need to perform the following tasks:.....	56
Calculate the sum of all even numbers in the list.....	56

Calculate the result of subtracting all odd numbers in the list from an initial value of zero.....	56
WEEK # 4	58
5. CODING IN UBUNTU TERMINAL	58
5.1.1. DECLRAING VARIABLE	58
5.1.2. DIPLAYING THE VARIBALE VALUE	58
5.1.3. DECLEARING A STRING	58
5.1.4. DIPLAYING A STRING.....	58
5.1.5. DECLEARING AN INTEGER	58
5.1.6. DIPLAYING AN INTEGER.....	59
5.1.7. IF-THEN-ELSE STATEMENT	59
5.1.8. CASE STATEMENT	59
5.1.9. FOR LOOP.....	60
5.1.10. DO-WHILE LOOP	60
5.1.11. WHILE LOOP	61
5.1.12. FILE COMPARING.....	61
5.1.13. VARIABLE COMPARING	62
5.1.14. LOGICAL OPERATORS.....	62
WEEK # 5	63
6. SYSTEM CALLS.....	63
6.1. fork ().....	63
6.1.1. SIMPLE PROGRAM	63
6.1.2. Pedict the Output of the following program:	64
6.1.2.1. Program	64
6.1.2.2. Program	65
6.1.2.3. Program	66
6.1.2.4. Program	67
6.1.2.5. Write a program to find sum of even numbers in parent process and sum of odd numbers in child process.....	68
6.2. Exit().....	70
6.3. Wait()	70
6.4. Sleep().....	70
WEEK # 6	72
7. FIRST COME FIRST SERVED (FCFS).....	72
7.1. FCFS WITH SAME ARRIVAL TIME i.e. 0.....	72
7.2. FCFS WITH DIFFERENT ARRIVAL TIME	74

8. SHORTEST JOB FIRST (SJF)	77
8.1. SJF WITH SAME ARRIVAL TIME i.e. 0	77
8.2. SJF WITH DIFFERENT ARRIVAL TIME.....	80
WEEK # 7.....	83
9. PRIORITY SCHEDULLING (NON-PREEMPTIVE)	83
9.1. PS WITH DIFFERENT ARRIVAL TIME.....	83
9.2. PS WITH SAME ARRIVAL TIME i.e. 0	86
10. SHORTEST REMAINING TIME FIRST (PREEMPTIVE) (SRTF)	89
11. ROUND ROBIN (PREEMPTIVE)	92
11.1. RR WITH SAME ARRIVAL TIME.....	92
11.2. RR WITH DIFFERENT ARRIVAL TIME	95
WEEK # 8.....	98
12. DEADLOCK DETECTION.....	98
12.1. SHOW IF THE SYSTEM IS IN DEADLOCK OR NOT?	98
12.2. SHOW THE SYSTEM IS IN SAFE STATE AND PROCESS THE SEQUENCE OF THE PROCESS?.....	102
WEEK # 9.....	106
13. THREADS	106
13.1. Write a C++ program to demonstrate basic single threading.....	106
13.2. Write a C++ program to demonstrate basic multi-threading.....	107
14. POSIX-THREAD	109
14.1. Write a C++ program to demonstrate basic single threading using P-Thread	109
14.2. Write a C++ program to demonstrate basic multi-threading using P-Thread.....	110
WEEK # 10	112
15. PAGE REPLACEMENT ALGORITHM.....	112
15.1. FIFO (FIRST IN FIRST OUT)	112
15.2. LRU (LEAST REACENTLY USED)	115
15.3. OPTIMAL PAGE REPLACEMENT.....	118
WEEK # 11	121
16. FRAGMENTATION.....	121
16.1. FIRST-FIT	121
16.2. WORST-FIT	122
16.3. BEST-FIT	124
WEEK # 12	127
17. IPC(INTER PROCESS COMMUNICATION)	127
17.1. WRITER PROCESS.....	127

17.2. READER PROCESS	129
WEEK # 13	131
18. DINING PHILOSOPHER PROBLEM	131
WEEK # 14	135
19. ZOMBIE PROCESS.....	135
20. ORPHAN PROCESS.....	137

WEEK # 1

1. Introduction to Operating System

An operating system (OS) is the fundamental software that manages computer hardware and software resources, providing essential services to both users and applications. It acts as an intermediary between hardware and user-level programs, enabling communication and resource management. Core functions of an OS include managing memory, processing tasks, handling input/output (I/O) operations, and ensuring security. Popular operating systems like Windows, macOS, Linux, and Android allow users to interact with computers efficiently, offering user-friendly interfaces while managing the underlying complexities of hardware resources.

1.2. Introduction to Ubuntu

Ubuntu is a popular open-source operating system based on the Linux kernel, known for its user-friendly interface and robust security features. Developed by Canonical, Ubuntu is designed to be accessible to both new and experienced users, offering a seamless experience for everyday computing tasks, programming, and server management. It comes pre-installed with essential software like LibreOffice, Firefox, and a Software Center for easy installation of additional applications. Ubuntu's regular updates, strong community support, and extensive documentation make it a preferred choice for developers, system administrators, and casual users alike. Available in both desktop and server versions, it is widely used for personal computers, cloud environments, and various enterprise solutions.

1.3. Installation of Ubuntu

Steps to Install Ubuntu on Virtual Box

1.3.1. Before we begin with the installation process, we need to Download ISO for Ubuntu. For that, all the versions of Ubuntu are available on the Official Site.

Ubuntu 18.04.3 LTS

Download the latest [LTS](#) version of Ubuntu, for desktop PCs and laptops. LTS stands for long-term support — which means five years, until April 2023, of free security and maintenance updates, guaranteed.

[Ubuntu 18.04 LTS release notes](#)

Recommended system requirements:

- ✓ 2 GHz dual core processor or better
- ✓ 4 GB system memory
- ✓ 25 GB of free hard drive space
- ✓ Either a DVD drive or a USB port for the installer media
- ✓ Internet access is helpful

[Download](#)

For other versions of Ubuntu Desktop including torrents, the network installer, a list of local mirrors, and past releases [see our alternative downloads](#).

Ubuntu 19.10

The latest version of the Ubuntu operating system for desktop PCs and laptops, Ubuntu 19.10 comes with nine months, until July 2020, of security and maintenance updates.

Recommended system requirements are the same as for Ubuntu 18.04.3 LTS.

[Ubuntu 19.10 release notes](#)

[Download](#)

[Alternative downloads and torrents](#)

Figure 1.3.1

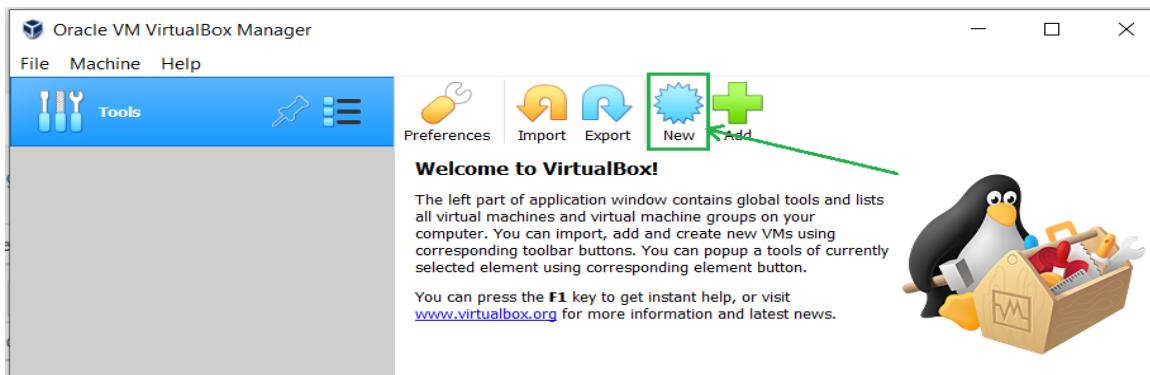
1.3.2. Open Virtual Box and click on the New button.

Figure 1.3.2

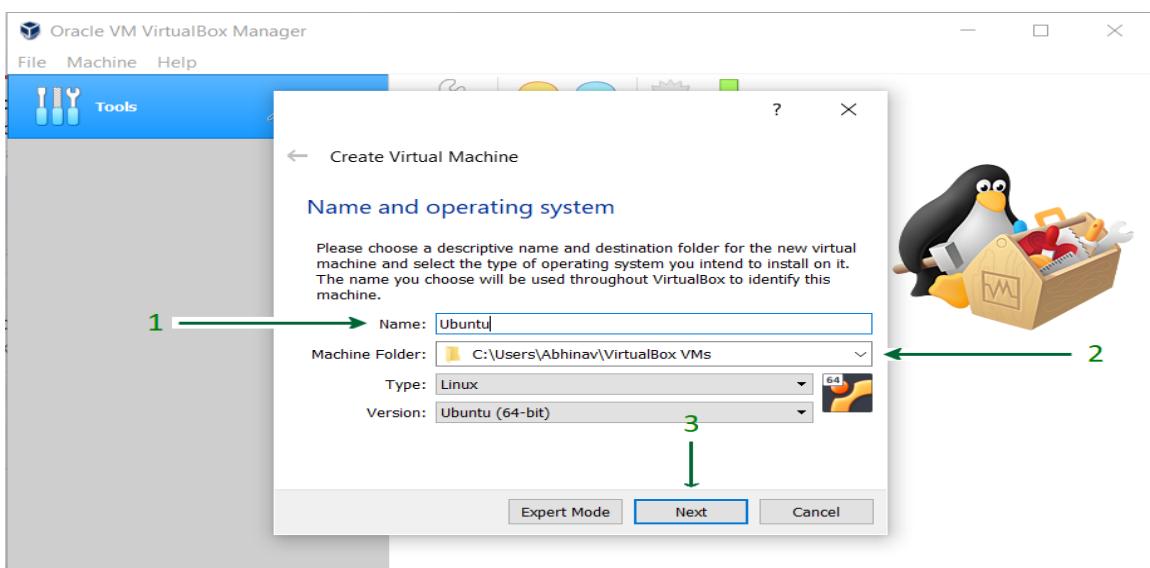
1.3.3. Give a Name to your Virtual Machine and select the Location for it to install.

Figure 1.3.3

1.3.4. Assign RAM Size to your Virtual Machine.

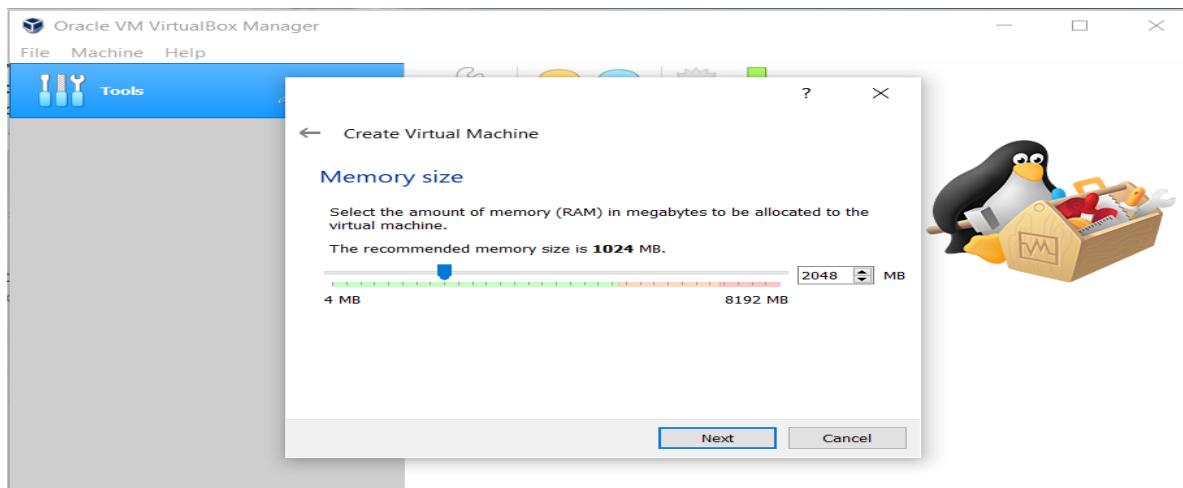


Figure 1.3.4

1.3.5. Create a Virtual Hard Disk for the machine to store files.

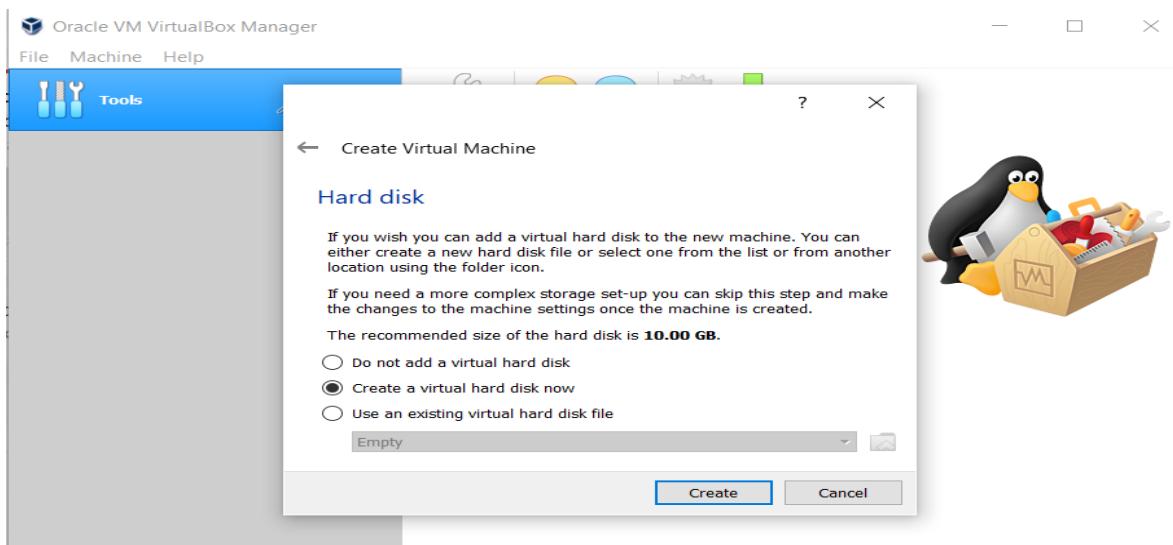


Figure 1.3.5

1.3.6. Select the type of Hard disk. Using VDI type is recommended.

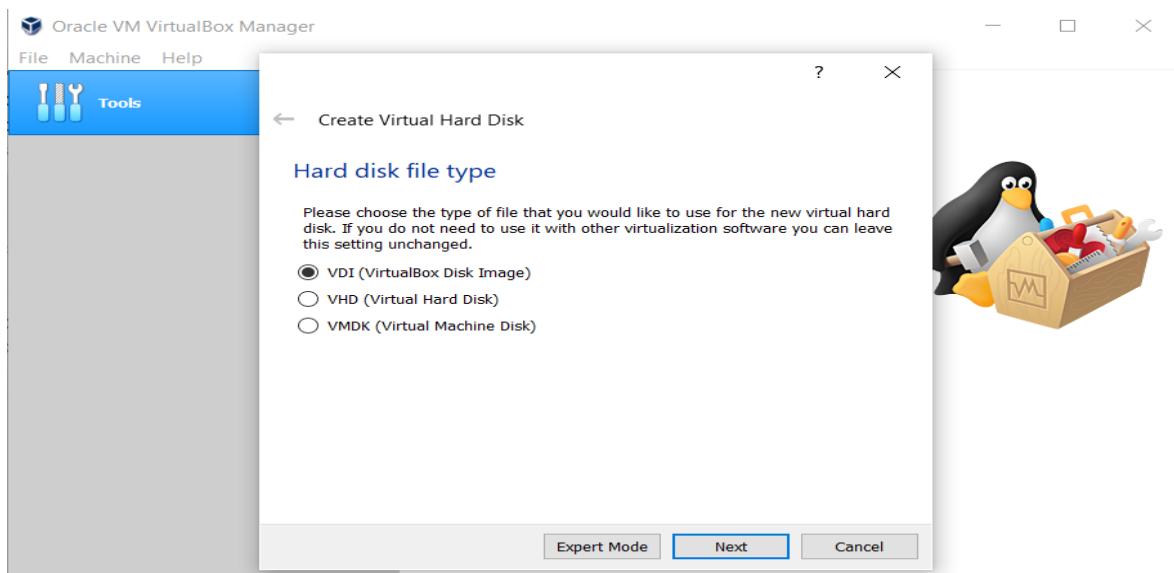


Figure 1.3.6

1.3.7. Either of the Physical Storage types can be selected. Using a Dynamically Allocated Disk is by default recommended.

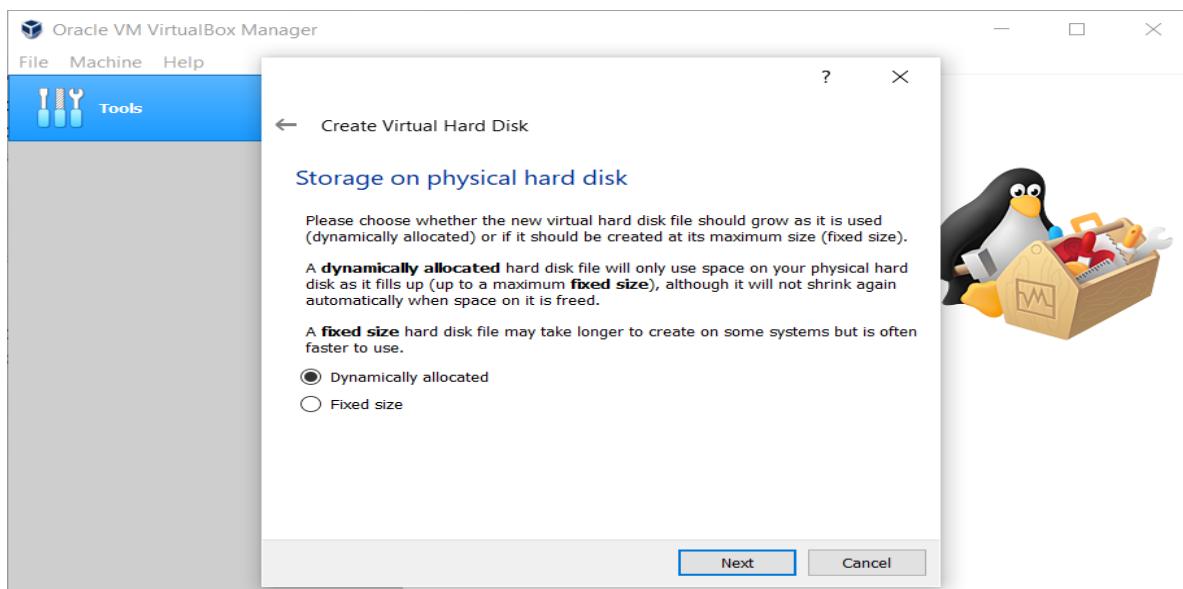


Figure 1.3.7

1.3.8. Select Disk Size and provide the Destination Folder to install.

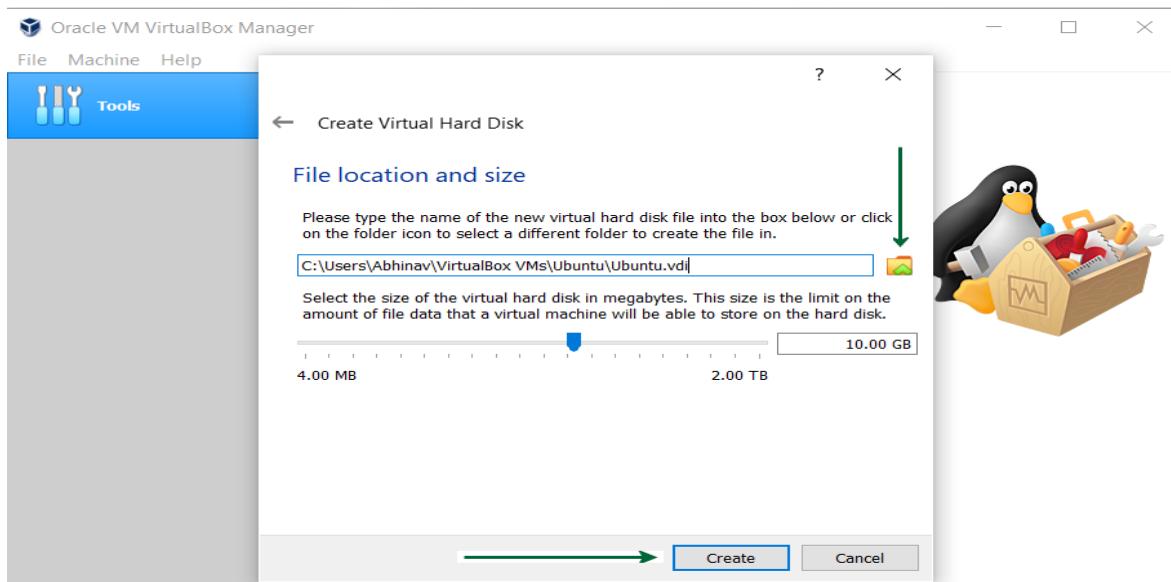


Figure 1.3.8

1.3.9. After the Disk creation is done, boot the Virtual Machine and begin installing Ubuntu.

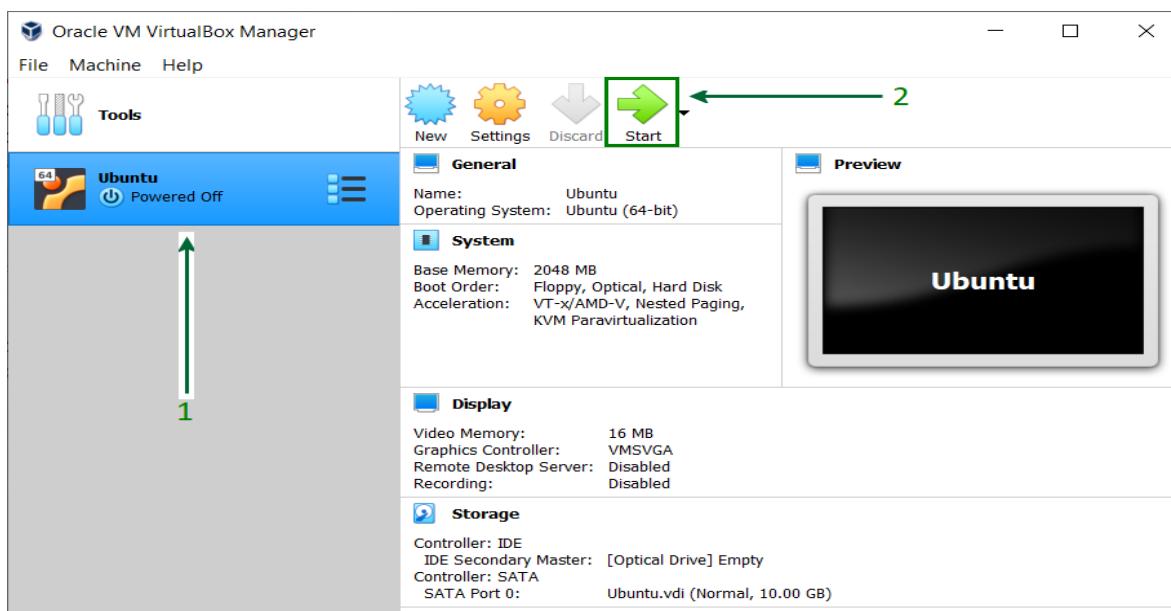


Figure 1.3.9

1.3.10. If the installation disk is not automatically detected. Browse the file location and select the ISO file for Ubuntu.

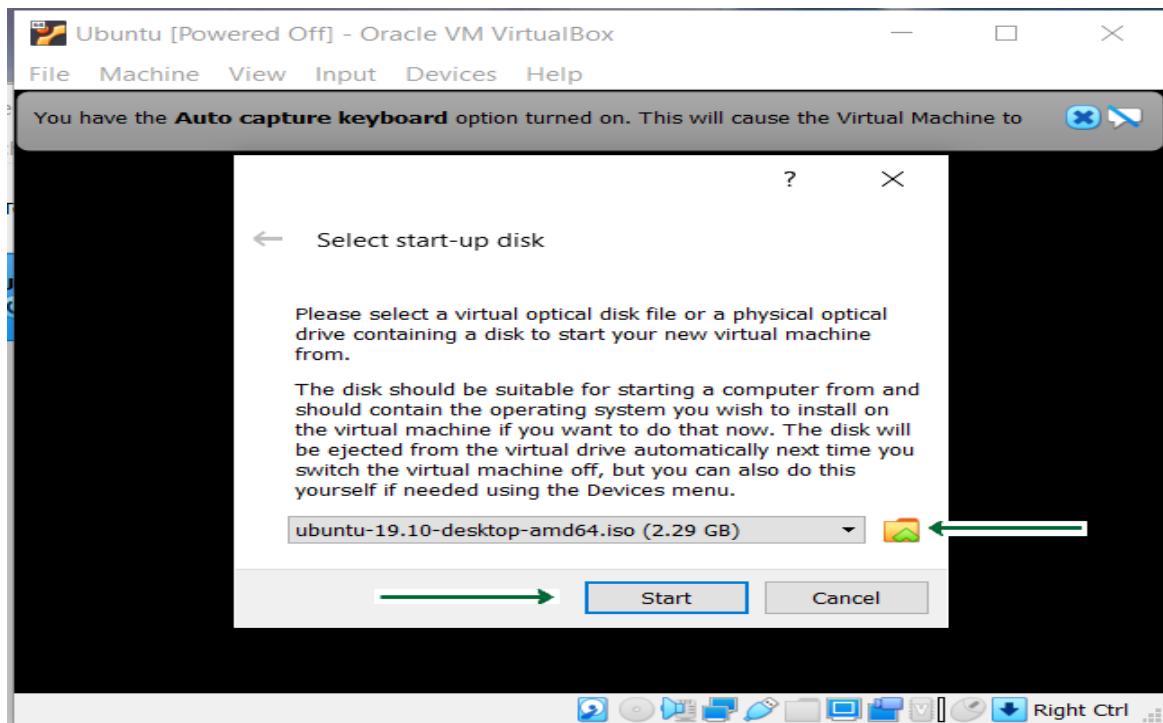


Figure 1.3.10

1.3.11. Proceed with the installation file and wait for further options.

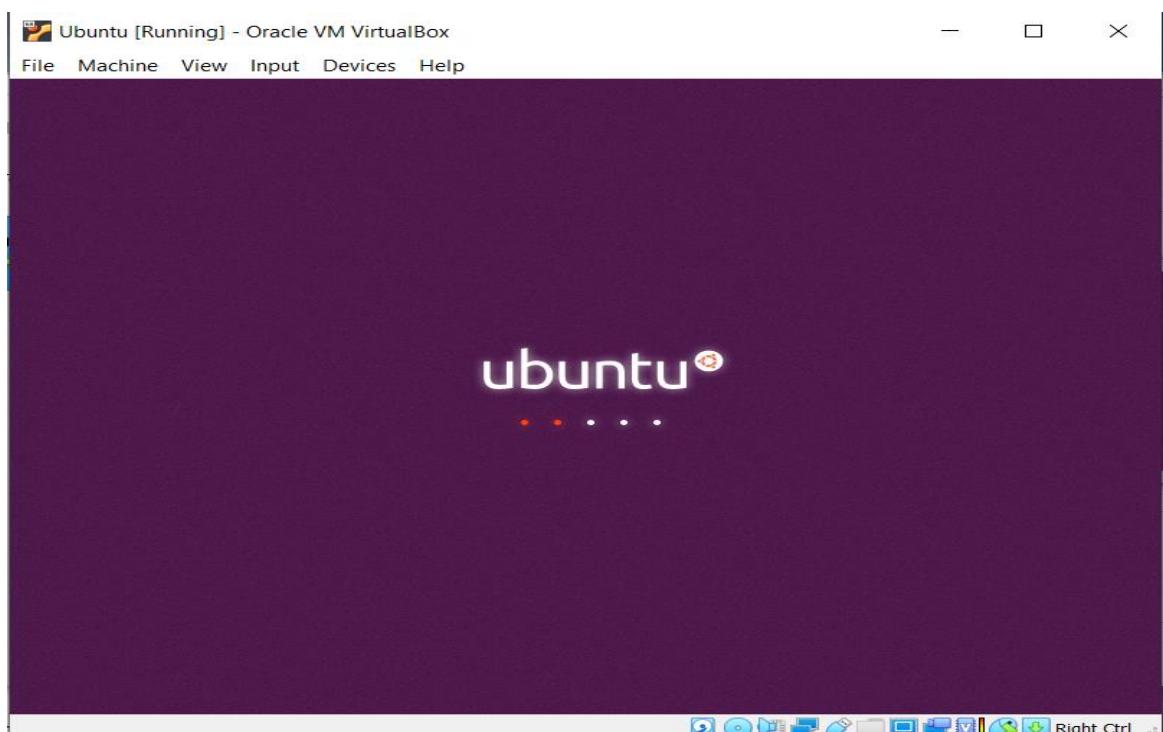


Figure 1.3.11

1.3.12. Click on the Install Ubuntu option, this might look different for other Ubuntu versions.

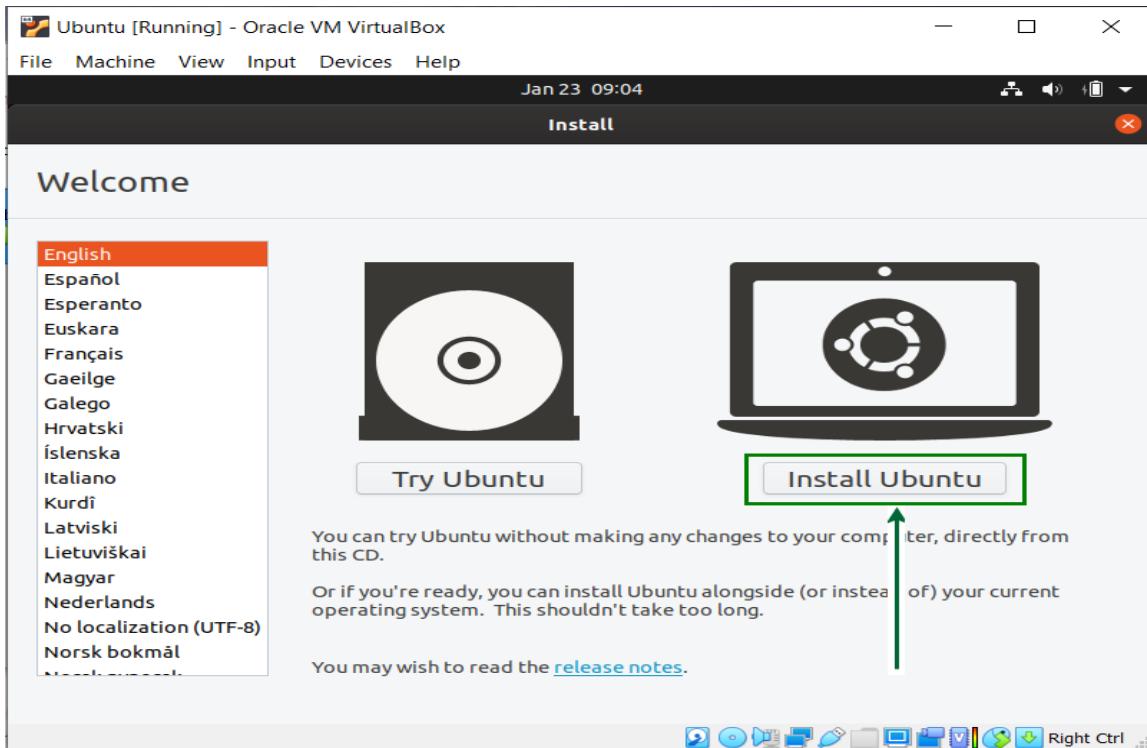


Figure 1.3.12

1.3.13. Select Keyboard Layout, if the defaults are compatible, just click on the Continue button and proceed.

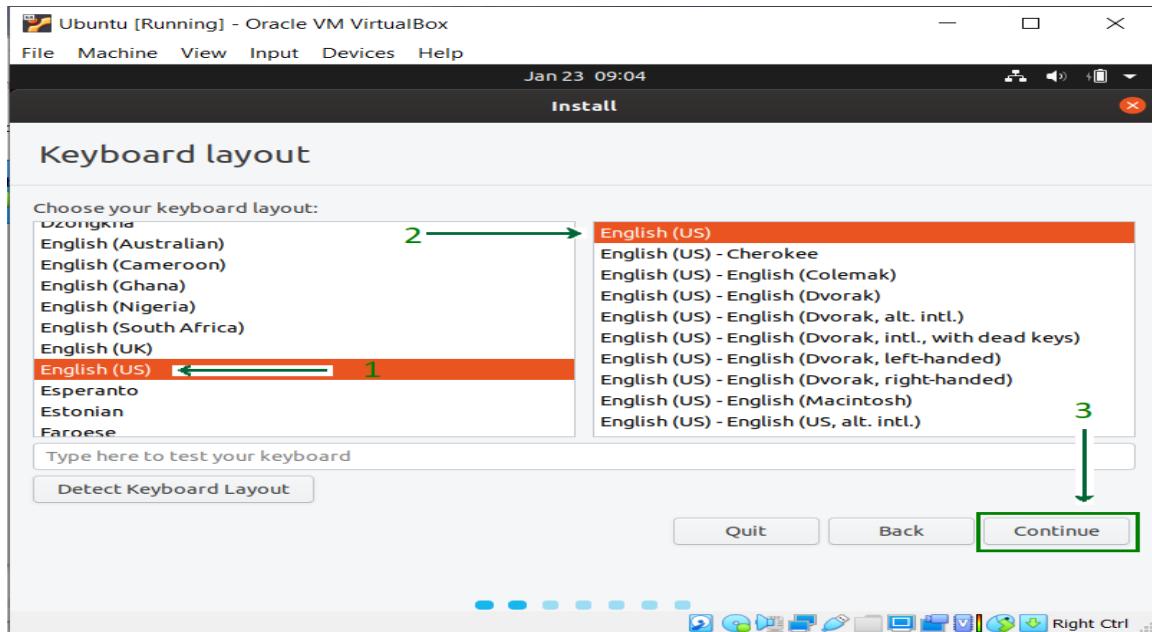


Figure 1.3.13

1.3.14. Wait for the installation process to complete.

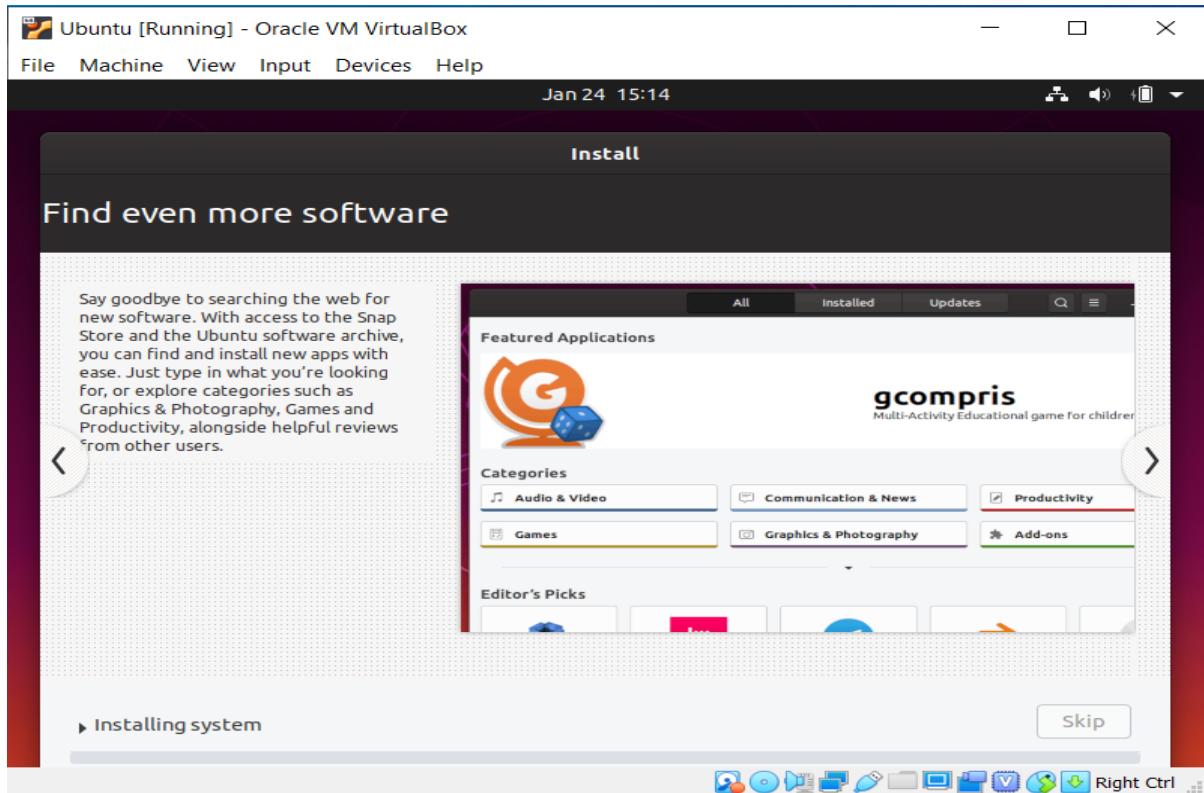


Figure 1.3.14

1.3.15. Once the installation process is over, reboot your Virtual Machine.

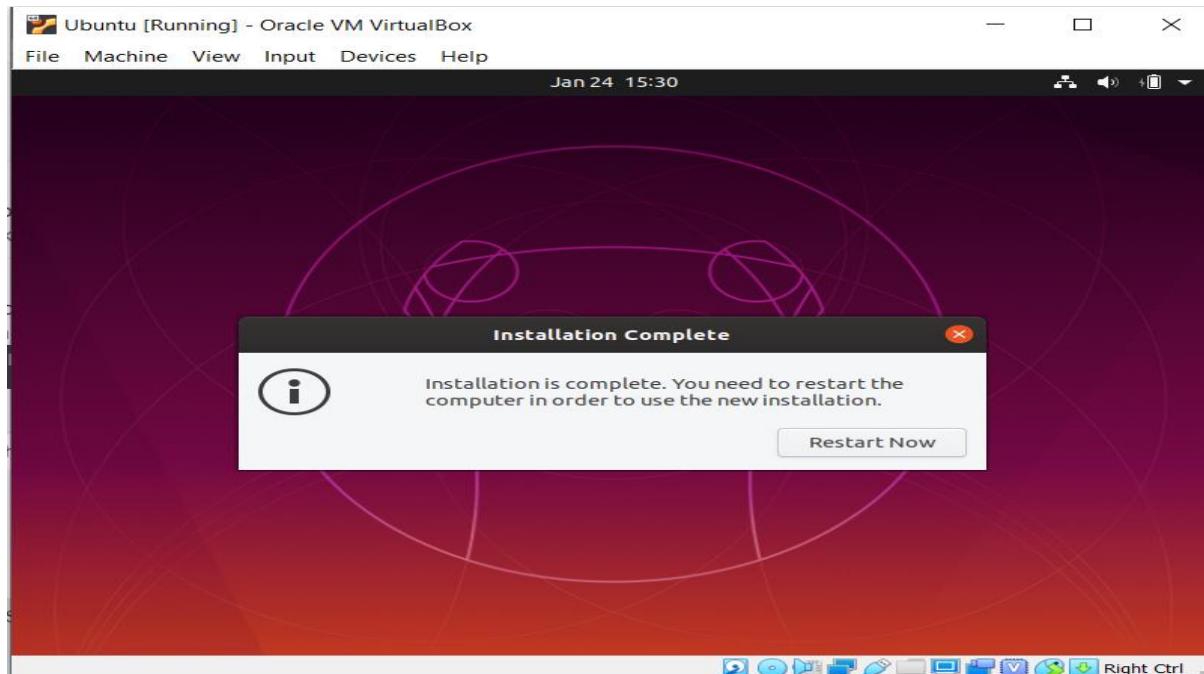


Figure 1.3.15

1.3.16. You're finished with the installation process. Now you can use Ubuntu along with Windows, without creating a dual boot.

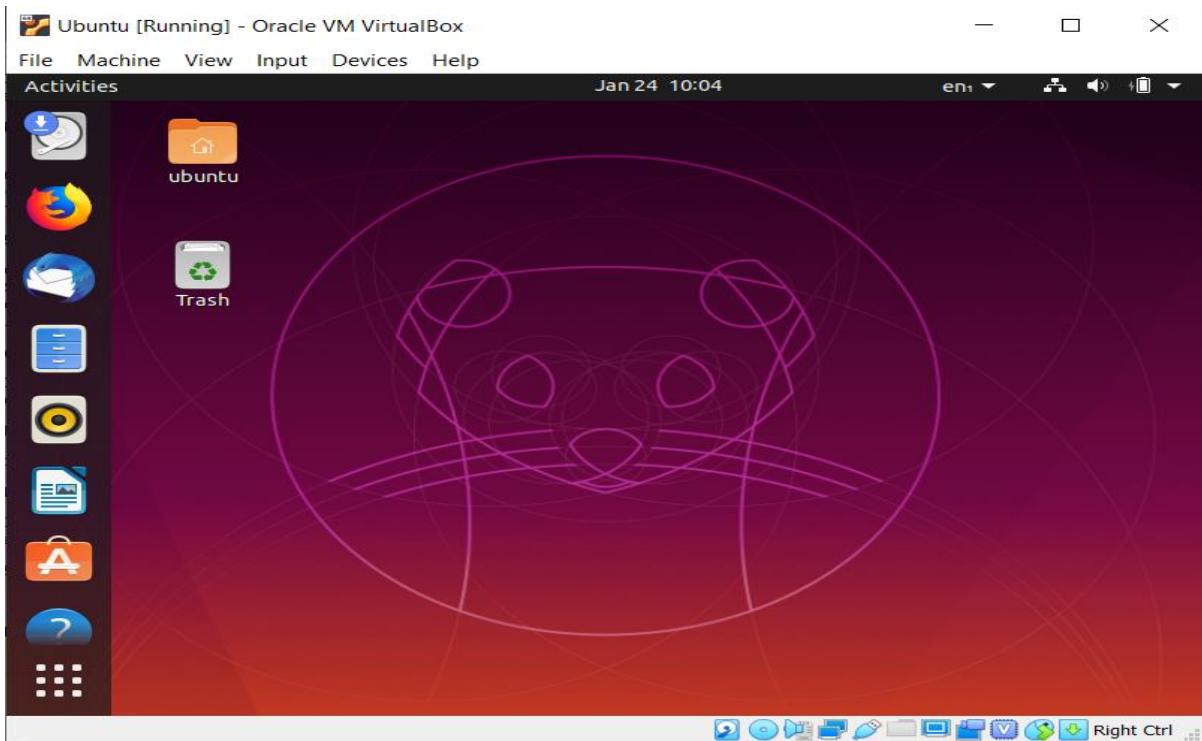
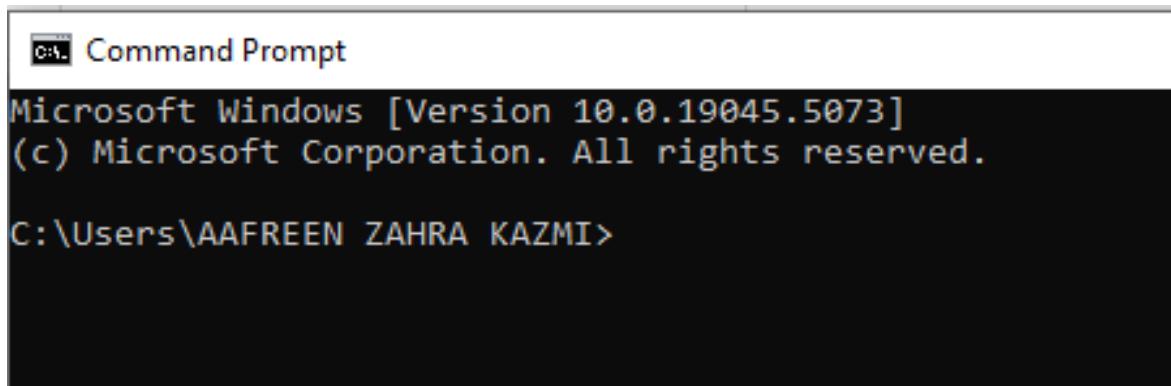


Figure 1.3.16

2. COMMAND PROMPT

The Command Prompt (CMD) in Windows is a command-line interpreter application available in most Windows operating systems. It is used to execute entered commands, which allows users to perform various tasks on the computer without needing a graphical user interface.



```
Command Prompt
Microsoft Windows [Version 10.0.19045.5073]
(c) Microsoft Corporation. All rights reserved.

C:\Users\AAFREEN ZAHRA KAZMI>
```

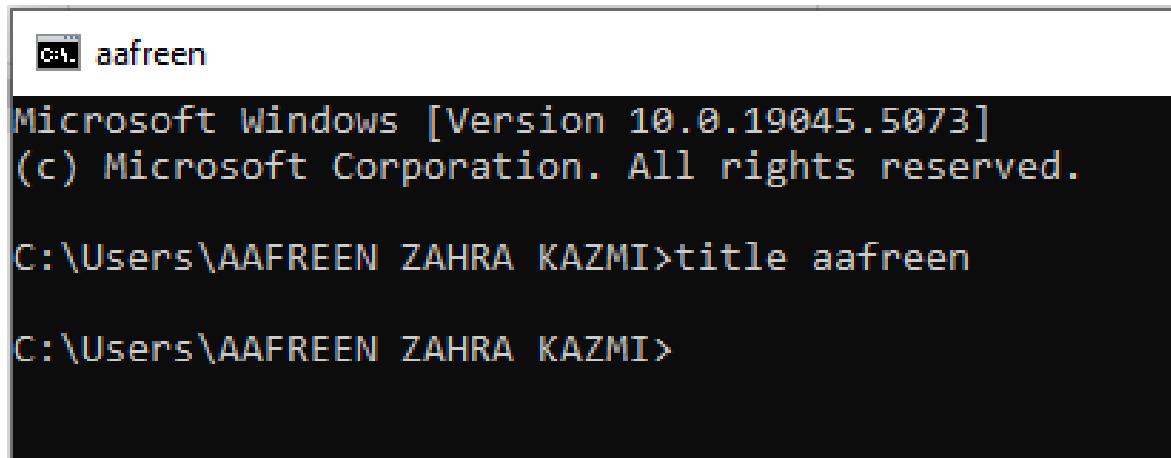
Figure 2

2.1. COMMAND'S EXECUTION

Files Command

2.1.1. Title aafreen

- This command sets the title of the Command Prompt window to the specified text. It's useful for customizing the Command Prompt window for specific tasks.



```
aafreen
Microsoft Windows [Version 10.0.19045.5073]
(c) Microsoft Corporation. All rights reserved.

C:\Users\AAFREEN ZAHRA KAZMI>title aafreen

C:\Users\AAFREEN ZAHRA KAZMI>
```

Figure 2.1.1

2.1.2. date

- Displays the current system date and allows the user to change it if necessary. When executed without options, it prompts for a new date.

```
C:\Users\AAFREEN ZAHRA KAZMI>date  
The current date is: Thu 10/24/2024  
Enter the new date: (mm-dd-yy)
```

```
C:\Users\AAFREEN ZAHRA KAZMI>
```

Figure 2.1.2

2.1.3. Time

- Displays or sets the system time.

```
C:\Users\AAFREEN ZAHRA KAZMI>time  
The current time is: 15:39:48.18  
Enter the new time:
```

```
C:\Users\AAFREEN ZAHRA KAZMI>
```

Figure 2.1.3

2.1.4. Hostname

- Displays the name of the computer

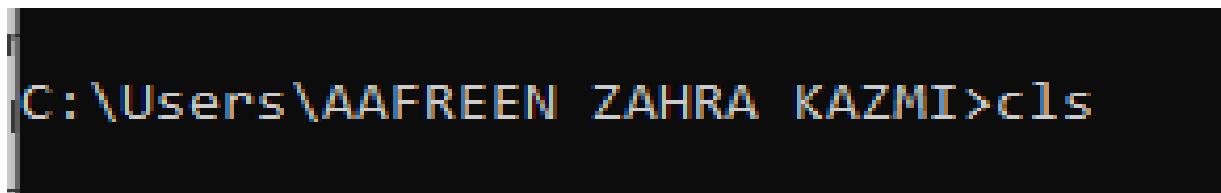
```
C:\Users\AAFREEN ZAHRA KAZMI>hostname  
DESKTOP-46UPK5M
```

```
C:\Users\AAFREEN ZAHRA KAZMI>
```

Figure 2.1.4

2.1.5. cls

- Clears the command prompt screen.



```
C:\Users\AAFREEN ZAHRA KAZMI>cls
```

*Figure 2.1.5***2.1.6. Color fb**

- Changes the text and background colors in CMD



```
C:\Users\aafreeen

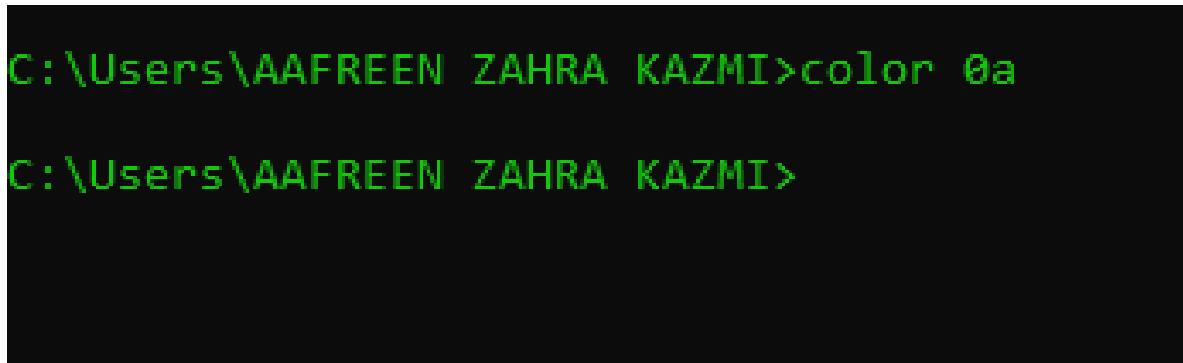
C:\Users\AAFREEN ZAHRA KAZMI>colorfb
'colorfb' is not recognized as an internal or external command,
operable program or batch file.

C:\Users\AAFREEN ZAHRA KAZMI>color fb

C:\Users\AAFREEN ZAHRA KAZMI>
```

*Figure 2.1.6***2.1.7. Color 0a**

- Changes the text and background colors



```
C:\Users\AAFREEN ZAHRA KAZMI>color 0a

C:\Users\AAFREEN ZAHRA KAZMI>
```

Figure 2.1.7

2.1.8. Ver

- Displays the version of Windows.

```
C:\Users\AAFREEN ZAHRA KAZMI>ver
Microsoft Windows [Version 10.0.19045.5073]
```

Figure 2.1.8

2.1.9. Cd

- Display the current directory.

```
C:\Users\AAFREEN ZAHRA KAZMI>cd
C:\Users\AAFREEN ZAHRA KAZMI
```

Figure 2.1.9

2.1.10. dir

- Lists the files and directories in the current directory.

```
C:\Users\AAFREEN ZAHRA KAZMI>dir
Volume in drive C has no label.
Volume Serial Number is 78B4-1C1D

Directory of C:\Users\AAFREEN ZAHRA KAZMI

10/22/2024  08:57 PM    <DIR>          .
10/22/2024  08:57 PM    <DIR>          ..
05/28/2024  05:44 PM    <DIR>          .dotnet
03/28/2024  12:09 PM    <DIR>          16.emulator_console_auth_token
03/25/2024  07:35 PM    <DIR>          .librarymanager
03/28/2024  11:07 AM    <DIR>          .m2
05/05/2024  12:05 PM    <DIR>          .nuget
03/21/2024  08:32 PM    <DIR>          .temp
03/25/2024  04:23 AM    <DIR>          .templateengine
10/24/2024  10:34 AM    <DIR>          .VirtualBox
08/17/2024  06:18 PM    <DIR>          .vscode
09/28/2024  05:17 PM    <DIR>          3D Objects
03/21/2024  08:32 PM    <DIR>          build-tools
03/21/2024  12:13 PM    <DIR>          Contacts
07/22/2024  03:38 PM    <DIR>          Desktop
07/08/2024  03:19 PM    <DIR>          Documents
10/24/2024  03:30 PM    <DIR>          Downloads
03/21/2024  08:05 PM    <DIR>          emulator
03/21/2024  08:32 PM    <DIR>          extras
03/21/2024  12:13 PM    <DIR>          Favorites
05/06/2024  04:38 PM    <DIR>          80,240.java_error_in_studio64_4144.log
03/21/2024  08:05 PM    <DIR>          licenses
03/21/2024  12:13 PM    <DIR>          Links
03/21/2024  12:13 PM    <DIR>          Music
10/24/2024  09:42 AM    <DIR>          OneDrive
03/21/2024  08:32 PM    <DIR>          platform-tools
03/21/2024  08:32 PM    <DIR>          platforms
03/21/2024  12:13 PM    <DIR>          Saved Games
03/21/2024  12:15 PM    <DIR>          Searches
07/19/2024  01:20 PM    <DIR>          source
03/21/2024  08:32 PM    <DIR>          system-images
08/28/2024  03:27 PM    <DIR>          Tracing
03/21/2024  02:21 PM    <DIR>          Videos
09/14/2024  11:20 AM    <DIR>          VirtualBox VMs
                           2 File(s)      80,256 bytes
                           32 Dir(s)   33,607,139,328 bytes free
```

Figure 2.1.10

2.1.11. find/?

- Displays help for the find command.

```
C:\Users\AAFREEN ZAHRA KAZMI>find "hello" hlo.txt  
File not found - HLO.TXT
```

*Figure 2.1.11***2.1.12. runas**

- Runs a program as a different user.

```
C:\Users\AAFREEN ZAHRA KAZMI>runas /user:administrator cmd  
Enter the password for administrator:  
Attempting to start cmd as user "DESKTOP-46UPK5M\administrator" ...  
RUNAS ERROR: Unable to run - cmd  
1326: The user name or password is incorrect.
```

*Figure 2.1.12***2.1.13. cd download**

- Changes the current directory.

```
C:\Users\AAFREEN ZAHRA KAZMI>CD downloads  
  
C:\Users\AAFREEN ZAHRA KAZMI\Downloads>
```

Figure 2.1.13

2.1.14. Sort

- Sorts the input and displays the sorted output.

```
C:\Users\AAFREEN ZAHRA KAZMI\Downloads>sort hlo.txt

_its_kashaf
2428aafreen#
kashaf123#
```

Figure 2.1.14

2.1.15. Taskkill

- Ends tasks or processes by process ID (PID) or image name.

```
C:\Users\AAFREEN ZAHRA KAZMI\Downloads>taskkill /PID 1234
ERROR: The process "1234" not found.
```

Figure 2.1.15

2.1.16. Tasklist

- Displays a list of currently running process

Image Name	PID	Session	Name	Session#	Mem Usage
System Idle Process	0	Services		0	8 K
System	4	Services		0	3,040 K
Secure System	56	Services		0	22,768 K
Registry	108	Services		0	84,468 K
smss.exe	428	Services		0	1,020 K
csrss.exe	672	Services		0	4,500 K
wininit.exe	868	Services		0	5,876 K
csrss.exe	876	Console		1	5,668 K
services.exe	948	Services		0	10,000 K
LsaIso.exe	968	Services		0	3,488 K
lsass.exe	976	Services		0	22,312 K
svchost.exe	616	Services		0	33,700 K
Fontdrvhost.exe	708	Services		0	3,152 K
svchost.exe	780	Services		0	19,264 K
svchost.exe	676	Services		0	8,300 K
winlogon.exe	800	Console		1	11,008 K
Fontdrvhost.exe	1072	Console		1	10,068 K
svchost.exe	1148	Services		0	7,300 K
svchost.exe	1160	Services		0	5,348 K
svchost.exe	1252	Services		0	10,740 K
svchost.exe	1260	Services		0	8,200 K
svchost.exe	1276	Services		0	11,444 K
svchost.exe	1304	Services		0	10,632 K
svchost.exe	1372	Services		0	10,252 K
svchost.exe	1508	Services		0	8,232 K
svchost.exe	1516	Services		0	16,960 K
dwm.exe	1612	Console		1	65,836 K
svchost.exe	1676	Services		0	7,860 K
svchost.exe	1772	Services		0	8,368 K
svchost.exe	1820	Services		0	11,856 K
svchost.exe	1852	Services		0	10,864 K
svchost.exe	1940	Services		0	7,836 K
svchost.exe	1952	Services		0	15,440 K
svchost.exe	2000	Services		0	11,220 K
svchost.exe	2072	Services		0	5,916 K
svchost.exe	2144	Services		0	10,736 K
svchost.exe	2208	Services		0	13,200 K
svchost.exe	2252	Services		0	7,788 K
svchost.exe	2280	Services		0	18,796 K
svchost.exe	2336	Services		0	10,720 K

Figure 2.1.16

2.1.17. w32tm

- Used for configuring and monitoring Windows Time service.

```
C:\Users\AAFREEN ZAHRA KAZMI>w32tm/monitor  
GetDcList failed with error code: 0x8007054B.  
Exiting with error 0x8007054B
```

Figure 2.1.17

2.1.18. Start

- Starts a program or opens a new command prompt window.

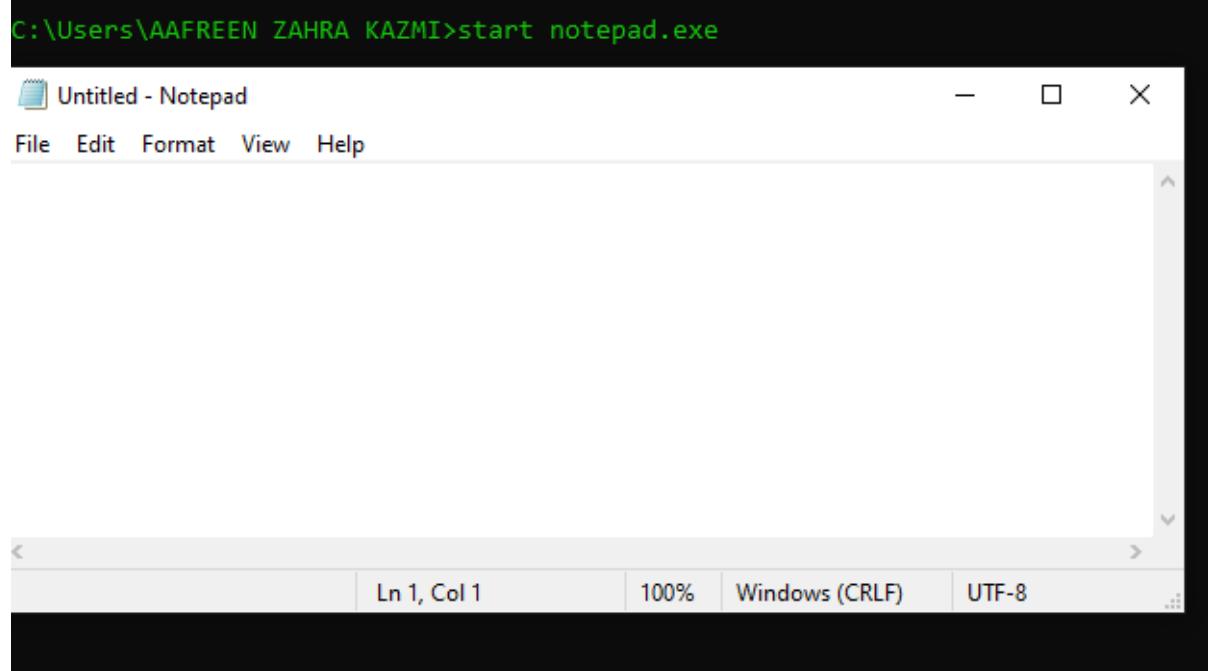


Figure 2.1.18

2.1.19. Timeout

- Pauses the command prompt for a specified amount of time.

```
C:\Users\AAFREEN ZAHRA KAZMI>timeout 20  
Waiting for 15_seconds, press a key to continue ...
```

Figure 2.1.19

2.1.20. Pause

- Pauses the execution of a batch script and waits for user input.

```
C:\Users\AAFREEN ZAHRA KAZMI>pause  
Press any key to continue . . .
```

Figure 2.1.20

2.1.21. Exit

- Exits the command prompt or a batch file.



Figure 2.1.21

2.1.22. shutdown

- Shuts down or restarts the computer.

```
C:\Users\AAFREEN ZAHRA KAZMI>shutdown /s /f /t 0
```

Figure 2.1.22

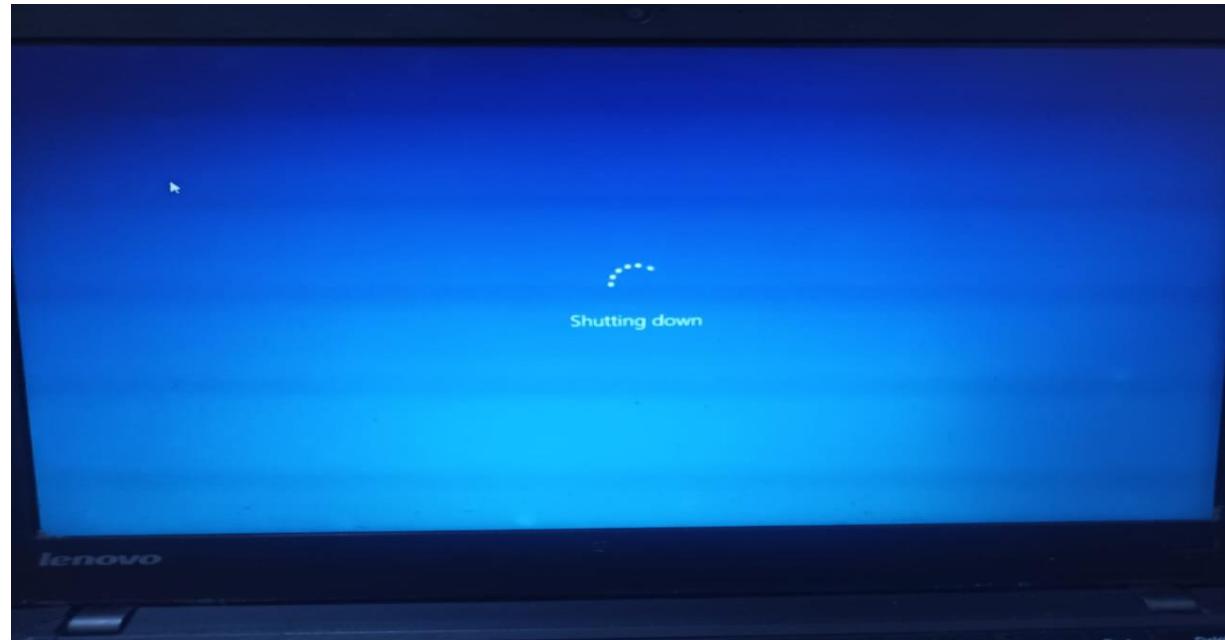


Figure 2.1.22

2.1.23. setfont

- Sets the console font.

```
C:\Users\AAFREEN ZAHRA KAZMI>setfont Arial  
'setfont' is not recognized as an internal or external command,  
operable program or batch file.
```

Figure 2.1.23

2.1.24. ftp

- Transfers files to and from a remote network.

```
C:\Users\AAFREEN ZAHRA KAZMI>ftp youtube.com  
> ftp: connect :Connection timed out  
ftp>  
ftp>
```

Figure 2.1.24

2.1.25. ftype

- Displays or modifies file types used in file extension associations.

```
C:\Users\AAFREEN ZAHRA KAZMI\Downloads>ftype txtfile="notepad.exe %1"  
Access is denied.  
Error occurred while processing: txtfile.
```

Figure 2.1.25

2.1.26. getmac

- Displays the MAC address of network adapters.

```
C:\Users\AAFREEN ZAHRA KAZMI\Downloads>getmac

Physical Address      Transport Name
===== =====
34-02-86-17-FE-76    \Device\Tcpip_{356A51F3-319A-42F0-A466-FB3B14307D34}
68-F7-28-7D-E3-A0    Media disconnected
34-02-86-17-FE-7A    Media disconnected
00-15-5D-3C-B0-D8    \Device\Tcpip_{C90B15F6-9655-45C5-A310-E3FA51B9420B}
0A-00-27-00-00-0E    \Device\Tcpip_{4CBD0FB5-E280-46BF-96A1-7FD651B3646C}
```

Figure 2.1.26

2.1.27. Ipconfig

- Displays the network configuration of the computer.

```
C:\Users\AAFREEN ZAHRA KAZMI\Downloads>ipconfig

Windows IP Configuration

Ethernet adapter Ethernet:

  Media State . . . . . : Media disconnected
  Connection-specific DNS Suffix . : Home

Ethernet adapter Ethernet 6:

  Connection-specific DNS Suffix . :
  Link-local IPv6 Address . . . . . : fe80::8e17:8130:9253:2996%14
  IPv4 Address. . . . . : 192.168.56.1
  Subnet Mask . . . . . : 255.255.255.0
  Default Gateway . . . . . :

Wireless LAN adapter Local Area Connection* 1:

  Media State . . . . . : Media disconnected
  Connection-specific DNS Suffix . :

Wireless LAN adapter Local Area Connection* 2:

  Media State . . . . . : Media disconnected
  Connection-specific DNS Suffix . :

Wireless LAN adapter WiFi:

  Connection-specific DNS Suffix . : Home
  Link-local IPv6 Address . . . . . : fe80::b183:4753:421f:6e28%12
  IPv4 Address. . . . . : 192.168.10.6
  Subnet Mask . . . . . : 255.255.255.0
  Default Gateway . . . . . : 192.168.10.1

Ethernet adapter Bluetooth Network Connection:

  Media State . . . . . : Media disconnected
  Connection-specific DNS Suffix . :
```

Figure 2.1.27

2.1.28. Ping

- Tests the connection between your computer and a remote machine.

```
C:\Users\AAFREEN ZAHRA KAZMI>ping youtube.com

Pinging youtube.com [142.250.180.46] with 32 bytes of data:
Reply from 142.250.180.46: bytes=32 time=39ms TTL=56
Reply from 142.250.180.46: bytes=32 time=37ms TTL=56
Reply from 142.250.180.46: bytes=32 time=37ms TTL=56
Reply from 142.250.180.46: bytes=32 time=37ms TTL=56

Ping statistics for 142.250.180.46:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 37ms, Maximum = 39ms, Average = 37ms
```

Figure 2.1.28

2.1.29. route

- Manipulates network routing tables.

```
C:\Users\AAFREEN ZAHRA KAZMI>route print
=====
Interface List
23...68 F7 28 7d e3 a0 ....Intel(R) Ethernet Connection (3) I218-LM
14...0a 00 27 00 00 0e ....VirtualBox Host-Only Ethernet Adapter
25...34 02 86 17 fe 77 ....Microsoft Wi-Fi Direct Virtual Adapter
6...36 02 86 17 fe 76 ....Microsoft Wi-Fi Direct Virtual Adapter #2
12...34 02 86 17 fe 76 ....Intel(R) Dual Band Wireless-AC 7265
15...34 02 86 17 fe 7a ....Bluetooth Device (Personal Area Network)
1..... ....Software Loopback Interface 1
59...00 15 5d 3c b0 d8 ....Hyper-V Virtual Ethernet Adapter
=====

IPv4 Route Table
=====
Active Routes:
Network Destination      Netmask      Gateway      Interface Metric
          0.0.0.0      0.0.0.0  192.168.10.1  192.168.10.6    60
          127.0.0.0      255.0.0.0  On-link       127.0.0.1    331
          127.0.0.1  255.255.255.255  On-link       127.0.0.1    331
127.255.255.255  255.255.255.255  On-link       127.0.0.1    331
 172.22.192.0      255.255.240.0  On-link      172.22.192.1    5256
 172.22.192.1  255.255.255.255  On-link      172.22.192.1    5256
 172.22.207.255  255.255.255.255  On-link      172.22.192.1    5256
 192.168.10.0      255.255.255.0  On-link      192.168.10.6    316
 192.168.10.6  255.255.255.255  On-link      192.168.10.6    316
 192.168.10.255  255.255.255.255  On-link      192.168.10.6    316
 192.168.16.0      255.255.255.0  On-link      192.168.56.1    281
 192.168.56.1  255.255.255.255  On-link      192.168.56.1    281
 192.168.56.255  255.255.255.255  On-link      192.168.56.1    281
 224.0.0.0      240.0.0.0  On-link       127.0.0.1    331
 224.0.0.0      240.0.0.0  On-link      192.168.56.1    281
 224.0.0.0      240.0.0.0  On-link      192.168.10.6    316
 224.0.0.0      240.0.0.0  On-link      172.22.192.1    5256
 255.255.255.255  255.255.255.255  On-link       127.0.0.1    331
 255.255.255.255  255.255.255.255  On-link      192.168.56.1    281
 255.255.255.255  255.255.255.255  On-link      192.168.10.6    316
 255.255.255.255  255.255.255.255  On-link      172.22.192.1    5256
=====
Persistent Routes:
 None
=====

IPv6 Route Table
=====
Active Routes:
If Metric Network Destination      Gateway
 1     331 ::1/128  On-link
 14    281 fe80::/64  On-link
 12    316 fe80::/64  On-link
 59    5256 fe80::/64  On-link
 14    281 fe80::8e17:8130:9253:2996/128  On-link
 59    5256 fe80::ae88:38c2:4a7f:9798/128  On-link
 12    316 fe80::b183:4753:421f:6e28/128  On-link
 1     331 ff00::/8   On-link
=====
```

Figure 2.1.29

2.1.30. systeminfo

- Displays detailed configuration information about the computer.

```
C:\Users\AAFREEN ZAHRA KAZMI>systeminfo

Host Name: DESKTOP-46UPK5M
OS Name: Microsoft Windows 10 Pro
OS Version: 10.0.19045 N/A Build 19045
OS Manufacturer: Microsoft Corporation
OS Configuration: Standalone Workstation
OS Build Type: Multiprocessor Free
Registered Owner: AAFREEN ZAHRA KAZMI
Registered Organization:
Product ID: 00330-50000-00000-AAOEM
Original Install Date: 3/21/2024, 4:54:07 PM
System Boot Time: 10/26/2024, 10:10:31 AM
System Manufacturer: LENOVO
System Model: 20CLS2KS0S
System Type: x64-based PC
Processor(s): 1 Processor(s) Installed.
[01]: Intel64 Family 6 Model 61 Stepping 4 GenuineIntel ~2295 Mhz
BIOS Version: LENOVO N10ET50W (1.29 ), 2/26/2018
Windows Directory: C:\Windows
System Directory: C:\Windows\system32
Boot Device: \Device\HarddiskVolume1
System Locale: en-gb;English (United Kingdom)
Input Locale: en-us;English (United States)
Time Zone: (UTC+05:00) Islamabad, Karachi
Total Physical Memory: 8,071 MB
Available Physical Memory: 3,403 MB
Virtual Memory: Max Size: 9,351 MB
Virtual Memory: Available: 4,957 MB
Virtual Memory: In Use: 4,394 MB
Page File Location(s): C:\pagefile.sys
Domain: WORKGROUP
Logon Server: \\DESKTOP-46UPK5M
Hotfix(s): 21 Hotfix(s) Installed.
[01]: KB5044029
[02]: KB5034468
[03]: KB4537759
[04]: KB4557968
[05]: KB4577586
[06]: KB5006120
[07]: KB5007115
[08]: KB5011048
[09]: KB5015684
[10]: KB5033052
```

Figure 2.1.30

2.1.31. netstat

- Displays network statistics and active connections.

```
C:\Users\AAFREEN ZAHRA KAZMI>netstat -a

Active Connections

  Proto  Local Address          Foreign Address        State
  TCP    0.0.0.0:135           DESKTOP-46UPK5M:0      LISTENING
  TCP    0.0.0.0:445           DESKTOP-46UPK5M:0      LISTENING
  TCP    0.0.0.0:2179          DESKTOP-46UPK5M:0      LISTENING
  TCP    0.0.0.0:5040          DESKTOP-46UPK5M:0      LISTENING
  TCP    0.0.0.0:5357          DESKTOP-46UPK5M:0      LISTENING
  TCP    0.0.0.0:7680          DESKTOP-46UPK5M:0      LISTENING
  TCP    0.0.0.0:49664          DESKTOP-46UPK5M:0      LISTENING
  TCP    0.0.0.0:49665          DESKTOP-46UPK5M:0      LISTENING
  TCP    0.0.0.0:49666          DESKTOP-46UPK5M:0      LISTENING
  TCP    0.0.0.0:49667          DESKTOP-46UPK5M:0      LISTENING
  TCP    0.0.0.0:49668          DESKTOP-46UPK5M:0      LISTENING
  TCP    0.0.0.0:53855          DESKTOP-46UPK5M:0      LISTENING
  TCP    172.22.192.1:139       DESKTOP-46UPK5M:0      LISTENING
  TCP    192.168.10.6:139       DESKTOP-46UPK5M:0      LISTENING
  TCP    192.168.10.6:51325     20.198.119.143:https ESTABLISHED
```

Figure 2.1.31

2.1.32. telnet

- Connects to remote computers using the Telnet protocol.

```
C:\Users\AAFREEN ZAHRA KAZMI>telnet <DESKTOP-46UPK5M> [1234]
The system cannot find the file specified.
```

Figure 2.1.32

2.1.33. netview

- Displays a list of computers on the network.

```
C:\Users\AAFREEN ZAHRA KAZMI>net view
System error 6118 has occurred.

The list of servers for this workgroup is not currently available
```

Figure 2.1.33

2.1.34. Arp

- Displays or modifies the ARP (Address Resolution Protocol) cache.

```
C:\Users\AAFREEN ZAHRA KAZMI>arp -a

Interface: 192.168.10.6 --- 0xc
 Internet Address      Physical Address      Type
192.168.10.1           40-9b-cd-6f-25-25    dynamic
192.168.10.255         ff-ff-ff-ff-ff-ff    static
224.0.0.2               01-00-5e-00-00-02    static
224.0.0.22              01-00-5e-00-00-16    static
224.0.0.251             01-00-5e-00-00-fb    static
224.0.0.252             01-00-5e-00-00-fc    static
239.255.255.250         01-00-5e-7f-ff-fa    static
255.255.255.255         ff-ff-ff-ff-ff-ff    static

Interface: 192.168.56.1 --- 0xe
 Internet Address      Physical Address      Type
192.168.56.255         ff-ff-ff-ff-ff-ff    static
224.0.0.2               01-00-5e-00-00-02    static
224.0.0.22              01-00-5e-00-00-16    static
224.0.0.251             01-00-5e-00-00-fb    static
224.0.0.252             01-00-5e-00-00-fc    static
239.255.255.250         01-00-5e-7f-ff-fa    static

Interface: 172.22.192.1 --- 0x3b
 Internet Address      Physical Address      Type
172.22.207.255         ff-ff-ff-ff-ff-ff    static
224.0.0.2               01-00-5e-00-00-02    static
224.0.0.22              01-00-5e-00-00-16    static
224.0.0.251             01-00-5e-00-00-fb    static
239.255.255.250         01-00-5e-7f-ff-fa    static
255.255.255.255         ff-ff-ff-ff-ff-ff    static
```

Figure 2.1.34

2.1.35. nslookup

- Queries the DNS to find IP addresses associated with a domain or vice versa.

```
C:\Users\AAFREEN ZAHRA KAZMI>nslookup
Default Server:  Broadcom.Home
Address: 192.168.10.1
```

Figure 2.1.35

2.1.36. echo

- Displays messages or turns command echoing on/off.

```
C:\Users\AAFREEN ZAHRA KAZMI>echo Hello, World!
Hello, World!
```

Figure 2.1.36

2.1.37. attrib

- Displays or changes file attributes.

```
C:\Users\AAFREEN ZAHRA KAZMI\Downloads>attrib +r aafreen.txt
```

Figure 2.1.37

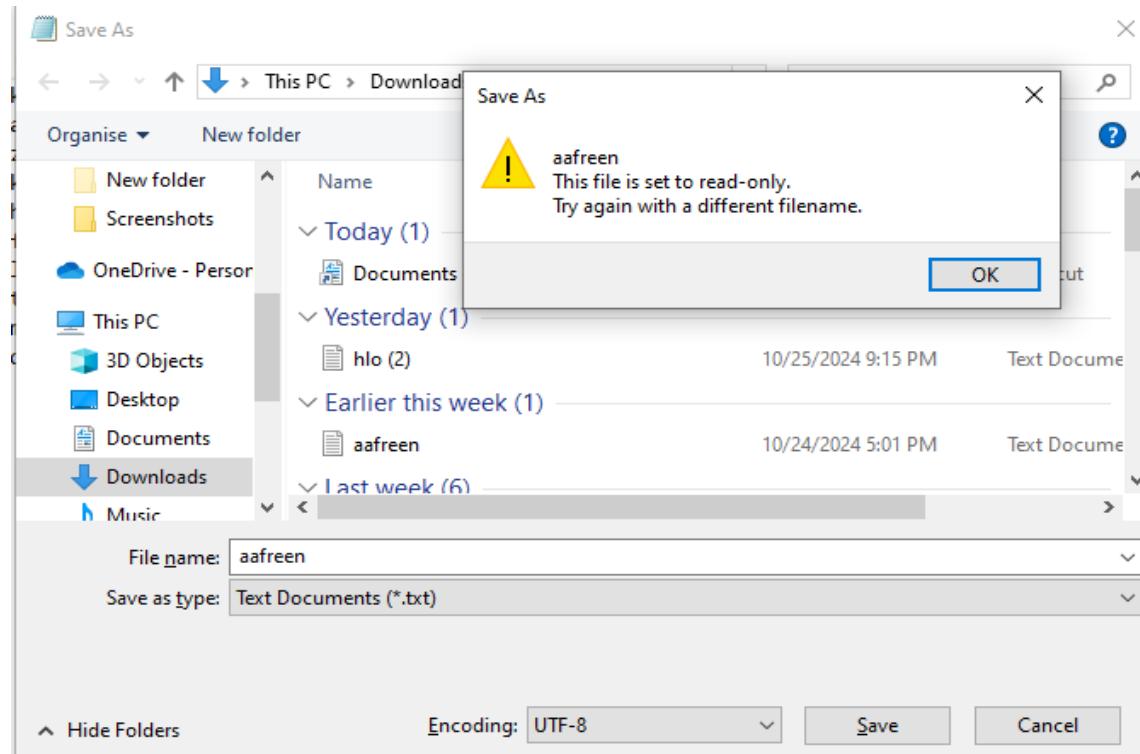


Figure 2.1.37

2.1.38. compact

- Displays or alters the compression of files on NTFS partitions.

```
C:\Users\AAFREEN ZAHRA KAZMI\Downloads>compact /c aafreen.txt  
Compressing files in C:\Users\AAFREEN ZAHRA KAZMI\Downloads\  
aafreen.txt 72 : 72 = 1.0 to 1 [OK]  
  
1 files within 1 directories were compressed.  
72 total bytes of data are stored in 72 bytes.  
The compression ratio is 1.0 to 1.
```

Figure 2.1.38

2.1.39. copy

- Copies files from one location to another.

```
C:\Users\AAFREEN ZAHRA KAZMI\Downloads>copy aafreen.txt D:\aafreen\  
1 file(s) copied.
```

Figure 2.1.39

Name	Date modified	Type	Size
aafreen	10/24/2024 5:01 PM	Text Document	1 KB

Figure 2.1.39

2.1.40. mkdir (or md)

- Creates a new directory.



```
C:\Users\AAFREEN ZAHRA KAZMI\Downloads>mkdir newfolder
```

Figure 2.1.40

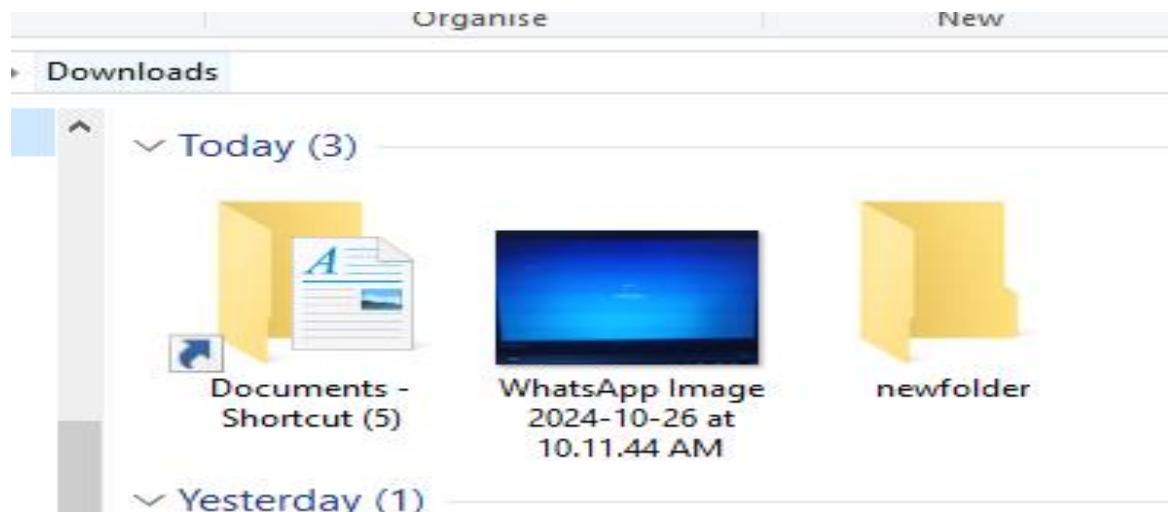


Figure 2.1.40

2.1.41. rename

(or ren)

- Renames a file or directory.

Before execution

Figure 2.1.41

After execution

```
C:\Users\AAFREEN ZAHRA KAZMI\Downloads>rename aafreen.txt fatima.txt
```

Figure 2.1.41



fatima

Figure 2.1.41

2.1.42. rmdir (or rd)

- Removes a directory.

Before execution

Figure 2.1.42

After execution

```
C:\Users\AAFREEN ZAHRA KAZMI\Downloads>rmdir newfolder
```

Figure 2.1.42



Figure 2.1.42

2.1.43. tree

- Displays a graphical representation of the directory structure.

```
C:\Users\AAFREEN ZAHRA KAZMI>tree D:\ 
Folder PATH listing for volume New Volume
Volume serial number is 0000006C 9082:E1BC
D:\ 
    aafreen
    Electronic Work Bench 5.12
    Embarcadero
        Dev-Cpp
            AStyle
                doc
            config
            Fonts
            Help
                FAQ
                Interface
                    Dialog Windows
                        Compiler Options
                        Profile Analysis
                        Project Options
                    Menus
                Subjects
            Icons
            Lang
                history
                recovery
            TDM-GCC-64
                bin
                gdb32
                    bin
                        DLLs
                        Lib
                            asyncio
                            collections
                                __pycache__
                            concurrent
                                futures
                            ctypes
                                macholib
                                test
                            curses
                            dbm
                            distutils
                                command
```

Figure 2.1.43

```
tests
email
    mime
encodings
    __pycache__
ensurepip
    _bundled
html
http
importlib
    __pycache__
json
lib2to3
    fixes
    pgen2
    tests
        data
            fixers
            myfixes
logging
msilib
multiprocessing
    dummy
pydoc_data
site-packages
sqlite3
    test
unittest
    test
        testmock
urllib
venv
    scripts
        common
            nt
            posix
wsgiref
xml
    dom
    etree
    parsers
        sax
    xmlrpc
    __pycache__
```

Figure 2.1.43

2.1.44. type

- Displays the contents of a text file.

```
C:\Users\AAFREEN ZAHRA KAZMI\Downloads>type fatima.txt
kasahf
aafreen
zahra
kazmi
hlo
fatima
laraib
tayaba
maria
```

Figure 2.1.44

2.1.45. chkdsk

- Checks the file system and file system metadata of a volume for logical and physical errors.

```
C:\Users\AAFREEN ZAHRA KAZMI\Downloads>chkdsk D: /f
Access Denied as you do not have sufficient privileges or
the disk may be locked by another process.
You have to invoke this utility running in elevated mode
and make sure the disk is unlocked.
```

Figure 2.1.45

2.1.46. chkntfs

- Displays or modifies the automatic disk checking at startup.

```
C:\Users\AAFREEN ZAHRA KAZMI\Downloads>chkntfs /d
```

Figure 2.1.46

2.1.47. defrag

- Locates and consolidates fragmented files on local volumes to improve system performance.

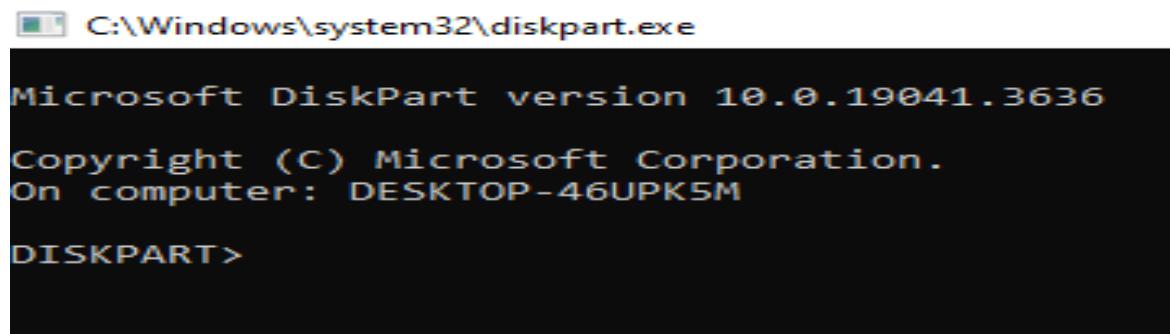
```
C:\Users\AAFREEN ZAHRA KAZMI\Downloads>defrag C: /o
```

```
The storage optimiser cannot start because you have insufficient privileges to perform this operation. (0x89000024)
```

Figure 2.1.47

2.1.48. diskpart

- A command-line disk partitioning utility.



C:\Windows\system32\diskpart.exe

Microsoft DiskPart version 10.0.19041.3636

Copyright (C) Microsoft Corporation.

On computer: DESKTOP-46UPK5M

DISKPART>

Figure 2.1.48

2.1.49. format

- Formats a disk for use with Windows.



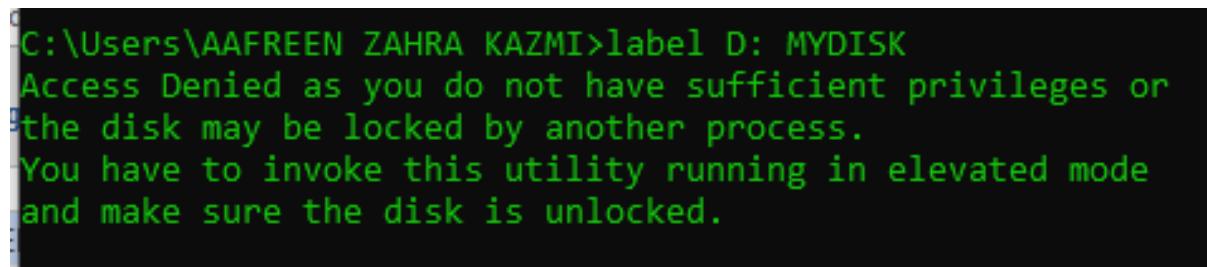
C:\Users\AAFREEN ZAHRA KAZMI>format D: /newfolder\

Invalid parameter - /newfolder\

Figure 2.1.49

2.1.50. label

- Creates, changes, or deletes the volume label of a disk.



C:\Users\AAFREEN ZAHRA KAZMI>label D: MYDISK

Access Denied as you do not have sufficient privileges or
the disk may be locked by another process.

You have to invoke this utility running in elevated mode
and make sure the disk is unlocked.

Figure 2.1.50

2.1.51. mode

- Configures system devices.



Command Prompt

C:\Users\AAFREEN ZAHRA KAZMI>

Figure 2.1.51

2.1.52. mountvol

- Creates, deletes, or lists a volume mount point.

cmd Command Prompt

```
C:\Users\AAFREEN ZAHRA KAZMI>mountvol D: /D  
Access is denied.
```

*Figure 2.1.52***2.1.53. verify**

- Tells Windows whether to verify that your files are written correctly to a disk.

```
C:\Users\AAFREEN ZAHRA KAZMI>verify on
```

*Figure 2.1.53***2.1.54. vol**

- Displays the disk volume label and serial number, if they exist.

```
C:\Users\AAFREEN ZAHRA KAZMI>vol C:  
Volume in drive C has no label.  
Volume Serial Number is 78B4-1C1D
```

*Figure 2.1.54***2.1.55. for**

- Runs a specified command for each file in a set of files.

```
C:\Users\AAFREEN ZAHRA KAZMI>for %i in (*.txt) do type %i  
  
C:\Users\AAFREEN ZAHRA KAZMI>
```

Figure 2.1.55

2.1.56. gpupdate

- Refreshes local and Active Directory-based Group Policy settings, including security settings.

```
C:\Users\AAFREEN ZAHRA KAZMI>gpupdate /force
Updating policy...
Computer Policy update has completed successfully.
```

Figure 2.1.56

2.1.57. gpresult

- Displays the Resultant Set of Policy (RSoP) information for a remote user and computer.

Displays RSoP summary data

```
cmd Command Prompt
OS Configuration: Standalone Workstation
OS Version: 10.0.19045
Site Name: N/A
Roaming Profile: N/A
Local Profile: C:\Users\AAFREEN ZAHRA KAZMI
Connected over a slow link?: No

USER SETTINGS
-----
Last time Group Policy was applied: 10/26/2024 at 12:12:41 PM
Group Policy was applied from: N/A
Group Policy slow link threshold: 500 kbps
Domain Name: DESKTOP-46UPK5M
Domain Type: <Local Computer>

Applied Group Policy Objects
-----
N/A

The following GPOs were not applied because they were filtered out
-----
Local Group Policy
Filtering: Not Applied (Empty)

The user is a part of the following security groups
-----
High Mandatory Level
Everyone
Local account and member of Administrators group
BUILTIN\Administrators
Performance Log Users
BUILTIN\Users
NT AUTHORITY\INTERACTIVE
CONSOLE LOGON
NT AUTHORITY\Authenticated Users
This Organization
aafreenzk1214@gmail.com
Local account
LOCAL
Cloud Account Authentication
```

Figure 2.1.57

Generates a report in HTML format

```
C:\Users\AAFREEN ZAHRA KAZMI>gpresult /h report.html
```

Figure 2.1.57

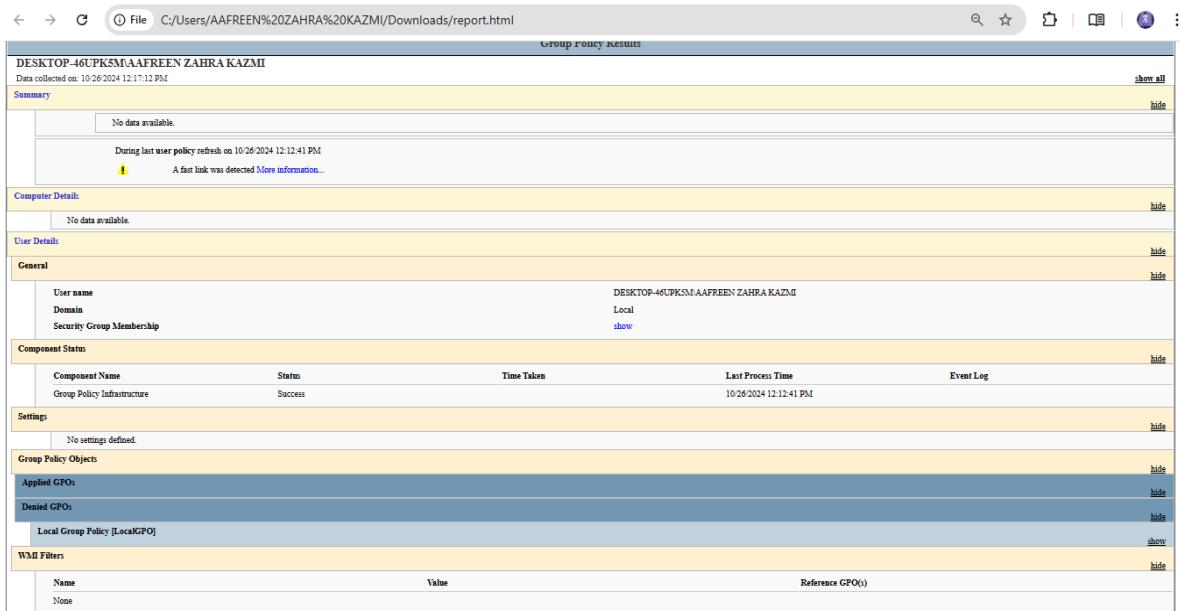


Figure 2.1.57

2.1.58. prompt

- Changes the command prompt.

```
C:\Users\AAFREEN ZAHRA KAZMI>prompt  
  
C:\Users\AAFREEN ZAHRA KAZMI>prompt $P$G  
  
C:\Users\AAFREEN ZAHRA KAZMI>
```

Figure 2.1.58

2.1.59. tracert

- Traces the route that packets take to reach a network host.

```
C:\Users\AAFREEN ZAHRA KAZMI\Downloads>tracert google.com

Tracing route to google.com [192.178.24.142]
over a maximum of 30 hops:

 1    2 ms      1 ms     10 ms  Broadcom.Home [192.168.10.1]
 2    7 ms      5 ms      5 ms  172.16.0.1
 3    6 ms     14 ms     32 ms  10.174.4.49
 4   29 ms      5 ms      6 ms  172.29.254.5
 5    5 ms      9 ms      7 ms  119.63.137.50
 6   34 ms     26 ms     25 ms  110.93.253.110
 7   27 ms     33 ms     25 ms  110.93.254.40
 8   43 ms     43 ms     43 ms  72.14.204.14
 9   46 ms     37 ms     38 ms  192.178.105.155
10   63 ms     39 ms     39 ms  192.178.87.253
11   39 ms      *       39 ms  google.com [192.178.24.142]

Trace complete.
```

Figure 2.1.59

WEEK # 2

3. UBUNTU

Ubuntu is a popular open-source Linux distribution based on Debian, known for its ease of use and robust performance. It features a user-friendly interface, regular updates, and a vast repository of software. Ubuntu is widely used for personal computers, servers, and cloud environments. It emphasizes security, with frequent updates and a strong focus on user privacy.

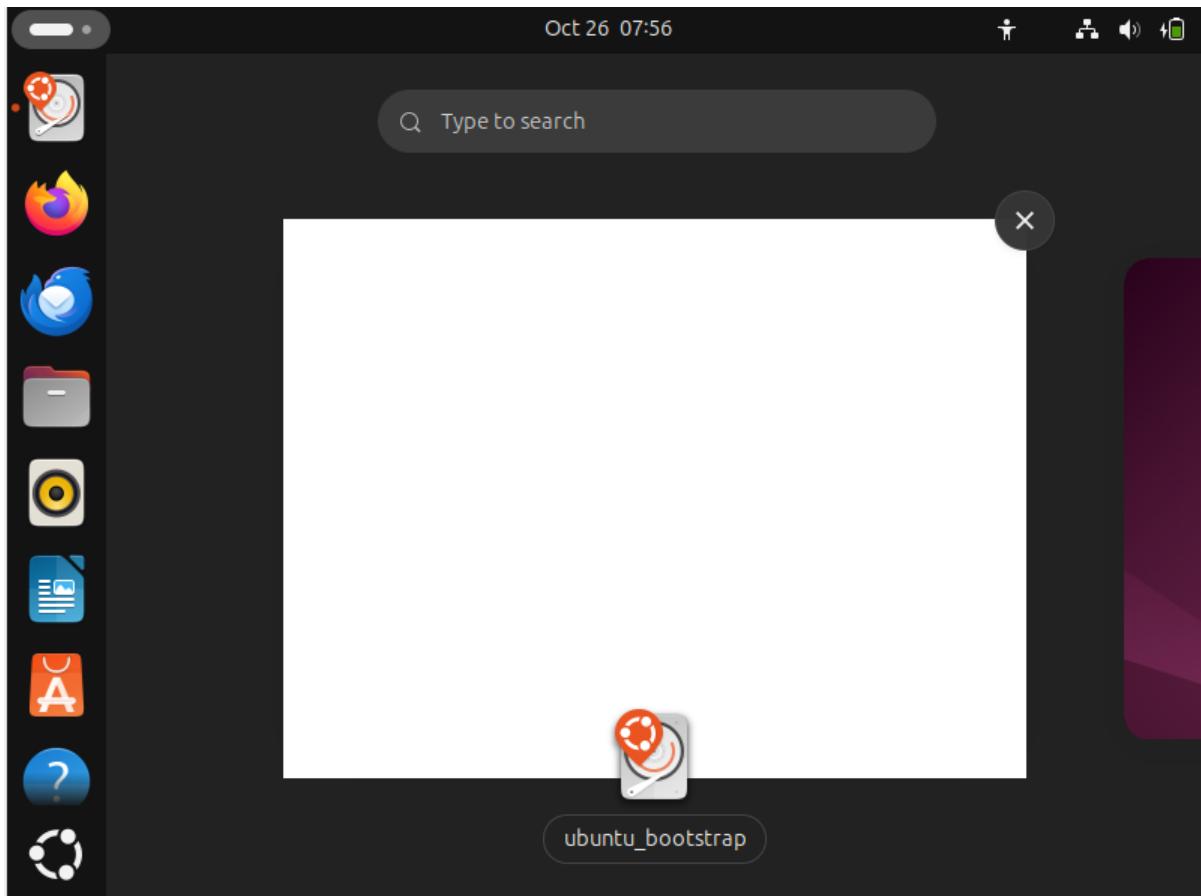


Figure 3

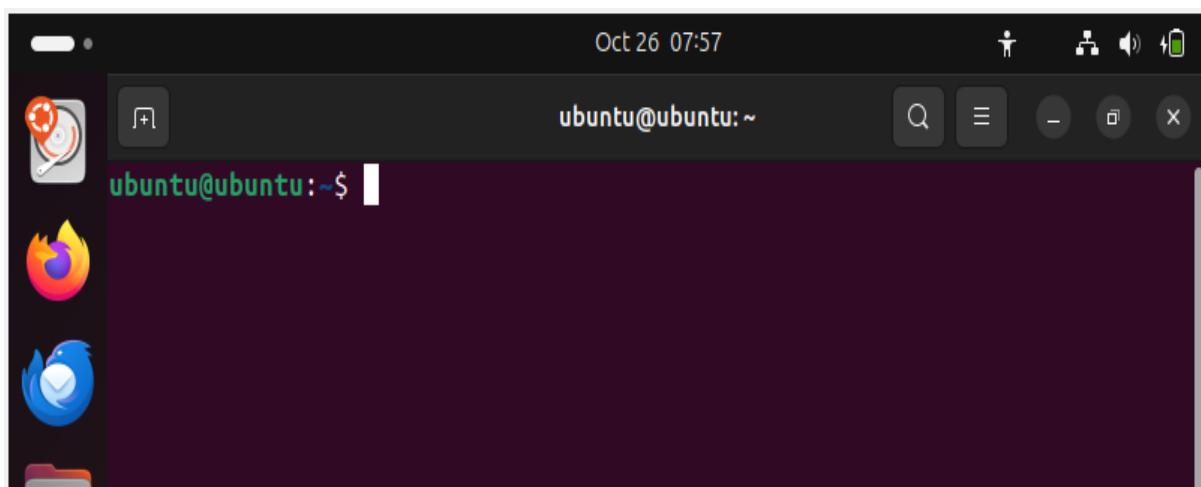


Figure 3

3.1. Files Command

3.1.1. ls

- Lists directory contents.

```
ubuntu@ubuntu:~$ ls
Desktop Downloads Pictures Templates snap
Documents Music Public Videos
ubuntu@ubuntu:~$ █
```

Figure 3.1.1

3.1.2. mkdir

- Creates a new directory.

```
ubuntu@ubuntu:~$ mkdir copies
ubuntu@ubuntu:~$ █
```

Figure 3.1.2

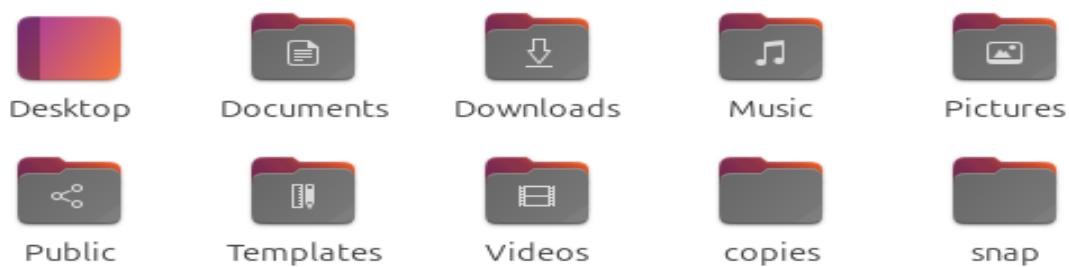


Figure 3.1.2

3.1.3. rmdir

- Removes empty directories.

```
ubuntu@ubuntu:~$ rmdir copies
ubuntu@ubuntu:~$ ls
Desktop Downloads Pictures Templates snap
Documents Music Public Videos
ubuntu@ubuntu:~$ █
```

Figure 3.1.3

3.1.4. cd

- Changes the current directory.

```
ubuntu@ubuntu:~$ cd aafreen  
ubuntu@ubuntu:~/aafreen$ █
```

Figure 3.1.4

3.1.5. sudo -i

- Opens a new shell with root privileges.

```
ubuntu@ubuntu:~/aafreen$ sudo -i  
root@ubuntu:~# █
```

Figure 3.1.5

3.1.6. sudo apt install ncal

- Installs the ncal package using APT (Advanced Package Tool).

```
ubuntu@ubuntu:~$ sudo apt install ncal  
Reading package lists... Done  
Building dependency tree... Done  
Reading state information... Done  
The following NEW packages will be installed:  
  ncal  
0 upgraded, 1 newly installed, 0 to remove and 0 not upgraded.  
Need to get 21.0 kB of archives.  
After this operation, 59.4 kB of additional disk space will be used.  
Get:1 http://archive.ubuntu.com/ubuntu noble/universe amd64 ncal amd64 1  
2.1.8 [21.0 kB]  
Fetched 21.0 kB in 1s (17.6 kB/s)  
Selecting previously unselected package ncal.  
(Reading database ... 210856 files and directories currently installed.)  
Preparing to unpack .../archives/ncal_12.1.8_amd64.deb ...  
Unpacking ncal (12.1.8) ...  
Setting up ncal (12.1.8) ...  
Processing triggers for man-db (2.12.0-4build2) ...  
ubuntu@ubuntu:~$
```

Figure 3.1.6

3.1.7. cal

- Displays a calendar.

Current month

```
ubuntu@ubuntu:~$ cal
          October 2024
Su Mo Tu We Th Fr Sa
                    1  2  3  4  5
 6  7  8  9  10 11 12
13 14 15 16 17 18 19
20 21 22 23 24 25 26
27 28 29 30 31
```

Figure 3.1.7

Calendar for the year 2004

```
ubuntu@ubuntu:~$ cal 2004
          2004
January           February          March
Su Mo Tu We Th Fr Sa Su Mo Tu We Th Fr Sa Su Mo Tu We Th Fr Sa
      1  2  3    1  2  3  4  5  6  7    1  2  3  4  5  6
 4  5  6  7  8  9 10    8  9 10 11 12 13 14    7  8  9 10 11 12 13
11 12 13 14 15 16 17   15 16 17 18 19 20 21   14 15 16 17 18 19 20
18 19 20 21 22 23 24   22 23 24 25 26 27 28   21 22 23 24 25 26 27
25 26 27 28 29 30 31   29                      28 29 30 31

April             May                June
Su Mo Tu We Th Fr Sa Su Mo Tu We Th Fr Sa Su Mo Tu We Th Fr Sa
      1  2  3    2  3  4  5  6  7  8    1  2  3  4  5
 4  5  6  7  8  9 10    9 10 11 12 13 14 15    6  7  8  9 10 11 12
11 12 13 14 15 16 17   16 17 18 19 20 21 22   13 14 15 16 17 18 19
18 19 20 21 22 23 24   23 24 25 26 27 28 29   20 21 22 23 24 25 26
25 26 27 28 29 30       30 31                      27 28 29 30

July              August            September
Su Mo Tu We Th Fr Sa Su Mo Tu We Th Fr Sa Su Mo Tu We Th Fr Sa
      1  2  3    1  2  3  4  5  6  7    1  2  3  4
 4  5  6  7  8  9 10    8  9 10 11 12 13 14    5  6  7  8  9 10 11
```

Figure 3.1.7

Calendar for any month 2004

```
ubuntu@ubuntu:~$ cal april 2004
        April 2004
Su Mo Tu We Th Fr Sa
                1   2   3
 4   5   6   7   8   9   10
11  12  13  14  15  16  17
18  19  20  21  22  23  24
25  26  27  28  29  30
```

Figure 3.1.7

Previous, current, and next month

```
ubuntu@ubuntu:~$ cal -3
September 2004          October 2004          November 2004
Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa
 1   2   3   4   5   6   7      1   2   3   4   5      1   2
 8   9   10  11  12  13  14     6   7   8   9   10  11  12     3   4   5   6   7   8   9
15  16  17  18  19  20  21    13  14  15  16  17  18  19    10  11  12  13  14  15  16
22  23  24  25  26  27  28    20  21  22  23  24  25  26    17  18  19  20  21  22  23
29  30                      27  28  29  30  31                  24  25  26  27  28  29  30
```

Figure 3.1.7

Calendar for the year 2004

```
ubuntu@ubuntu:~$ cal -y 2004
2004
January           February           March
Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa
                    1   2   3      1   2   3   4   5   6   7      1   2   3   4   5   6
 4   5   6   7   8   9   10     8   9   10  11  12  13  14     7   8   9   10  11  12  13
11  12  13  14  15  16  17    15  16  17  18  19  20  21    14  15  16  17  18  19  20
18  19  20  21  22  23  24    22  23  24  25  26  27  28    21  22  23  24  25  26  27
25  26  27  28  29  30  31    29                      28  29  30  31

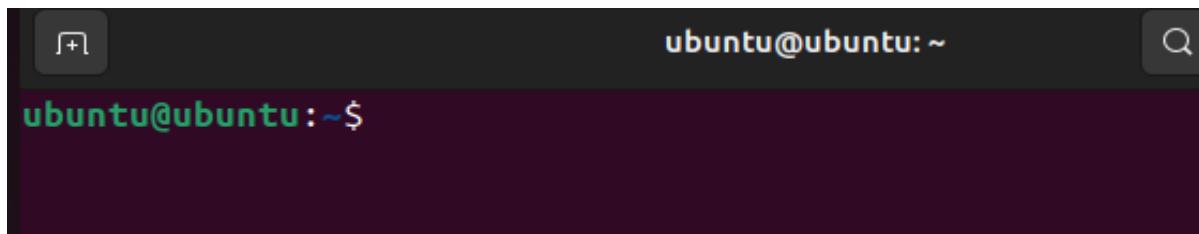
April             May               June
Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa
                    1   2   3      1   2   3   4   5   6   7      1   2   3   4   5
 4   5   6   7   8   9   10     2   3   4   5   6   7   8      6   7   8   9   10  11  12
11  12  13  14  15  16  17    9   10  11  12  13  14  15     13  14  15  16  17  18  19
18  19  20  21  22  23  24   16  17  18  19  20  21  22     20  21  22  23  24  25  26
25  26  27  28  29  30          23  24  25  26  27  28  29     27  28  29  30
                                30  31

July              August            September
Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa
                    1   2   3      1   2   3   4   5   6   7      1   2   3   4
 4   5   6   7   8   9   10     8   9   10  11  12  13  14     5   6   7   8   9   10  11
```

Figure 3.1.7

3.1.8. clear

- Clears the terminal screen.

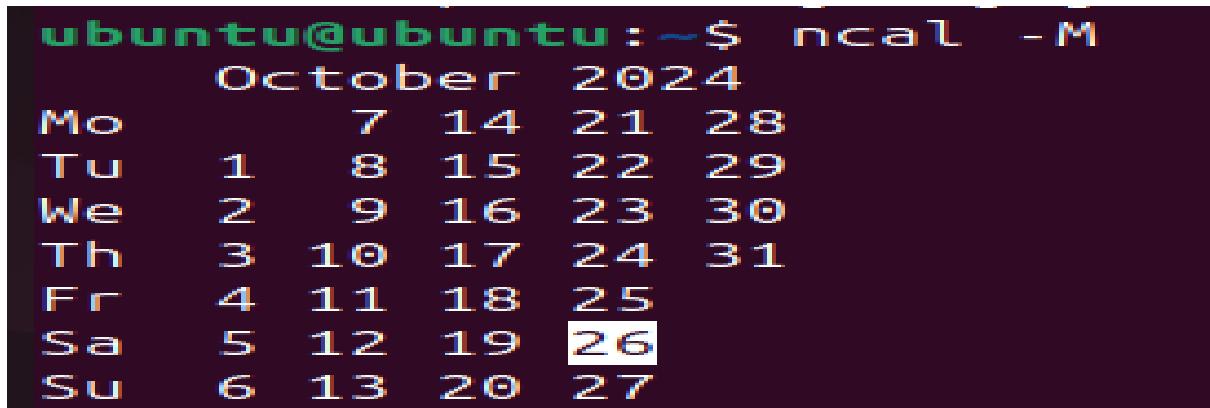


```
ubuntu@ubuntu:~$
```

Figure 3.1.8

3.1.9. ncal -M

- Displays the current month with highlighted current date

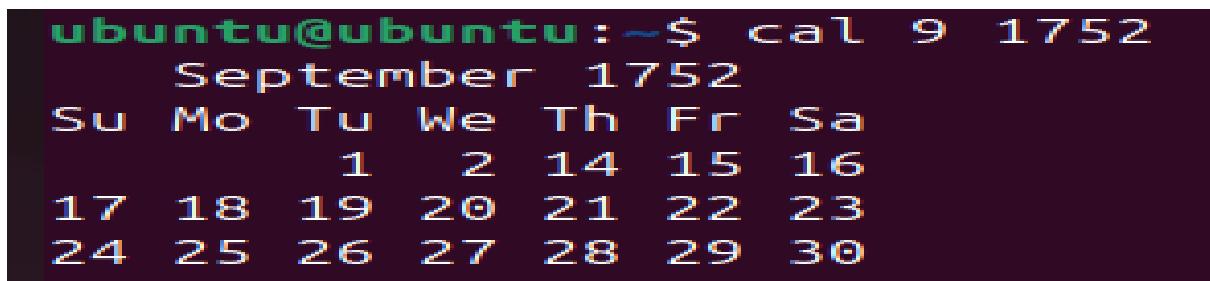


```
ubuntu@ubuntu:~$ ncal -M
          October 2024
Mo       7  14  21  28
Tu       1  8  15  22  29
We       2  9  16  23  30
Th       3  10  17  24  31
Fr       4  11  18  25
Sa       5  12  19  26
Su       6  13  20  27
```

Figure 3.1.9

3.1.10. cal 9 1752

- Displays the calendar for September 1752 (special case in the Gregorian calendar).



```
ubuntu@ubuntu:~$ cal 9 1752
          September 1752
Su Mo Tu We Th Fr Sa
              1  2  14  15  16
17 18 19 20 21 22 23
24 25 26 27 28 29 30
```

Figure 3.1.10

3.1.11. time

- Displays the current time.



```
ubuntu@ubuntu:~$ time
real      0m0.000s
user      0m0.000s
sys       0m0.000s
```

Figure 3.1.11

3.1.12. date

- Displays or sets the system date and time.

```
ubuntu@ubuntu:~$ date
Sat Oct 26 08:16:16 UTC 2024
```

Figure 3.1.12

3.1.13. hostname

- Displays or sets the system's hostname.

```
ubuntu@ubuntu:~$ hostname
ubuntu
```

Figure 3.1.13

3.1.14. sudo snap install chromium

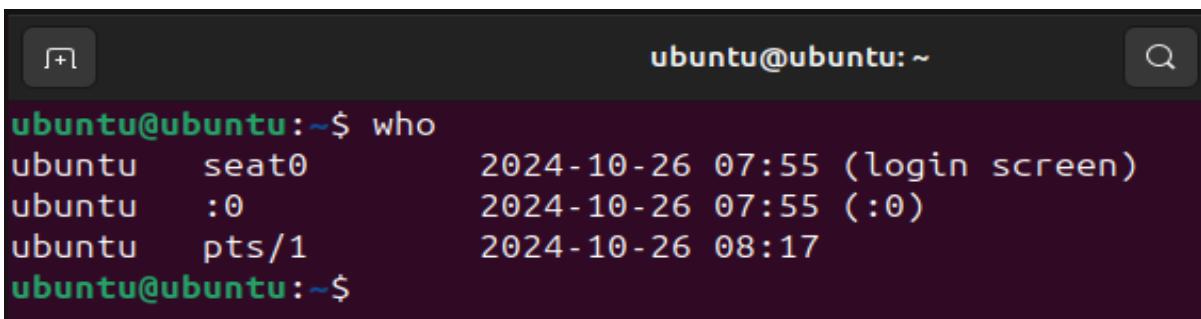
- Installs the Chromium web browser using Snap.

```
ubuntu@ubuntu:~$ sudo apt install chromium
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Note, selecting 'chromium-browser' instead of 'chromium'
The following NEW packages will be installed:
  chromium-browser
0 upgraded, 1 newly installed, 0 to remove and 0 not upgraded.
Need to get 50.0 kB of archives.
After this operation, 105 kB of additional disk space will be used.
Get:1 http://archive.ubuntu.com/ubuntu noble/universe amd64 chromium-bro
wser amd64 2:1snap1-0ubuntu2 [50.0 kB]
Fetched 50.0 kB in 4s (11.5 kB/s)
Preconfiguring packages ...
Selecting previously unselected package chromium-browser.
(Reading database ... 210864 files and directories currently installed.)
Preparing to unpack .../chromium-browser_2%3a1snap1-0ubuntu2_amd64.deb .
..
=> Installing the chromium snap
==> Checking connectivity with the snap store
==> Installing the chromium snap
```

Figure 3.1.14

3.1.15. who

- Shows who is logged on.



```
ubuntu@ubuntu:~$ who
ubuntu    seat0      2024-10-26 07:55 (login screen)
ubuntu    :0        2024-10-26 07:55 (:0)
ubuntu    pts/1      2024-10-26 08:17
ubuntu@ubuntu:~$
```

Figure 3.1.15

3.1.16. History

- Displays the command history.

```
ubuntu@ubuntu:~$ history
 1  ls
 2  mkdir copies
 3  rmdir copies
 4  ls
 5  cd aafreen
 6  mkdir aafreen
 7  cd aafreen
 8  sudo -i
 9  sudo apt install ncal
10  cal
11  cal 2004
12  cal april 2004
13  cal -3
14  cal -y 2004
15  clear
16  ncal -m
17  ncal -M
18  cal 9 1752
19  time
20  date
21  hostname
22  sudo apt install chromium
23  .
```

Figure 3.1.16

3.1.17. sudo snap install vlc

- Installs the VLC media player using Snap.

```
ubuntu@ubuntu:~$ sudo snap install vlc
Download snap "core18" (2846) from channel "stable"    1% 24.9kB/s 38.4m
```

Figure 3.1.17

3.1.18. cd /

- Changes to the root directory.

```
ubuntu@ubuntu:~$ cd /
ubuntu@ubuntu:/$
```

Figure 3.1.18

3.1.19. cat >> bar.txt

- Appends text to bar.txt file (press Ctrl+D to save and exit).

```
ubuntu@ubuntu:~$ sudo sh -c "cat >> bar.txt"
this is aafreen
ubuntu@ubuntu:~$ ls
Desktop   Downloads  Pictures  Templates  aafreen  snap
Documents  Music      Public    Videos     bar.txt
ubuntu@ubuntu:~$
```

Figure 3.1.19

3.1.20. Permissions to make a file readable,writeable ,executable

- Symbolic Notation

- r - read
- w - write
- x - execute
- u - user (owner)
- g - group
- o - others
- a - all (user, group, others)

Readable

```
ubuntu@ubuntu:~$ chmod a=r aafreen.txt
ubuntu@ubuntu:~$
```

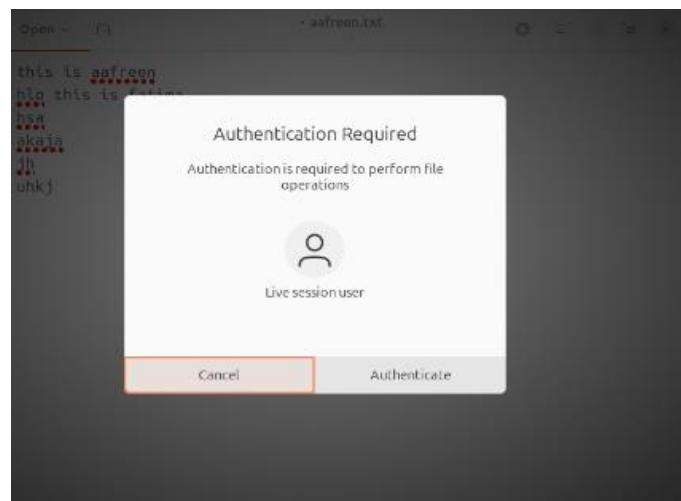


Figure 3.1.20

Figure 3.1.20

Writable

```
ubuntu@ubuntu:~$ chmod a=w aafreen.txt  
ubuntu@ubuntu:~$ █
```

Figure 3.1.20



Figure 3.1.20

Executable

```
ubuntu@ubuntu:~$ chmod a=x aafreen.txt  
ubuntu@ubuntu:~$ █
```

Figure 3.1.20

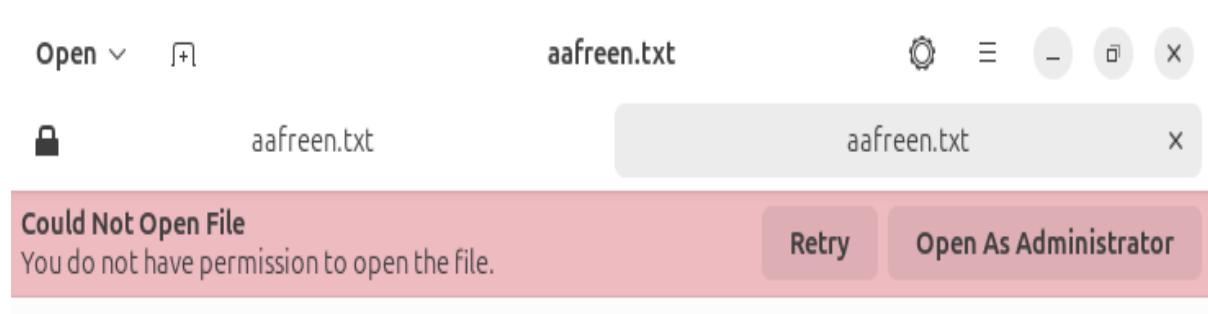


Figure 3.1.20

3.1.21. File

- Determines the type of a file

```
ubuntu@ubuntu:~$ file aafreen.txt  
aafreen.txt: executable, regular file, no read permission
```

Figure 3.1.21

3.1.22. Head

- Outputs the first few lines of a file.

```
ubuntu@ubuntu:~$ head aafreen.txt
this is aafreen
hlo this is fatima
hsa
akaja
jh
uhkj
ahjahi
```

Figure 3.1.22

3.1.23. Which

- Displays the full path of the executable file for a command.

```
ubuntu@ubuntu:~$ which java
ubuntu@ubuntu:~$ which c++
ubuntu@ubuntu:~$ which make
```

Figure 3.1.23

3.1.24. Whereis

- Locates the binary, source, and manual page files for a command.

```
ubuntu@ubuntu:~$ whereis make
make:
```

Figure 3.1.24

3.1.25. Locate

- Quickly finds files and directories by name using a pre-built index.

```
[Processing triggers for libc-bin (2.27-3ubuntu1) ...
ubuntu@ubuntu:~$ locate aafreen.txt
/home/ubuntu/aafreen.txt
ubuntu@ubuntu:~$
```

Figure 3.1.25

3.1.26. Find

- Searches for files and directories based on various criteria.

```
ubuntu@ubuntu:~$ find aafreen.txt  
aafreen.txt
```

*Figure 3.1.26***3.1.27. Ps**

- Displays information about running processes.

```
ubuntu@ubuntu:~$ ps  
 PID  TTY      TIME   CMD  
3860  pts/0    00:00:00  bash  
4954  pts/0    00:00:00  ps
```

*Figure 3.1.27***3.1.28. Tty**

- Prints the file name of the terminal connected to the standard input.

```
ubuntu@ubuntu:~$ tty  
/dev/pts/0
```

Figure 3.1.28

3.1.29. ps aux

- Displays a detailed list of all running processes for all users.

USER	PID	%CPU	%MEM	VSZ	RSS	TTY	STAT	START	TIME	COMMA
ND										
root	1	0.3	0.6	23360	12928	?	Ss	08:44	0:08	/sbin
root	2	0.0	0.0	0	0	?	S	08:44	0:00	[kthr]
root	3	0.0	0.0	0	0	?	S	08:44	0:00	[pool]
root	4	0.0	0.0	0	0	?	I<	08:44	0:00	[kwor]
root	5	0.0	0.0	0	0	?	I<	08:44	0:00	[kwor]
root	6	0.0	0.0	0	0	?	I<	08:44	0:00	[kwor]
root	7	0.0	0.0	0	0	?	I<	08:44	0:00	[kwor]
root	10	0.0	0.0	0	0	?	I<	08:44	0:00	[kwor]
root	11	0.0	0.0	0	0	?	I	08:44	0:00	[kwor]
root	12	0.0	0.0	0	0	?	I<	08:44	0:00	[kwor]
root	13	0.0	0.0	0	0	?	I	08:44	0:00	[rcu_]
root	14	0.0	0.0	0	0	?	I	08:44	0:00	[rcu_]
root	15	0.0	0.0	0	0	?	I	08:44	0:00	[rcu_]
root	16	0.2	0.0	0	0	?	S	08:44	0:07	[ksof]
root	17	0.2	0.0	0	0	?	I	08:44	0:05	[rcu_]
root	18	0.0	0.0	0	0	?	S	08:44	0:00	[migr]
root	19	0.0	0.0	0	0	?	S	08:44	0:00	[idle]
root	20	0.0	0.0	0	0	?	S	08:44	0:00	[cpuh]
root	21	0.0	0.0	0	0	?	S	08:44	0:00	[cpuh]
root	22	0.0	0.0	0	0	?	S	08:44	0:00	[idle]

Figure 3.1.29

3.1.30. Kill

- Sends a signal to terminate a process.

15:50 pts/0		00:00:00 ps	
ubuntu@ubuntu:~\$ kill 3860		ubuntu@ubuntu:~\$ ps	
PID	TTY	TIME	CMD
3860	pts/0	00:00:00	bash
4987	pts/0	00:00:00	ps

Figure 3.1.30

3.1.31. Echo

- Displays a line of text or variable values.

```
ubuntu@ubuntu:~$ echo "hello ! aafreen ! how are you "
hello ! aafreen ! how are you
```

Figure 3.1.31

3.1.32. Rm

- Removes files or directories.

```
ubuntu@ubuntu:~$ rm aafreen.txt
rm: remove write-protected regular file 'aafreen.txt'? Y
ubuntu@ubuntu:~$ ls
Desktop   Downloads  Pictures  Templates  snap
Documents  Music      Public    Videos

```

Figure 3.1.32

3.1.33. Tail

- Outputs the last few lines of a file.

```
ubuntu@ubuntu:~$ tail fatima.txt
aafren
kahsa
fatima
laraib
laiba
hlo
karaci
```

Figure 3.1.33

3.1.34. Sort

- Arranges lines of text files in a specified order.

```
ubuntu@ubuntu:~$ sort fatima.txt
aafren
fatima
hlo
kahsa
karaci
laiba
laraib
```

*Figure 3.1.34***3.1.35. Grep**

- Searches for a specified pattern in files.

```
ubuntu@ubuntu:~$ grep "laiba" fatima.txt
laiba
```

*Figure 3.1.35***3.1.36. Less**

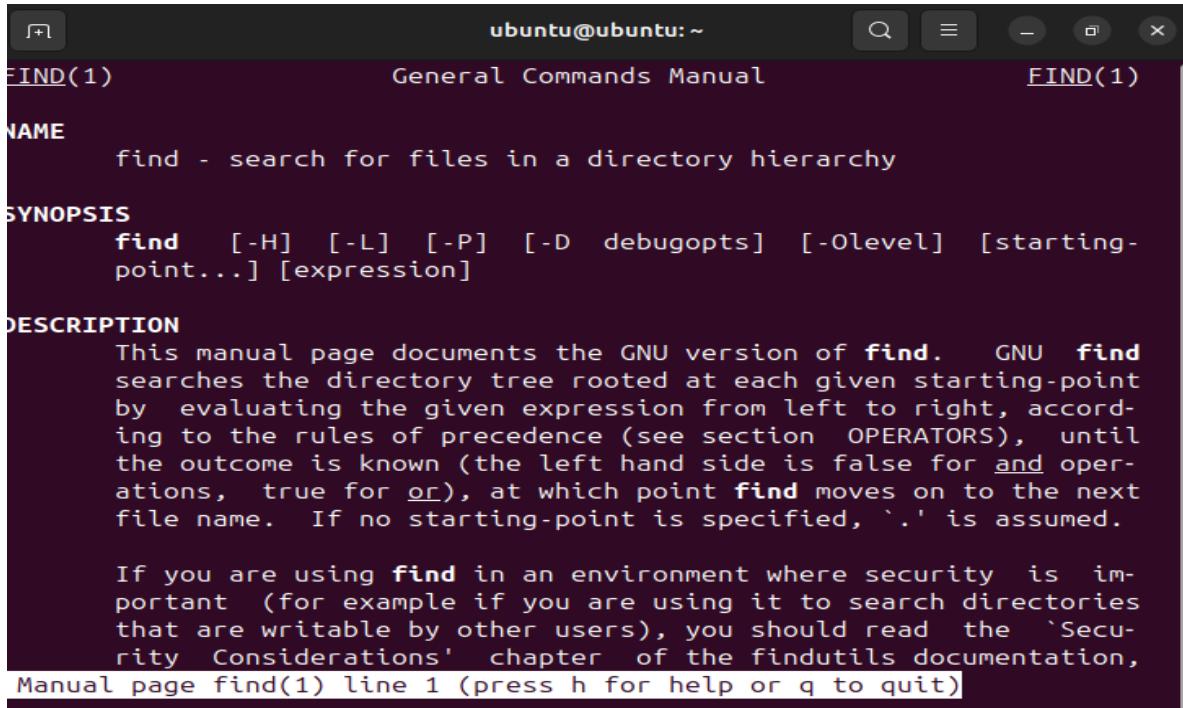
- Displays file contents one screen at a time, allowing for scrolling.

```
aafren
kahsa
fatima
Laraib
Laiba
hlo
karaci
|
```

Figure 3.1.36

3.1.37. Man

- Displays the manual page for a command.



```
FIND(1)           General Commands Manual          FIND(1)

NAME
    find - search for files in a directory hierarchy

SYNOPSIS
    find    [-H]   [-L]   [-P]   [-D  debugopts]   [-Olevel]   [starting-
                  point...] [expression]

DESCRIPTION
    This manual page documents the GNU version of find.  GNU find
    searches the directory tree rooted at each given starting-point
    by evaluating the given expression from left to right, accord-
    ing to the rules of precedence (see section OPERATORS), until
    the outcome is known (the left hand side is false for and oper-
    ations, true for or), at which point find moves on to the next
    file name.  If no starting-point is specified, `.' is assumed.

    If you are using find in an environment where security is im-
    portant (for example if you are using it to search directories
    that are writable by other users), you should read the `Secu-
    rity Considerations' chapter of the findutils documentation,
Manual page find(1) line 1 (press h for help or q to quit)
```

Figure 3.1.37

3.1.38. Output Redirection

- Redirects output from the command line to a file or another command.

```
ubuntu@ubuntu:~$ ls -1 fatima.txt
fatima.txt
```

Figure 3.1.38

```
ubuntu@ubuntu:~$ echo "hello">>output.txt
ubuntu@ubuntu:~$ ls
Desktop  Downloads  Pictures  Templates  fatima.txt  snap
Documents  Music      Public     Videos      output.txt
```

Figure 3.1.38

3.1.39. ls -l

- Displays detailed information about files and directories in a long format.

```
ubuntu@ubuntu:~$ ls -l
Desktop
Documents
Downloads
Music
Pictures
Public
Templates
Videos
fatima.txt
output.txt
snap
```

*Figure 3.1.39***3.1.40. Pwd**

- Prints the current working directory.

```
ubuntu@ubuntu:~$ pwd
/home/ubuntu
```

*Figure 3.1.40***3.1.41. touch hello.cpp**

- Creates an empty file or updates the timestamp of an existing file

```
ubuntu@ubuntu:~$ touch hlo.cpp
ubuntu@ubuntu:~$ ls
Desktop  Downloads  Pictures  Templates  fatima.txt  output.txt
Documents  Music      Public    Videos      hlo.cpp      snap
```

Figure 3.1.41

3.1.42. g++ hello.cpp -o test

- Compiles a C++ source file into an executable.

```
ubuntu@ubuntu:~$ g++ hlo.cpp -o test
```

*Figure 3.1.42***3.1.43. ./test**

- Executes the compiled program.

```
ubuntu@ubuntu:~$ ./test  
HELO ! WORLD
```

Figure 3.1.43

WEEK # 3

4. CODING IN UBUNTU WITH C++ PROGRAMMING LANGUAGE

4.1. SCENARIOS:

- 4.1.1. You are tasked with developing an interactive calculator that evaluates mathematical expressions while adhering to the BODMAS rule.

SOLUTION

Code

```
#include<iostream>
using namespace std;
int main ()
{
    int a,b,c,d,result;
    cout <<"Enter the value of a=";
    cin>>a;
    cout <<"Enter the value of b=";
    cin>>b;
    cout <<"Enter the value of c=";
    cin>>c;
    result=(a+b)/2*9+c+12*7;
    cout<<"The result of an expression="<<result<<endl;
    return 0;
}
```

Output

```
ubuntu@ubuntu:~$ g++ BODMAS.cpp -o test
ubuntu@ubuntu:~$ ./test
Enter the value of a=9
Enter the value of b=8
Enter the value of c=7
The result of an expression=163
ubuntu@ubuntu:~$
```

Figure 4.1.1

- 4.1.2. You are a teacher who wants to analyze the scores of students in a recent exam. You have collected the scores from all students, and now you want to find out: The highest score in the class (maximum).The lowest score in the class (minimum).

Solution**Code**

```
#include<iostream>
using namespace std;
int main()
{
    int n;
    cout<<"Enter the Number of Students:";
    cin>>n;
    int arr[n];
    for (int i=0;i<n;i++)
    {
        cout<<"Enter the marks of student " <<i<<":";
        cin>>arr[i];
    }
    int max=arr[0];
    int min=arr[0];
    for (int i=0;i<n;i++)
    {
        if (arr[i]>max)
```

```
if (arr[i]>max)
{
    max=arr[i];
}
if(arr[i]<min)
{
    min=arr[i];
}
cout<<"maximum number of marks are :"<<max<<endl;
cout<<"minimum number of marks are :"<<min<<endl;
}
```

Output

```
ubuntu@ubuntu:~$ g++ minmax.cpp -o test
ubuntu@ubuntu:~$ ./test
Enter the Number of Students:4
Enter the marks of student 0:12
Enter the marks of student 1:16
Enter the marks of student 2:19
Enter the marks of student 3:20
maximum number of marks are :20
minimum number of marks are :12
```

Figure 4.1.2

4.1.3. You are a data analyst working with a list of integers representing the daily sales of a product over a month. You need to perform the following tasks:

Calculate the sum of all even numbers in the list.

Calculate the result of subtracting all odd numbers in the list from an initial value of zero.

Code

```
#include<iostream>
using namespace std;
int main()
{
    int n;
    cout<<"Enter the Number:";
    cin>>n;
    int arr[n];
    for (int i=0;i<n;i++)
    {
        cout<<"Enter the Value of " <<i<<": number";
        cin>>arr[i];
    }
    int sum=0;
    int odd=0;
    for (int i=0;i<n;i++)
    {
        if (arr[i]%2==0)
```

```
{  
if (arr[i]%2==0)  
{  
sum=sum+arr[i];  
}  
}  
cout<<"the sum of even number:"<<sum<<endl;  
for (int i=0;i<n;i++)  
{  
if (arr[i]%2!=0)  
{  
odd=odd+arr[i];  
}  
}  
cout<<"sum of odd number :"<<odd<<endl;  
return 0;  
}
```

Output

```
ubuntu@ubuntu:~$ g++ evenodd.cpp -o test  
ubuntu@ubuntu:~$ ./test  
Enter the Number:6  
Enter the Value of 0: number 2  
Enter the Value of 1: number 4  
Enter the Value of 2: number7  
Enter the Value of 3: number 3  
Enter the Value of 4: number5  
Enter the Value of 5: number9  
the sum of even number:6  
sum of odd number :24
```

Figure 4.1.3

WEEK # 4

5. CODING IN UBUNTU TERMINAL

5.1.1. DECLARING VARIABLE

```
ubuntu@ubuntu:~$ myvar=hello
```

Figure 5.1.1

5.1.2. DISPLAYING THE VARIABLE VALUE

```
ubuntu@ubuntu:~$ echo $myvar  
hello
```

Figure 5.1.2

5.1.3. DECLARING A STRING

```
ubuntu@ubuntu:~$ myvar="i hate people"
```

Figure 5.1.3

5.1.4. DISPLAYING A STRING

```
ubuntu@ubuntu:~$ echo $myvar  
i hate people
```

Figure 5.1.4

5.1.5. DECLARING AN INTEGER

```
ubuntu@ubuntu:~$ myvar=8
```

Figure 5.1.5

5.1.6. DISPLAYING AN INTEGER

```
ubuntu@ubuntu:~$ echo $myvar
```

```
8
```

Figure 5.1.6

5.1.7. IF-THEN-ELSE STATEMENT

```
ubuntu@ubuntu:~$ count=100
ubuntu@ubuntu:~$ if [ $count -eq 100 ]
> then
> echo " count is 100"
> else
> echo " count is not 100 "
> fi
count is 100
ubuntu@ubuntu:~$ █
```

Figure 5.1.7

5.1.8. CASE STATEMENT

```
ubuntu@ubuntu:~$ option="A"
ubuntu@ubuntu:~$ case $option in
> "A") echo " Option A selected "
> ;;
> "B") echo " Option B selected "
> ;;
> "C") echo " Option C selected "
> ;;
> *) echo "INVALID OPTION "
> ;;
> esac
Option A selected
```

Figure 5.1.8

5.1.9. FOR LOOP

```
ubuntu@ubuntu:~$ for i in {1..7};do  
> echo "i hate people $i"  
> done  
i hate people 1  
i hate people 2  
i hate people 3  
i hate people 4  
i hate people 5  
i hate people 6  
i hate people 7  
ubuntu@ubuntu:~$
```

Figure 5.1.9

5.1.10. DO-WHILE LOOP

```
ubuntu@ubuntu:~$ count=0  
ubuntu@ubuntu:~$ while true; do  
> echo "count is $count"  
> count=$((count+1))  
> if [ $count -ge 5 ]; then  
> break  
> fi  
> done  
count is 0  
count is 1  
count is 2  
count is 3  
count is 4
```

Figure 5.1.10

5.1.11. WHILE LOOP

```
ubuntu@ubuntu:~$ count=1
ubuntu@ubuntu:~$ while [ $count -le 10 ]; do
> echo " i hate people $count"
> count=$((count+1))
> done
 i hate people 1
 i hate people 2
 i hate people 3
 i hate people 4
 i hate people 5
 i hate people 6
 i hate people 7
 i hate people 8
 i hate people 9
 i hate people 10
```

Figure 5.1.11

5.1.12. FILE COMPARING

```
ubuntu@ubuntu:~$ file1="aafreen.txt"
ubuntu@ubuntu:~$ file2="fatima.txt"
ubuntu@ubuntu:~$ if [ -f $file1 ] && [ -f $file2 ] ; then
> if cmp -s $file1 $file2 ; then
> echo "files are identical"
> else
> echo "files are different "
> fi
> else
> echo "one or both file do not exist"
> fi
one or both file do not exist
```

Figure 5.1.12

5.1.13. VARIABLE COMPARING

```
ubuntu@ubuntu:~$ count=10
ubuntu@ubuntu:~$ sum=10
ubuntu@ubuntu:~$ if [ $count -eq $sum ] ; then
> echo "they are same "
> else
> echo "they are different "
> fi
they are same
```

Figure 5.1.13

5.1.14. LOGICAL OPERATORS

```
ubuntu@ubuntu:~$ A=10
ubuntu@ubuntu:~$ B=20
ubuntu@ubuntu:~$ if [ $A -le 15 ] && [ $B -gt 15 ]; then
> echo " A IS LESS THEN 15 AND B IS GREATER THEN 15 "
> else
> echo "ONE OR BOTH CONDITION ARE FALSE "
> fi
 A IS LESS THEN 15 AND B IS GREATER THEN 15
ubuntu@ubuntu:~$ █
```

Figure 5.1.14

WEEK # 5

6. SYSTEM CALLS

System calls provide an interface between user programs and the operating system. They allow user-level processes to request services from the kernel, such as file operations, process control, and communication. Here are some common system calls related to process management in Unix-like operating systems

6.1. fork ()

The fork () system call is used to create a new process by duplicating the calling process. The new process, referred to as the child process, is a copy of the parent process, except for the returned value. The fork () call returns:

- 0 to the child process.
- The child's PID (process ID) to the parent process.
- -1 if the creation of the child process fails.

6.1.1. SIMPLE PROGRAM

CODE

```
#include <iostream>
#include<unistd.h>
#include<sys/types.h>
using namespace std;
int main ()
{
pid_t pid=fork();
if (pid==0){
cout<<"This is Child process"<<pid<<endl;
else if (pid>0)
{
cout<<"This is Parent process"<<pid<<endl;
}
else {
cout<<"fork failed";
return 0;
}
```

Output

```
ubuntu@ubuntu:~$ g++ fork.cpp -o test
ubuntu@ubuntu:~$ ./test
This is Parent process8580
This is Child process0
```

*Figure 6.1.1***6.1.2. Predict the Output of the following program:****6.1.2.1. Program**

```
#include      <stdio.h>
#include<sys/types.h>
#include  <unistd.h>    int
main()
{
    // make two process which run same
// program after this instruction    fork();
    printf("Hello world!\n");
    return 0;
}
```

Output

```
ubuntu@ubuntu:~$ g++ fork1.cpp -o test
./ubuntu@ubuntu:~$ ./test
hello world
hello world
ubuntu@ubuntu:~$
```

Figure 6.1.2.1

6.1.2.2. Program

```
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
int main()
{ // make two process which run same
// program after this instruction
fork();
fork();
fork();
printf("Hello world!\n");
return 0;
}
```

Output

```
ubuntu@ubuntu:~$ touch fork2.cpp
ubuntu@ubuntu:~$ g++ fork2.cpp -o test
ubuntu@ubuntu:~$ ./test
hello world
```

Figure 6.1.2.2

6.1.2.3. Program

```
#include <stdio.h>
#include <unistd.h>

int main(int argc, char **argv)
{
    printf("--beginning of program\n");

    int counter = 0;    pid_t pid = fork();

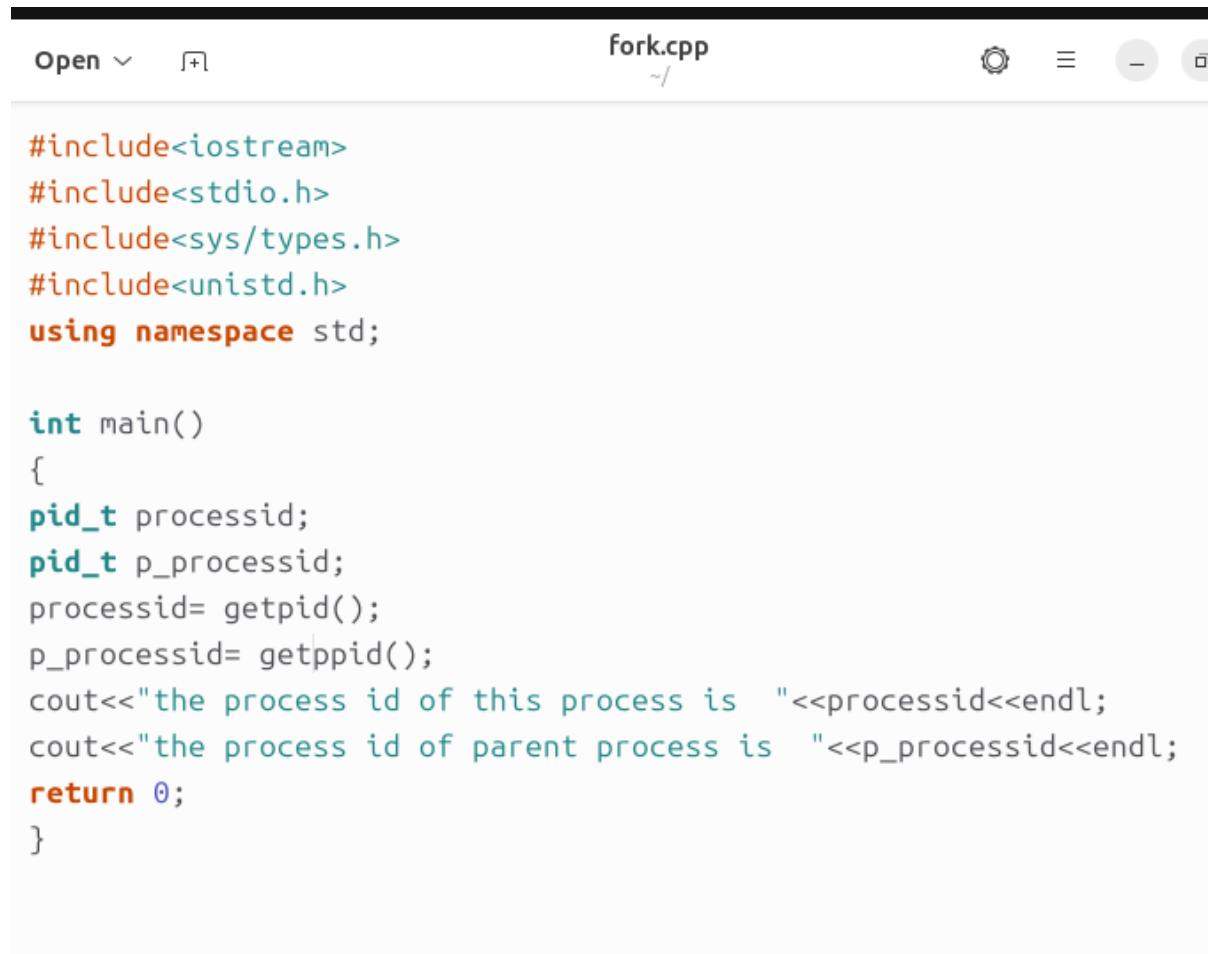
    if (pid == 0)
    {
        // child process      int i = 0;      for (; i < 5; ++i)
        {
            printf("child process: counter=%d\n", ++counter);
        }
    }
    else if (pid > 0)
    {
        // parent process     int j = 0;      for (; j < 5; ++j)
        {
            printf("parent process: counter=%d\n", ++counter);
        }
    }    }    else
    {
        // fork failed      printf("fork() failed!\n");      return 1;
    }
    printf("--end of program--\n");
    return 0; }
```

Output

```
ubuntu@ubuntu:~$ g++ fork1.cpp -o test
ubuntu@ubuntu:~$ ./test
parent process:counter = 0
parent process:counter = 1
parent process:counter = 2
parent process:counter = 3
parent process:counter = 4
child process:counter = 0
child process:counter = 1
child process:counter = 2
child process:counter = 3
child process:counter = 4
```

Figure 6.1.2.3

6.1.2.4. Program



```
fork.cpp
#include<iostream>
#include<stdio.h>
#include<sys/types.h>
#include<unistd.h>
using namespace std;

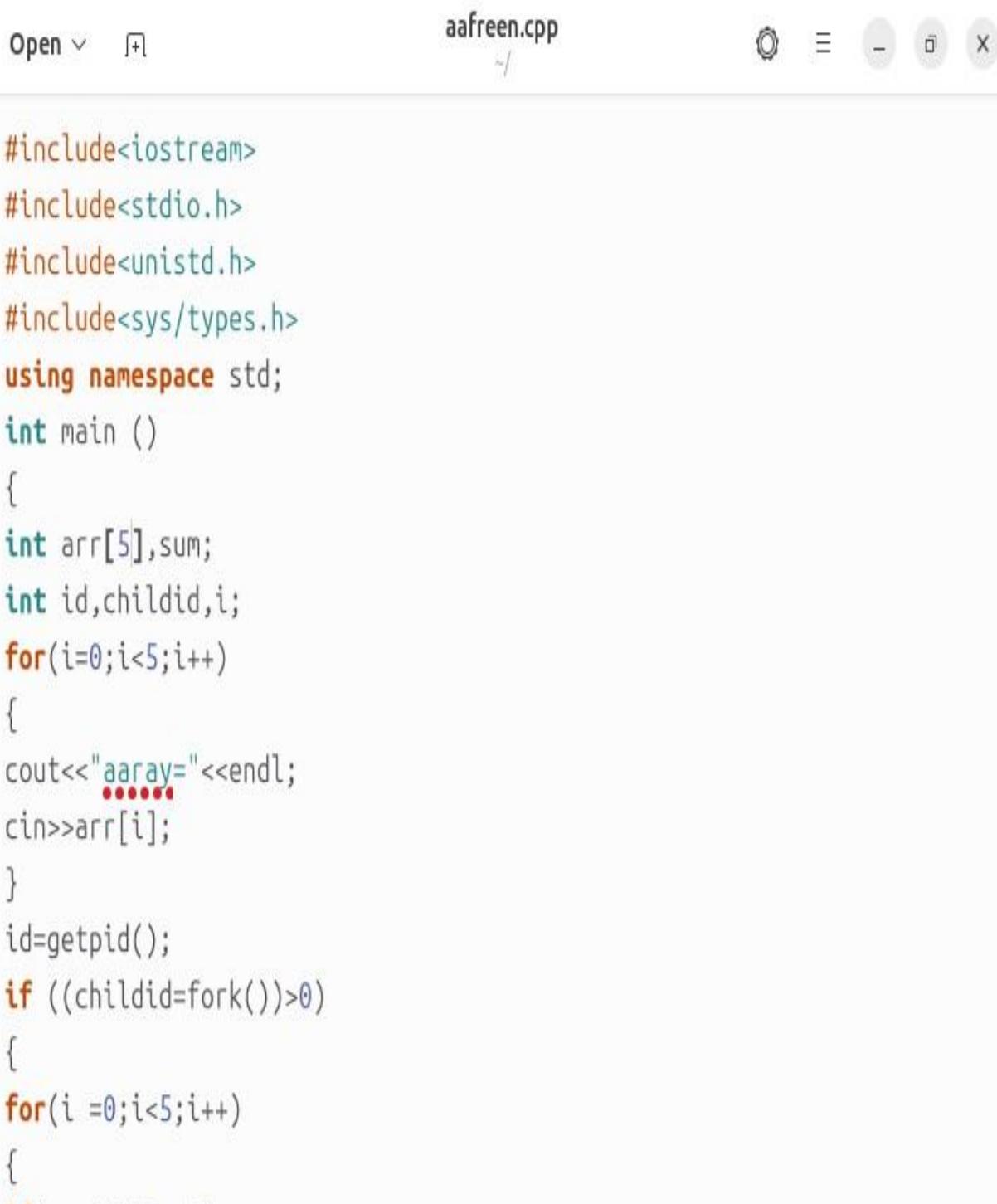
int main()
{
pid_t processid;
pid_t p_processid;
processid= getpid();
p_processid= getppid();
cout<<"the process id of this process is "<<processid<<endl;
cout<<"the process id of parent process is "<<p_processid<<endl;
return 0;
}
```

Output

```
|          getppid
ubuntu@ubuntu:~$ g++ fork.cpp -o test
ubuntu@ubuntu:~$ ./test
the process id of this process is 5401
the process id of parent process is 4170
ubuntu@ubuntu:~$ █
```

Figure 6.1.2.4

6.1.2.5. Write a program to find sum of even numbers in parent process and sum of odd numbers in child process.

CODE

The screenshot shows a code editor window with the following details:

- File name: **aafreen.cpp**
- File path: **~/**
- Toolbar icons: Open, Save, Minimize, Maximize, Close.

```
#include<iostream>
#include<stdio.h>
#include<unistd.h>
#include<sys/types.h>
using namespace std;
int main ()
{
    int arr[5],sum;
    int id,childid,i;
    for(i=0;i<5;i++)
    {
        cout<<"aaray=";
        .....  
        cin>>arr[i];
    }
    id=getpid();
    if ((childid=fork())>0)
    {
        for(i =0;i<5;i++)
        {
            .....
        }
    }
}
```

```
for(i =0;i<5;i++)
{
if(arr[i]%2==0)
{
sum=sum+arr[i];
cout<<"the sum of even="<<sum<<endl;
}
}
}

else if ((childid=fork())==0)
{
for( i =0;i<5;i++)
{
if(arr[i]%2 !=0)
{
sum= sum+arr[i];
cout<<"the sum of odd no ="<<sum<<endl;
}
}
}
```

OUTPUT

```
array=
7
array=
8
array=
9
the sum of even=6
the sum of even=14
the sum of odd no =5
the sum of odd no =12
the sum of odd no =21
```

Figure 6.1.2.5

6.2. Exit()

The exit function is used to terminate a program. It performs cleanup operations, such as flushing output buffers and closing open files. The function is declared in the <stdlib.h> header.

6.3. Wait()

The wait function makes the parent process wait until all of its child processes have terminated. It is used for process synchronization. The function is declared in the <sys/wait.h> header.

6.4. Sleep()

The sleep function suspends the execution of the calling process for a specified number of seconds. It is useful for delaying program execution. The function is declared in the <unistd.h> header.

Code

```
#include <iostream>
#include <unistd.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <cstdlib>
using namespace std;

int main() {
    pid_t pid = fork(); // Create a new process
    if (pid < 0) {
        // Fork failed
        cerr << "Fork failed!" << endl;
        return 1;
    }
    else if (pid == 0) {
        // Child process
        cout << "Child process (PID: " << getpid() << ") is sleeping for 3 seconds..." << endl;
        sleep(3); // Sleep for 3 seconds
        cout << "Child process (PID: " << getpid() << ") has finished sleeping." << endl;
        exit(0); // Exit child process
    }
    else {
        // Parent process
        cout << "Parent process (PID: " << getpid() << ") is waiting for child process (PID: " << pid << ") to finish..." << endl;
    }
}
```

```
    wait(NULL); // Wait for the child process to finish  
    cout << "Child process has finished. Parent process (PID: " << getpid() << ") is continuing." <<  
    endl;  
}  
return 0;  
}
```

Output

Output Clear

```
/tmp/dBxRhf1ntT.o  
Parent process (PID: 1098) is waiting for child process (PID: 1099) to  
    finish...  
Child process (PID: 1099) is sleeping for 3 seconds...  
Child process (PID: 1099) has finished sleeping.  
Child process has finished. Parent process (PID: 1098) is continuing.  
  
==== Code Execution Successful ====
```

Figure 6.4

WEEK # 6

7. FIRST COME FIRST SERVED (FCFS)

The **First Come First Served (FCFS)** scheduling algorithm is the simplest CPU scheduling method where processes are executed in the order of their arrival. It is non-preemptive, meaning once a process starts, it runs to completion without interruption. FCFS uses a First-In-First-Out (FIFO) queue structure, ensuring fairness since all processes are executed in the sequence they arrive. However, it suffers from drawbacks like the **convoy effect**, where a long process delays others, leading to inefficient CPU utilization and poor average turnaround time. Despite its simplicity and fairness, FCFS is rarely used in modern systems due to its limitations.

7.1. FCFS WITH SAME ARRIVAL TIME i.e. 0

Code

```
#include<iostream>

using namespace std;

// Function to find waiting time for all processes
void findWaitingTime(int processes[], int n, int bt[], int wt[]) {
    wt[0] = 0; // First process has no waiting time
    // Calculate waiting time for each process
    for (int i = 1; i < n; i++) {
        wt[i] = bt[i - 1] + wt[i - 1];
    }
}

// Function to calculate turnaround time for all processes
void findTurnaroundTime(int processes[], int n, int bt[], int wt[], int tat[]) {
    for (int i = 0; i < n; i++) {
        tat[i] = bt[i] + wt[i];
    }
}

// Function to calculate average time
void findAverageTime(int processes[], int n, int bt[]) {
    int wt[n], tat[n], total_wt = 0, total_tat = 0;
    findWaitingTime(processes, n, bt, wt);
    findTurnaroundTime(processes, n, bt, wt, tat);
    cout << "Processes " << "Burst time " << "Waiting time " << "Turnaround time\n";
    for (int i = 0; i < n; i++) {
```

```

total_wt += wt[i];
total_tat += tat[i];
cout << " " << i + 1 << "\t\t" << bt[i] << "\t " << wt[i] << "\t\t " << tat[i] << endl;
}

cout << "\nAverage waiting time = " << (float)total_wt / (float)n;
cout << "\nAverage turnaround time = " << (float)total_tat / (float)n;
}

int main() {
    int n;

    // Taking input for number of processes
    cout << "Enter number of processes: ";
    cin >> n;

    int processes[n], burst_time[n];

    // Taking burst time for each process
    for (int i = 0; i < n; i++) {
        cout << "Enter burst time for process " << i + 1 << ": ";
        cin >> burst_time[i];
        processes[i] = i + 1; // Process number
    }

    findAverageTime(processes, n, burst_time);

    return 0;
}

```

Output

Processes	Burst time	Waiting time	Turnaround time
1	24	0	24
2	3	24	27
3	4	27	31

Average waiting time = 17
Average turnaround time = 27.3333

Figure 7.1

7.2. FCFS WITH DIFFERENT ARRIVAL TIME

Code

```
#include <iostream>
#include <algorithm>
using namespace std;

struct Process {
    int pid;
    int arrival_time;
    int burst_time;
    int completion_time;
    int waiting_time;
    int turnaround_time;
};

// Comparator function to sort processes based on arrival time
bool compare(Process a, Process b) {
    return a.arrival_time < b.arrival_time;
}

void calculateCompletionTime(Process proc[], int n) {
    proc[0].completion_time = proc[0].arrival_time + proc[0].burst_time;
    for (int i = 1; i < n; i++) {
        // If the CPU is idle, wait for the process to arrive
        if (proc[i].arrival_time > proc[i-1].completion_time) {
            proc[i].completion_time = proc[i].arrival_time + proc[i].burst_time;
        } else {
            proc[i].completion_time = proc[i-1].completion_time + proc[i].burst_time;
        }
    }
}

void calculateTurnaroundTime(Process proc[], int n) {
    for (int i = 0; i < n; i++) {
```

```
proc[i].turnaround_time = proc[i].completion_time - proc[i].arrival_time;  
}  
}  
  
void calculateWaitingTime(Process proc[], int n) {  
    for (int i = 0; i < n; i++) {  
        proc[i].waiting_time = proc[i].turnaround_time - proc[i].burst_time;  
    }  
}  
  
void fcfs(Process proc[], int n) {  
    calculateCompletionTime(proc, n);  
    calculateTurnaroundTime(proc, n);  
    calculateWaitingTime(proc, n);  
}  
  
void printFcfs(Process proc[], int n) {  
    cout << "\t\t\tFCFS SCHEDULING\t\t\t\n";  
    cout << "PID\tArrival Time\tBurst Time\tCompletion Time\tTurnAround Time\tWaiting Time\n";  
    for (int i = 0; i < n; i++) {  
        cout << proc[i].pid << "\t\t" << proc[i].arrival_time << "\t\t" << proc[i].burst_time <<  
        "\t\t" << proc[i].completion_time << "\t\t" << proc[i].turnaround_time << "\t\t" <<  
        proc[i].waiting_time << "\n";  
    }  
}  
  
int main() {  
    int n = 5;  
    Process proc[5] = {{1, 2, 6}, {2, 5, 2}, {3, 1, 8}, {4, 8, 3}, {5, 4, 4}};  
    // Sort processes based on arrival time  
    sort(proc, proc + n, compare);  
    // Perform FCFS scheduling  
    fcfs(proc, n);  
    // Print the results
```

```
printFcfs(proc, n);  
return 0;  
}
```

Output

```
Select C:\Users\AAFREEN ZAHRA KAZMI\Downloads\fcfs.exe  
FCFS SCHEDULING  
PID    Arrival Time    Burst Time    Completion Time TurnAround Time Waiting Time  
3        1            8              9                8                  0  
1        2            6              15               13                 7  
5        4            4              19               15                 11  
2        5            2              21               16                 14  
4        8            3              24               16                 13  
-----  
Process exited after 0.1378 seconds with return value 0  
Press any key to continue . . .
```

Figure 7.2

8. SHORTEST JOB FIRST (SJF)

The **Shortest Job First (SJF)** scheduling algorithm selects the process with the shortest burst time for execution, aiming to minimize the average waiting and turnaround times. It can be implemented as **non-preemptive**, where a process runs to completion once started, or **preemptive** (known as Shortest Remaining Time First, SRTF), where a new process with a shorter burst time can interrupt the current one. SJF is optimal in terms of reducing average waiting time but requires precise knowledge of process burst times, which can be difficult to predict. This limitation makes it less practical for real-time systems.

8.1. SJF WITH SAME ARRIVAL TIME i.e. 0

Code

```
#include <iostream>
#include <algorithm>
using namespace std;

// Structure for process
struct Process {
    int id; // Process ID
    int bt; // Burst Time
    int at; // Arrival Time
};

// Function to find waiting time for all processes
void findWaitingTime(Process proc[], int n, int wt[]) {
    int service_time[n];
    service_time[0] = proc[0].at;
    wt[0] = 0; // First process doesn't wait

    // Calculating waiting time
    for (int i = 1; i < n; i++) {
        service_time[i] = service_time[i - 1] + proc[i - 1].bt;
        wt[i] = service_time[i] - proc[i].at;

        // If waiting time for a process is negative, set it to zero
        if (wt[i] < 0) {
            wt[i] = 0;
        }
    }
}
```

```

}

// Function to calculate turnaround time for all processes

void findTurnaroundTime(Process proc[], int n, int wt[], int tat[]) {
    // Turnaround time = burst time + waiting time
    for (int i = 0; i < n; i++) {
        tat[i] = proc[i].bt + wt[i];
    }
}

// Function to calculate average time

void findAverageTime(Process proc[], int n) {
    int wt[n], tat[n], total_wt = 0, total_tat = 0;
    // Calculate waiting time for all processes
    findWaitingTime(proc, n, wt);
    // Calculate turnaround time for all processes
    findTurnaroundTime(proc, n, wt, tat);
    // Display processes along with burst time, waiting time, and turnaround time
    cout << "Processes " << "Burst time " << "Arrival time " << "Waiting time " << "Turnaround
time\n";
    for (int i = 0; i < n; i++) {
        total_wt += wt[i];
        total_tat += tat[i];
        cout << " " << proc[i].id << "\t\t" << proc[i].bt << "\t\t" << proc[i].at
        << "\t\t" << wt[i] << "\t\t" << tat[i] << endl;
    }
    cout << "\nAverage waiting time = " << (float)total_wt / (float)n;
    cout << "\nAverage turnaround time = " << (float)total_tat / (float)n;
}

// Function to implement SJF scheduling

void sjfScheduling(Process proc[], int n) {
    // Sort processes by burst time, if two processes have same burst time, sort by arrival time
    sort(proc, proc + n, [](Process a, Process b) {
        return (a.bt == b.bt) ? (a.at < b.at) : (a.bt < b.bt);
    });
}

```

```

// Calculate average waiting and turnaround time
findAverageTime(proc, n);}

int main() {
    int n;
    // Taking input for number of processes
    cout << "Enter number of processes: ";
    cin >> n;
    Process proc[n];
    // Taking burst time and arrival time for each process
    for (int i = 0; i < n; i++) {
        proc[i].id = i + 1;
        cout << "Enter burst time for process " << i + 1 << ": ";
        cin >> proc[i].bt;
        cout << "Enter arrival time for process " << i + 1 << ": ";
        cin >> proc[i].at;
    }
    // Perform SJF scheduling
    sjfScheduling(proc, n);
    return 0;
}

```

Output

```

C:\Users\AAFREEN ZAHRA KAZMI\Downloads\shortestjobfirst.exe
Enter number of processes: 4
Enter burst time for process 1: 24
Enter arrival time for process 1: 0
Enter burst time for process 2: 27
Enter arrival time for process 2: 0
Enter burst time for process 3: 2
Enter arrival time for process 3: 0
Enter burst time for process 4: 19
Enter arrival time for process 4: 0
Processes    Burst time    Arrival time    Waiting time    Turnaround time
  3           2                 0                  0                   2
  4           19                0                  2                 21
  1           24                0                 21                 45
  2           27                0                 45                 72
Average waiting time = 17
Average turnaround time = 35
-----
Process exited after 21.56 seconds with return value 0
Press any key to continue . . .

```

Figure 8.1

8.2. SJF WITH DIFFERENT ARRIVAL TIME

```
#include <iostream>
#include <algorithm>
using namespace std;

// Structure for process
struct Process {
    int id;    // Process ID
    int bt;    // Burst Time
    int at;    // Arrival Time
};

// Function to find waiting time for all processes
void findWaitingTime(Process proc[], int n, int wt[]) {
    int service_time[n];
    service_time[0] = proc[0].at;
    wt[0] = 0; // First process doesn't wait
    // Calculating waiting time
    for (int i = 1; i < n; i++) {
        service_time[i] = service_time[i - 1] + proc[i - 1].bt;
        wt[i] = service_time[i] - proc[i].at;
        // If waiting time for a process is negative, set it to zero
        if (wt[i] < 0) {
            wt[i] = 0;
        }
    }
}

// Function to calculate turnaround time for all processes
void findTurnaroundTime(Process proc[], int n, int wt[], int tat[]) {
    // Turnaround time = burst time + waiting time
    for (int i = 0; i < n; i++) {
        tat[i] = proc[i].bt + wt[i];
    }
}
```

```
// Function to calculate average time
void findAverageTime(Process proc[], int n) {
    int wt[n], tat[n], total_wt = 0, total_tat = 0;
    // Calculate waiting time for all processes
    findWaitingTime(proc, n, wt);
    // Calculate turnaround time for all processes
    findTurnaroundTime(proc, n, wt, tat);
    // Display processes along with burst time, waiting time, and turnaround time
    cout << "Processes " << "Burst time " << "Arrival time " << "Waiting time " << "Turnaround
time\n";
    for (int i = 0; i < n; i++) {
        total_wt += wt[i];
        total_tat += tat[i];
        cout << " " << proc[i].id << "\t\t" << proc[i].bt << "\t\t" << proc[i].at
            << "\t\t" << wt[i] << "\t\t" << tat[i] << endl;
    }
    cout << "\nAverage waiting time = " << (float)total_wt / (float)n;
    cout << "\nAverage turnaround time = " << (float)total_tat / (float)n;
}

// Function to implement SJF scheduling
void sjfScheduling(Process proc[], int n) {
    // Sort processes by burst time, if two processes have same burst time, sort by arrival time
    sort(proc, proc + n, [](Process a, Process b) {
        return (a.bt == b.bt) ? (a.at < b.at) : (a.bt < b.bt);
    });
    // Calculate average waiting and turnaround time
    findAverageTime(proc, n);
}

int main() {
    int n;
    // Taking input for number of processes
    cout << "Enter number of processes: ";
    cin >> n;
```

```
Process proc[n];  
// Taking burst time and arrival time for each process  
for (int i = 0; i < n; i++) {  
    proc[i].id = i + 1;  
    cout << "Enter burst time for process " << i + 1 << ":";  
    cin >> proc[i].bt;  
    cout << "Enter arrival time for process " << i + 1 << ":";  
    cin >> proc[i].at;  
}  
  
// Perform SJF scheduling  
sjfScheduling(proc, n);  
return 0;  
}
```

OUTPUT

Select C:\Users\AAFREEN ZAHRA KAZMI\Downloads\shortestjobfirst.exe

```
Enter number of processes: 4  
Enter burst time for process 1: 34  
Enter arrival time for process 1: 0  
Enter burst time for process 2: 9  
Enter arrival time for process 2: 3  
Enter burst time for process 3: 9  
Enter arrival time for process 3: 4  
Enter burst time for process 4: 12  
Enter arrival time for process 4: 7  
Processes  Burst time  Arrival time  Waiting time  Turnaround time  
      2          9            3              0                9  
      3          9            4              8                17  
      4         12            7             14                26  
      1         34            0             33                67  
  
Average waiting time = 13.75  
Average turnaround time = 29.75  
-----  
Process exited after 25.36 seconds with return value 0  
Press any key to continue . . .
```

Figure 8.2

WEEK # 7

9. PRIORITY SCHEDULLING (NON-PREEMPTIVE)

Priority Scheduling (Non-Preemptive) is a CPU scheduling algorithm where each process is assigned a priority, and the process with the highest priority (lowest numerical value) is executed first. If two processes have the same priority, they are scheduled based on their arrival order. In this non-preemptive approach, once a process starts execution, it runs to completion before another process is selected. While it ensures that critical tasks are addressed first, it can lead to **starvation**, where low-priority processes are indefinitely delayed if high-priority processes keep arriving. Aging techniques can be used to mitigate this issue.

9.1. PS WITH DIFFERENT ARRIVAL TIME

CODE

```
#include <iostream>
#include <algorithm>
using namespace std;

struct Process {
    int pid;          // Process ID
    int arrival_time; // Arrival Time
    int burst_time;   // Burst Time
    int completion_time; // Completion Time
    int waiting_time; // Waiting Time
    int turnaround_time; // Turnaround Time
    int priority;     // Priority (1 is the highest priority)
};

// Comparator function to sort processes based on priority and arrival time
bool compare(Process a, Process b) {
    if (a.priority == b.priority) {
        return a.arrival_time < b.arrival_time; // If priorities are the same, sort by arrival time
    }
    return a.priority < b.priority; // Sort by priority
}

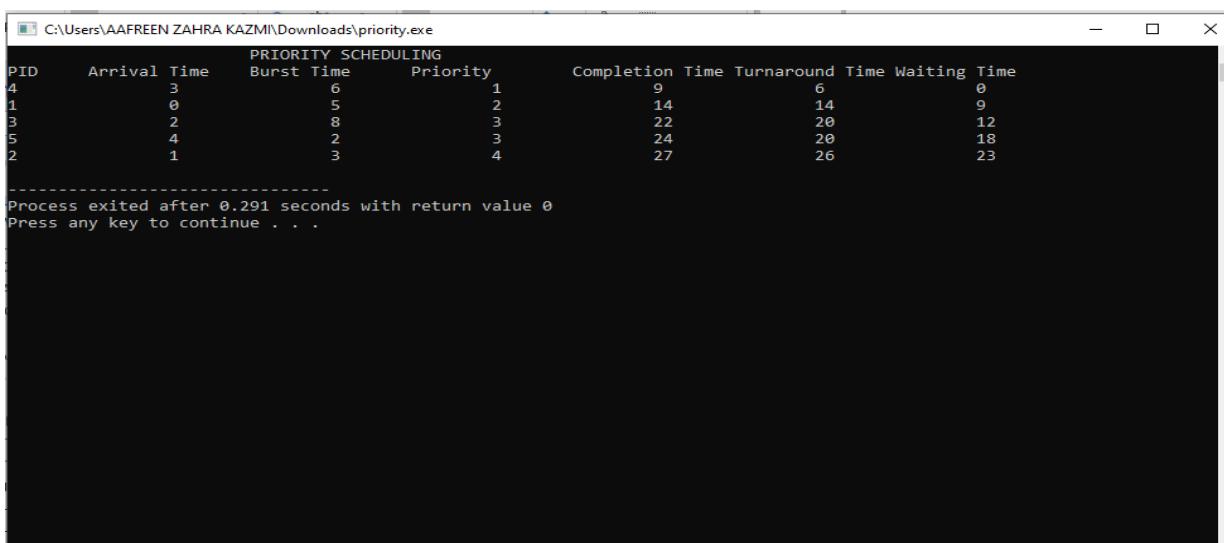
void calculateCompletionTime(Process proc[], int n) {
    proc[0].completion_time = proc[0].arrival_time + proc[0].burst_time;
    for (int i = 1; i < n; i++) {

```

```
if (proc[i].arrival_time > proc[i-1].completion_time) {  
    proc[i].completion_time = proc[i].arrival_time + proc[i].burst_time;  
}  
else {  
    proc[i].completion_time = proc[i-1].completion_time + proc[i].burst_time;  
}  
}  
}  
  
void calculateTurnaroundTime(Process proc[], int n) {  
    for (int i = 0; i < n; i++) {  
        proc[i].turnaround_time = proc[i].completion_time - proc[i].arrival_time;  
    }  
}  
  
void calculateWaitingTime(Process proc[], int n) {  
    for (int i = 0; i < n; i++) {  
        proc[i].waiting_time = proc[i].turnaround_time - proc[i].burst_time;  
    }  
}  
  
void priorityScheduling(Process proc[], int n) {  
    // Sort processes based on priority and arrival time  
    sort(proc, proc + n, compare);  
    calculateCompletionTime(proc, n);  
    calculateTurnaroundTime(proc, n);  
    calculateWaitingTime(proc, n);  
}  
  
void printPrioritySchedule(Process proc[], int n) {  
    cout << "\t\t\tPRIORITY SCHEDULING\t\t\t\n";  
    cout << "PID\tArrival Time\tBurst Time\tPriority\tCompletion Time\tTurnaround Time\tWaiting Time\n";  
    for (int i = 0; i < n; i++) {  
}
```

```
cout << proc[i].pid << "\t\t" << proc[i].arrival_time << "\t\t" << proc[i].burst_time <<
"\t\t" << proc[i].priority << "\t\t" << proc[i].completion_time << "\t\t" <<
proc[i].turnaround_time << "\t\t" << proc[i].waiting_time << "\n";  
}  
}  
  
int main() {  
    int n = 5;  
  
    Process proc[5] = {  
        {1, 0, 5, 0, 0, 0, 2}, // {PID, Arrival Time, Burst Time, Priority}  
        {2, 1, 3, 0, 0, 0, 4},  
        {3, 2, 8, 0, 0, 0, 3},  
        {4, 3, 6, 0, 0, 0, 1},  
        {5, 4, 2, 0, 0, 0, 3}  
    };  
  
    // Perform Priority Scheduling  
    priorityScheduling(proc, n);  
  
    // Print the results  
    printPrioritySchedule(proc, n);  
  
    return 0;  
}
```

OUTPUT



PRIORITY SCHEDULING						
PID	Arrival Time	Burst Time	Priority	Completion Time	Turnaround Time	Waiting Time
4	3	6	1	9	6	0
1	0	5	2	14	14	9
3	2	8	3	22	20	12
5	4	2	3	24	20	18
2	1	3	4	27	26	23

Process exited after 0.291 seconds with return value 0
Press any key to continue . . .

Figure 9.1

9.2. PS WITH SAME ARRIVAL TIME i.e. 0

Code

```
#include <iostream>
#include <algorithm>
using namespace std;

struct Process {
    int pid;          // Process ID
    int arrival_time; // Arrival Time
    int burst_time;   // Burst Time
    int completion_time; // Completion Time
    int waiting_time; // Waiting Time
    int turnaround_time; // Turnaround Time
    int priority;     // Priority (1 is the highest priority)
};

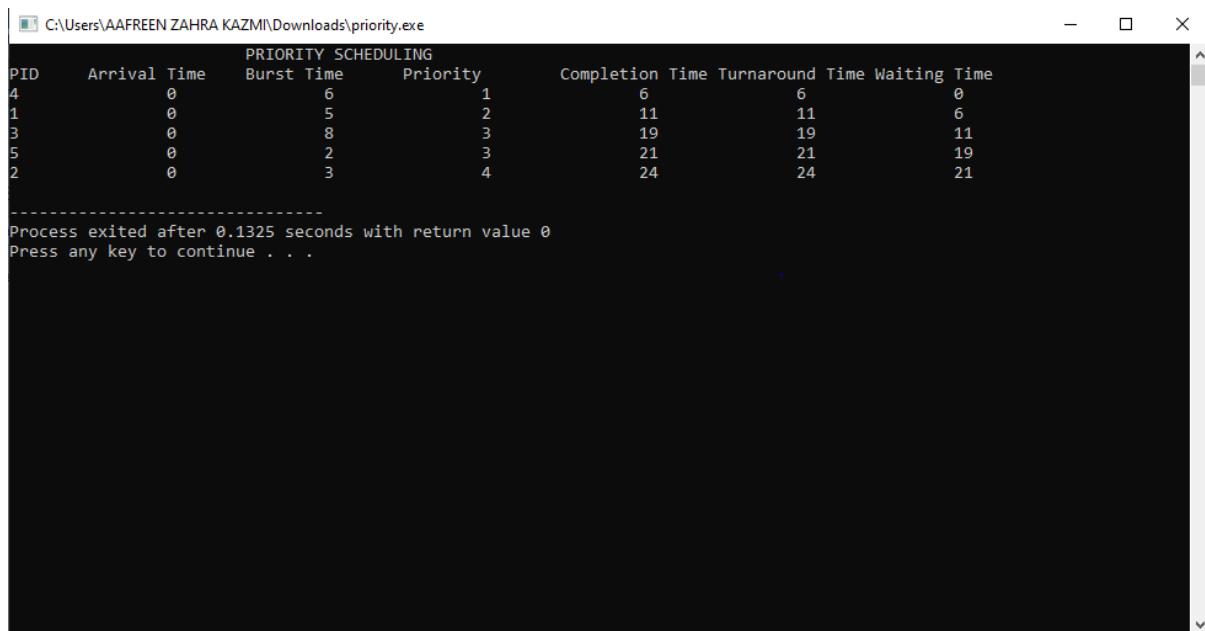
// Comparator function to sort processes based on priority and arrival time
bool compare(Process a, Process b) {
    if (a.priority == b.priority) {
        return a.arrival_time < b.arrival_time; // If priorities are the same, sort by arrival time
    }
    return a.priority < b.priority; // Sort by priority
}

void calculateCompletionTime(Process proc[], int n) {
    proc[0].completion_time = proc[0].arrival_time + proc[0].burst_time;
    for (int i = 1; i < n; i++) {
        if (proc[i].arrival_time > proc[i-1].completion_time) {
            proc[i].completion_time = proc[i].arrival_time + proc[i].burst_time;
        } else {
            proc[i].completion_time = proc[i-1].completion_time + proc[i].burst_time;
        }
    }
}
```

```
void calculateTurnaroundTime(Process proc[], int n) {  
    for (int i = 0; i < n; i++) {  
        proc[i].turnaround_time = proc[i].completion_time - proc[i].arrival_time;  
    }  
}  
  
void calculateWaitingTime(Process proc[], int n) {  
    for (int i = 0; i < n; i++) {  
        proc[i].waiting_time = proc[i].turnaround_time - proc[i].burst_time;  
    }  
}  
  
void priorityScheduling(Process proc[], int n) {  
    // Sort processes based on priority and arrival time  
    sort(proc, proc + n, compare);  
    calculateCompletionTime(proc, n);  
    calculateTurnaroundTime(proc, n);  
    calculateWaitingTime(proc, n);  
}  
  
void printPrioritySchedule(Process proc[], int n) {  
    cout << "\t\t\tPRIORITY SCHEDULING\t\t\t\n";  
    cout << "PID\tArrival Time\tBurst Time\tPriority\tCompletion Time\tTurnaround Time\tWaiting Time\n";  
    for (int i = 0; i < n; i++) {  
        cout << proc[i].pid << "\t\t" << proc[i].arrival_time << "\t\t" << proc[i].burst_time << "\t\t" << proc[i].priority << "\t\t" << proc[i].completion_time << "\t\t" << proc[i].turnaround_time << "\t\t" << proc[i].waiting_time << "\n";  
    }  
}  
  
int main() {  
    int n = 5;  
    Process proc[5] = {  
        {1, 0, 5, 0, 0, 0, 2}, // {PID, Arrival Time, Burst Time, Priority}  
    };
```

```
{2, 0, 3, 0, 0, 0, 4},  
{3, 0, 8, 0, 0, 0, 3},  
{4, 0, 6, 0, 0, 0, 1},  
{5, 0, 2, 0, 0, 0, 3}  
};  
  
// Perform Priority Scheduling  
priorityScheduling(proc, n);  
  
// Print the results  
printPrioritySchedule(proc, n);  
  
return 0;  
}
```

Output



The screenshot shows a terminal window titled 'C:\Users\AAFREEN ZAHRA KAZMI\Downloads\priority.exe'. The window displays the results of a priority scheduling algorithm for five processes (PID 4, 1, 3, 5, 2) based on their arrival time, burst time, and priority. The columns in the table are PID, Arrival Time, Burst Time, Priority, Completion Time, Turnaround Time, Waiting Time, and a header row for the first four. Process 4 has the highest priority (1) and shortest burst time (6), while Process 2 has the lowest priority (4) and longest burst time (3). The turnaround time is calculated as completion time minus arrival time, and waiting time is calculated as turnaround time minus burst time.

PRIORITY SCHEDULING							
PID	Arrival Time	Burst Time	Priority	Completion Time	Turnaround Time	Waiting Time	
4	0	6	1	6	6	0	
1	0	5	2	11	11	6	
3	0	8	3	19	19	11	
5	0	2	3	21	21	19	
2	0	3	4	24	24	21	

Process exited after 0.1325 seconds with return value 0
Press any key to continue . . .

Figure 9.2

10. SHORTEST REMAINING TIME FIRST (PREEMPTIVE) (SRTF) Code

```
#include <iostream>
#include <climits>
using namespace std;

struct Process {
    int pid;          // Process ID
    int arrival_time; // Arrival Time
    int burst_time;   // Burst Time
    int remaining_time; // Remaining Burst Time
    int completion_time; // Completion Time
    int waiting_time; // Waiting Time
    int turnaround_time; // Turnaround Time
};

void calculateTimes(Process proc[], int n) {
    int time = 0; // Current time
    int completed = 0; // Number of processes completed
    int shortest_remaining_time;
    int idx;
    // Initialize remaining burst times
    for (int i = 0; i < n; i++) {
        proc[i].remaining_time = proc[i].burst_time;
    }
    while (completed != n) {
        idx = -1;
        shortest_remaining_time = INT_MAX;
        // Find the process with the shortest remaining burst time that has arrived
        for (int i = 0; i < n; i++) {
            if (proc[i].arrival_time <= time && proc[i].remaining_time > 0) {
                if (proc[i].remaining_time < shortest_remaining_time) {
                    shortest_remaining_time = proc[i].remaining_time;
                    idx = i;
                }
            }
        }
        if (idx != -1) {
            time += proc[idx].remaining_time;
            proc[idx].completion_time = time;
            proc[idx].turnaround_time = time - proc[idx].arrival_time;
            proc[idx].waiting_time = time - proc[idx].arrival_time - proc[idx].burst_time;
            completed++;
        }
    }
}
```

```

        }
    }
}

if (idx != -1) {
    proc[idx].remaining_time--;
    time++;
    if (proc[idx].remaining_time == 0) {
        completed++;
        proc[idx].completion_time = time;
        proc[idx].turnaround_time = proc[idx].completion_time - proc[idx].arrival_time;
        proc[idx].waiting_time = proc[idx].turnaround_time - proc[idx].burst_time;
    }
} else {
    time++;
}
}

void printSchedule(Process proc[], int n) {
    cout << "\t\t\tPREEMPTIVE SJF SCHEDULING\t\t\t\n";
    cout << "PID\tArrival Time\tBurst Time\tCompletion Time\tTurnaround Time\tWaiting Time\n";
    for (int i = 0; i < n; i++) {
        cout << proc[i].pid << "\t\t" << proc[i].arrival_time << "\t\t" << proc[i].burst_time << "\t\t" <<
        proc[i].completion_time << "\t\t" << proc[i].turnaround_time << "\t\t" << proc[i].waiting_time <<
        "\n";
    }
}

int main() {
    int n = 5;
    Process proc[5] = {
        {1, 0, 7, 0, 0, 0, 0}, // {PID, Arrival Time, Burst Time, Remaining Time, Completion Time,
        Waiting Time, Turnaround Time}
        {2, 2, 4, 0, 0, 0, 0},
        {3, 4, 1, 0, 0, 0, 0},

```

```
{4, 5, 4, 0, 0, 0, 0},  
{5, 6, 2, 0, 0, 0, 0}  
};  
// Calculate scheduling times  
calculateTimes(proc, n);  
// Print the schedule  
printSchedule(proc, n);  
return 0;  
}
```

Output

PID	Arrival Time	Burst Time	Completion Time	Turnaround Time	Waiting Time
1	0	7	18	18	11
2	2	4	7	5	1
3	4	1	5	1	0
4	5	4	13	8	4
5	6	2	9	3	1

Process exited after 0.1154 seconds with return value 0
Press any key to continue . . .

Figure 10

11. ROUND ROBIN (PREEMPTIVE)

11.1. RR WITH SAME ARRIVAL TIME

Code

```
#include <iostream>

using namespace std;

// Function to find the waiting time for all processes
void findWaitingTime(int processes[], int n, int bt[], int wt[], int quantum) {
    int rem_bt[n];
    for (int i = 0; i < n; i++)
        rem_bt[i] = bt[i];
    int t = 0; // Current time
    while (true) {
        bool done = true;
        // Traverse all processes one by one repeatedly
        for (int i = 0; i < n; i++) {
            if (rem_bt[i] > 0) {
                done = false; // There is a pending process
                if (rem_bt[i] > quantum) {
                    t += quantum;
                    rem_bt[i] -= quantum;
                } else {
                    t += rem_bt[i];
                    wt[i] = t - bt[i];
                    rem_bt[i] = 0;
                }
            }
        }
        if (done) // If no process is left
            break;
    }
}

// Function to calculate turn around time
void findTurnAroundTime(int processes[], int n, int bt[], int wt[], int tat[]) {
    for (int i = 0; i < n; i++)
```

```
    tat[i] = bt[i] + wt[i];}

// Function to calculate average time

void findavgTime(int processes[], int n, int bt[], int quantum) {

    int wt[n], tat[n], total_wt = 0, total_tat = 0;

    findWaitingTime(processes, n, bt, wt, quantum);

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

    cout << "PN\tBT\tWT\tTAT\n";

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

        total_wt += wt[i];

        total_tat += tat[i];

        cout << " " << processes[i] << "\t" << bt[i] << "\t" << wt[i] << "\t" << tat[i] << endl;

    }

    cout << "Average waiting time = " << (float)total_wt / (float)n;

    cout << "\nAverage turn around time = " << (float)total_tat / (float)n;

}

// Driver code

int main() {

    int processes[] = {1, 2, 3};

    int n = sizeof(processes) / sizeof(processes[0]);



    int burst_time[] = {10, 5, 8};

    int quantum = 2;

    findavgTime(processes, n, burst_time, quantum);

    return 0;

}
```

Output

```
C:\Users\AAFREEN ZAHRA KAZMI\Downloads\roundrobin.exe
PN      BT      WT      TAT
1       10      13      23
2       5       10      15
3       8       13      21
Average waiting time = 12
Average turn around time = 19.6667
-----
Process exited after 0.1243 seconds with return value 0
Press any key to continue . . .
```

Figure 11.1

11.2. RR WITH DIFFERENT ARRIVAL TIME

Code

```
#include <iostream>
using namespace std;

struct Process {
    int pid;          // Process ID
    int arrival_time; // Arrival Time
    int burst_time;   // Burst Time
    int remaining_time; // Remaining Burst Time
    int completion_time; // Completion Time
    int waiting_time; // Waiting Time
    int turnaround_time; // Turnaround Time
};

// Function to find the waiting time for all processes
void findWaitingTime(Process proc[], int n, int quantum) {
    int t = 0; // Current time
    int rem_bt[n];
    int arrival[n];
    for (int i = 0; i < n; i++) {
        rem_bt[i] = proc[i].burst_time;
        arrival[i] = proc[i].arrival_time;
    }
    while (true) {
        bool done = true;
        for (int i = 0; i < n; i++) {
            if (arrival[i] <= t && rem_bt[i] > 0) {
                done = false;
                if (rem_bt[i] > quantum) {
                    t += quantum;
                    rem_bt[i] -= quantum;
                } else {
                    t += rem_bt[i];
                    rem_bt[i] = 0;
                }
            }
        }
        if (done)
            break;
    }
}
```

```
proc[i].waiting_time = t - proc[i].arrival_time - proc[i].burst_time;
rem_bt[i] = 0;
}
}
}
if (done)
    break;
bool all_arrived = true;
for (int i = 0; i < n; i++) {
    if (arrival[i] > t) {
        all_arrived = false;
        break;
    }
}
if (!all_arrived) {
    t++;
}
}

// Function to calculate turn around time
void findTurnAroundTime(Process proc[], int n) {
    for (int i = 0; i < n; i++)
        proc[i].turnaround_time = proc[i].burst_time + proc[i].waiting_time;
}

// Function to calculate average time
void findavgTime(Process proc[], int n, int quantum) {
    findWaitingTime(proc, n, quantum);
    findTurnAroundTime(proc, n);
    cout << "PN\tArrival Time\tBurst Time\tWaiting Time\tTurnaround Time\n";
    int total_wt = 0, total_tat = 0;
    for (int i = 0; i < n; i++) {
        total_wt += proc[i].waiting_time;
```

```
total_tat += proc[i].turnaround_time;

cout << " " << proc[i].pid << "\t" << proc[i].arrival_time << "\t\t" << proc[i].burst_time << "\t\t"
<< proc[i].waiting_time << "\t\t" << proc[i].turnaround_time << endl;

}

cout << "\nAverage waiting time = " << (float)total_wt / (float)n;

cout << "\nAverage turnaround time = " << (float)total_tat / (float)n;

}

int main() {

    Process proc[] = {{1, 0, 10}, {2, 1, 5}, {3, 2, 8}};

    int n = sizeof(proc) / sizeof(proc[0]);

    int quantum = 2;

    findavgTime(proc, n, quantum);

    return 0;
}
```

Output

```
C:\Users\AAFREEN ZAHRA KAZMI\Downloads\rourndrbindiffarrivaltime.exe

PN      Arrival Time      Burst Time      Waiting Time      Turnaround Time
1        0                  10                13                23
2        1                  5                 9                 14
3        2                  8                 11                19

Average waiting time = 11
Average turnaround time = 18.6667
-----
Process exited after 0.2247 seconds with return value 0
Press any key to continue . . .
```

Figure 11.2

WEEK # 8

12. DEADLOCK DETECTION

12.1. SHOW IF THE SYSTEM IS IN DEADLOCK OR NOT?

CODE

```
#include <stdio.h>

int max[100][100];
int alloc[100][100];
int need[100][100];
int avail[100];
int n, r;
void input();
void show();
void cal();
int main() {
    printf("**Deadlock Detection Algorithm***\n");
    input();
    show();
    cal();
    return 0;
}
void input() {
    printf("Enter The No of processes: ");
    scanf("%d", &n);
    printf("Enter No of Resource Instances: ");
    scanf("%d", &r);
    printf("Enter The Max Matrix\n");
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < r; j++) {
            scanf("%d", &max[i][j]);
        }
    }
    printf("Enter The Allocation Matrix\n");
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < r; j++) {
```

```
    scanf("%d", &alloc[i][j]);  
}  
}  
printf("Enter The Available Resources\n");  
for (int j = 0; j < r; j++) {  
    scanf("%d", &avail[j]);  
}  
}  
  
void show() {  
    printf("Process\tAllocation\t Max\t\tAvailable\n");  
    for (int i = 0; i < n; i++) {  
        printf("%d\t", i + 1);  
        for (int j = 0; j < r; j++) {  
            printf("%d\t ", alloc[i][j]);  
        }  
        printf("\t");  
        for (int j = 0; j < r; j++) {  
            printf("%d \t", max[i][j]);  
        }  
        printf("\t");  
        if (i == 0) {  
            for (int j = 0; j < r; j++) {  
                printf("%d \t", avail[j]);  
            }  
        }  
        printf("\n");  
    }  
}
```

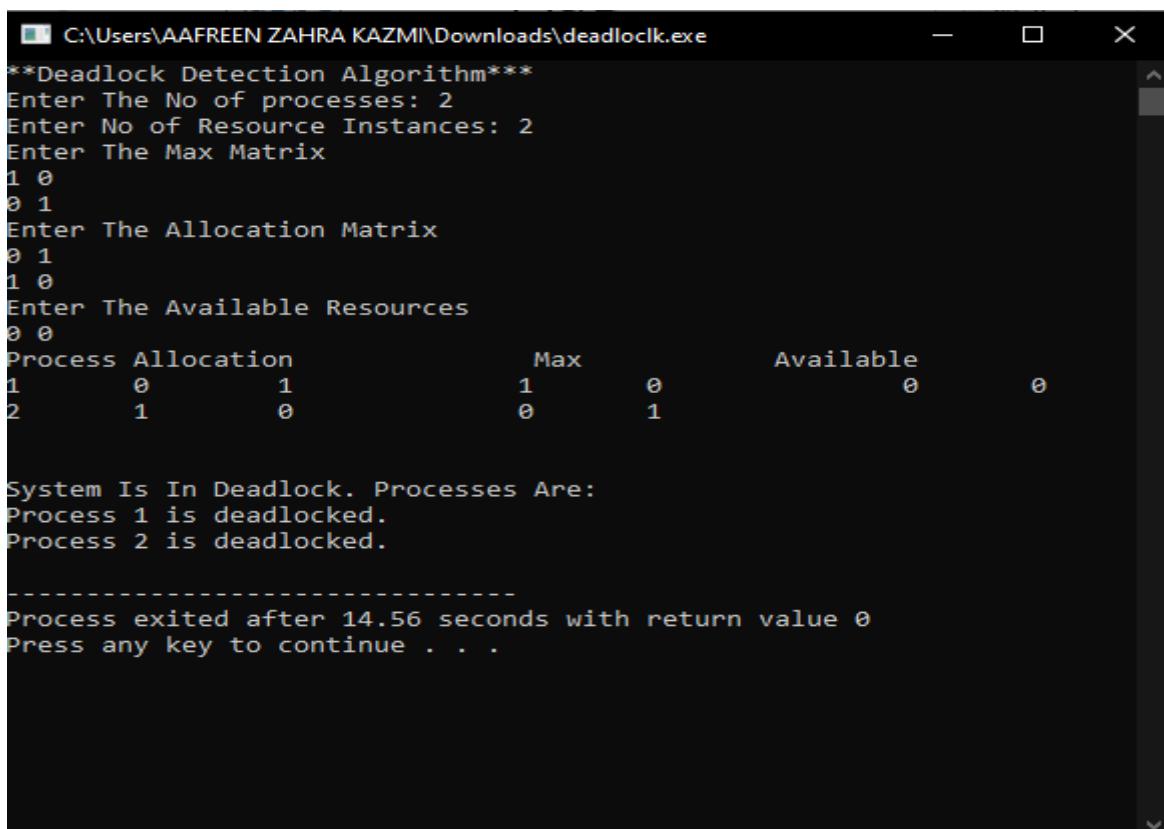


```
void cal() {  
    int finish[100], dead[100];  
    int flag = 1;  
    int j = 0;  
    // Initialize finish array to 0  
    for (int i = 0; i < n; i++) {
```

```
    finish[i] = 0;  
}  
  
// Calculate the Need matrix  
  
for (int i = 0; i < n; i++) {  
    for (int j = 0; j < r; j++) {  
        need[i][j] = max[i][j] - alloc[i][j];  
    }  
}  
  
// Deadlock detection algorithm  
  
while (flag) {  
    flag = 0;  
    for (int i = 0; i < n; i++) {  
        int c = 0;  
        for (int j = 0; j < r; j++) {  
            if (finish[i] == 0 && need[i][j] <= avail[j]) {  
                c++;  
            }  
        }  
        if (c == r) {  
            for (int k = 0; k < r; k++) {  
                avail[k] += alloc[i][k];  
            }  
            finish[i] = 1;  
            flag = 1;  
        }  
    }  
}  
  
// Check if there are any unfinished processes (deadlocked)  
  
for (int i = 0; i < n; i++) {  
    if (finish[i] == 0) {  
        dead[j] = i + 1; // Store the deadlocked process index  
        j++;  
    }  
}
```

```
// Output the result  
if (j > 0) {  
    printf("\n\nSystem Is In Deadlock. Processes Are:\n");  
    for (int i = 0; i < j; i++) {  
        printf("Process %d is deadlocked.\n", dead[i]);  
    }  
} else {  
  
printf("\nNo Deadlock Occurred.\n");  
}  
}
```

OUTPUT



```
**Deadlock Detection Algorithm***  
Enter The No of processes: 2  
Enter No of Resource Instances: 2  
Enter The Max Matrix  
1 0  
0 1  
Enter The Allocation Matrix  
0 1  
1 0  
Enter The Available Resources  
0 0  
Process Allocation           Max          Available  
1      0         1           1       0           0       0  
2      1         0           0       1  
  
System Is In Deadlock. Processes Are:  
Process 1 is deadlocked.  
Process 2 is deadlocked.  
-----  
Process exited after 14.56 seconds with return value 0  
Press any key to continue . . .
```

Figure 12.1

12.2. SHOW THE SYSTEM IS IN SAFE STATE AND PROCESS THE SEQUENCE OF THE PROCESS?

CODE

```
#include <stdio.h>

int max[100][100];
int alloc[100][100];
int need[100][100];
int avail[100];
int n, r;
void input();
void show();
void cal();
int main() {
    printf("##Deadlock Detection Algorithm###\n");
    input();
    show();
    cal();
    return 0;
}
void input() {
    printf("Enter The Number of Processes: ");
    scanf("%d", &n);
    printf("Enter Number of Resource Types: ");
    scanf("%d", &r);
    printf("Enter The Max Matrix:\n");
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < r; j++) {
            scanf("%d", &max[i][j]);
        }
    }
    printf("Enter The Allocation Matrix:\n");
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < r; j++) {
            scanf("%d", &alloc[i][j]);
        }
    }
}
```

```
        }
    }

printf("Enter The Available Resources:\n");
for (int j = 0; j < r; j++) {
    scanf("%d", &avail[j]);
}

void show() {
    printf("Process\tAllocation\t\t Max\t\tAvailable\n");
    for (int i = 0; i < n; i++) {
        printf("P%d\t", i + 1);
        for (int j = 0; j < r; j++) {
            printf("%d ", alloc[i][j]);
        }
        printf("\t\t");
        for (int j = 0; j < r; j++) {
            printf("%d ", max[i][j]);
        }
        printf("\t\t");
    }
    if (i == 0) {
        for (int j = 0; j < r; j++) {
            printf("%d ", avail[j]);
        }
    }
    printf("\n");
}
}

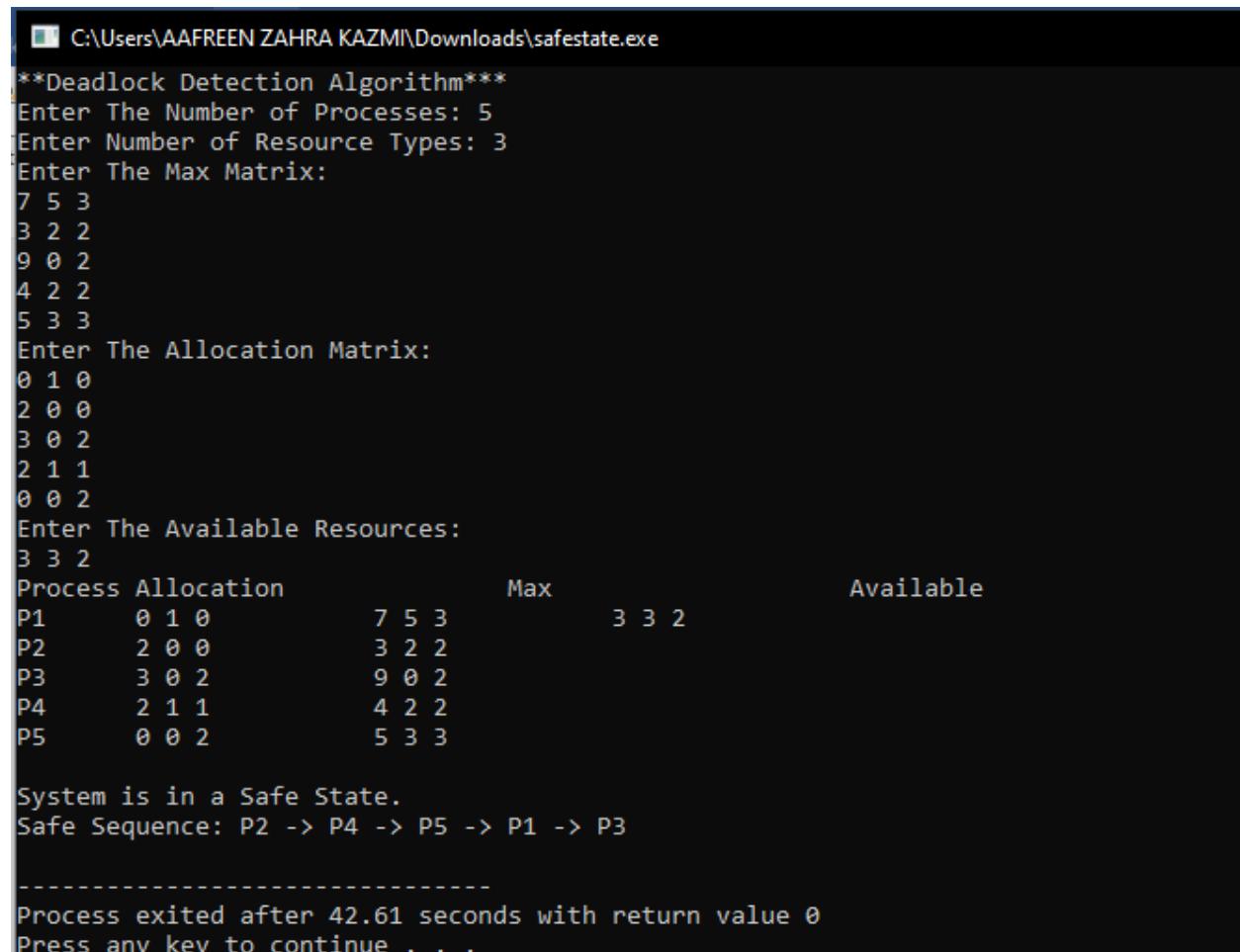
void cal() {
    int finish[100] = {0};
    int safeSeq[100];
    int work[100];
    int index = 0;
    // Copy available resources to work array
    for (int i = 0; i < r; i++) {
```

```
work[i] = avail[i];  
}  
// Calculate the Need matrix  
for (int i = 0; i < n; i++) {  
    for (int j = 0; j < r; j++) {  
        need[i][j] = max[i][j] - alloc[i][j];  
    }  
}  
// Safety algorithm  
int count = 0;  
while (count < n) {  
    int found = 0;  
    for (int i = 0; i < n; i++) {  
        if (!finish[i]) {  
            int j;  
            for (j = 0; j < r; j++) {  
                if (need[i][j] > work[j]) {  
                    break;  
                }  
            }  
            if (j == r) { // If all needs are satisfied  
                for (int k = 0; k < r; k++) {  
                    work[k] += alloc[i][k];  
                }  
                safeSeq[index++] = i;  
                finish[i] = 1;  
                found = 1;  
                count++;  
            }  
        }  
    }  
  
    if (!found) {  
        printf("\nSystem is in Deadlock.\n");  
    }  
}
```

```
        return;
    }
}

// If all processes are finished, output the safe sequence
printf("\nSystem is in a Safe State.\nSafe Sequence: ");
for (int i = 0; i < n; i++) {
    printf("P%d", safeSeq[i] + 1);
    if (i < n - 1) {
        printf(" -> ");
    }
}
printf("\n");
}
```

OUTPUT



```
C:\Users\AAFREEN ZAHRA KAZMI\Downloads\safestate.exe
**Deadlock Detection Algorithm**
Enter The Number of Processes: 5
Enter Number of Resource Types: 3
Enter The Max Matrix:
7 5 3
3 2 2
9 0 2
4 2 2
5 3 3
Enter The Allocation Matrix:
0 1 0
2 0 0
3 0 2
2 1 1
0 0 2
Enter The Available Resources:
3 3 2
Process Allocation          Max           Available
P1    0 1 0          7 5 3          3 3 2
P2    2 0 0          3 2 2
P3    3 0 2          9 0 2
P4    2 1 1          4 2 2
P5    0 0 2          5 3 3

System is in a Safe State.
Safe Sequence: P2 -> P4 -> P5 -> P1 -> P3

-----
Process exited after 42.61 seconds with return value 0
Press any key to continue . . .
```

Figure 12.2

WEEK # 9

13. THREADS

13.1. Write a C++ program to demonstrate basic single threading **CODE**

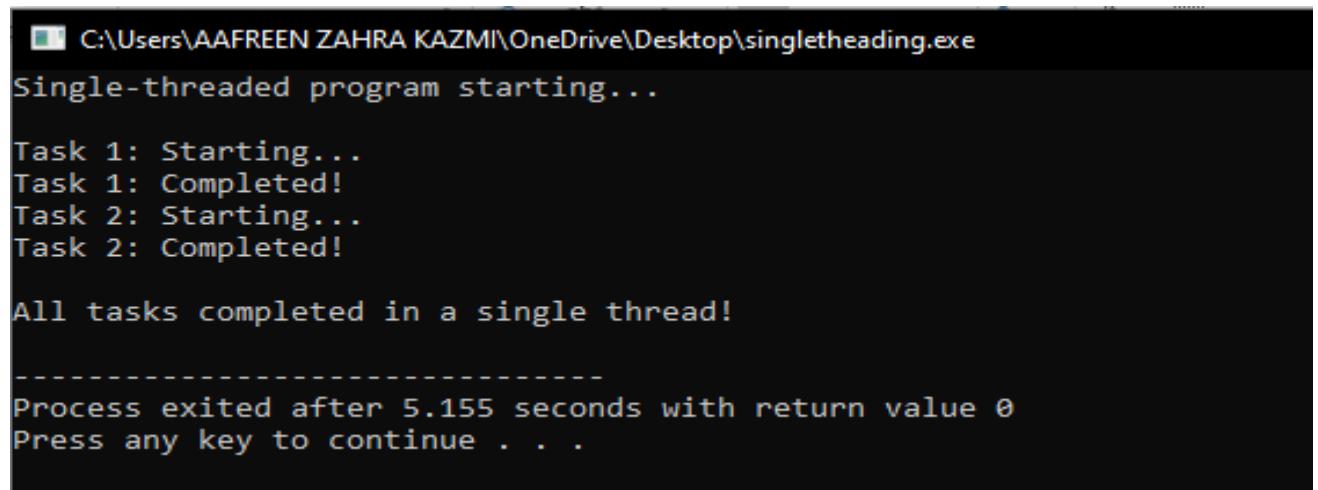
```
#include <iostream>
#include <chrono>
#include <thread> // Used for simulating task duration
using namespace std;

void task1() {
    cout << "Task 1: Starting..." << endl;
    this_thread::sleep_for(chrono::seconds(2)); // Simulate a 2-second task
    cout << "Task 1: Completed!" << endl;
}

void task2() {
    cout << "Task 2: Starting..." << endl;
    this_thread::sleep_for(chrono::seconds(3)); // Simulate a 3-second task
    cout << "Task 2: Completed!" << endl;
}

int main() {
    cout << "Single-threaded program starting...\n" << endl;
    task1(); // Execute Task 1
    task2(); // Execute Task 2
    cout << "\nAll tasks completed in a single thread!" << endl;
    return 0;
}
```

OUTPUT



```
C:\Users\AAFREEN ZAHRA KAZMI\OneDrive\Desktop\singlethreading.exe
Single-threaded program starting...

Task 1: Starting...
Task 1: Completed!
Task 2: Starting...
Task 2: Completed!

All tasks completed in a single thread!

-----
Process exited after 5.155 seconds with return value 0
Press any key to continue . . .
```

Figure 13.1.

13.2. Write a C++ program to demonstrate basic multi-threading.

CODE

```
#include <iostream>
#include <thread>
#include <chrono>

using namespace std;

void postUpdate(string username, string message) {
    cout << username << " is posting: " << message << endl;
    this_thread::sleep_for(chrono::seconds(2)); // Simulate time taken to post
    cout << username << " finished posting: " << message << endl;
}

int main() {
    cout << "Multi-threaded social media updates simulation starting...\n" << endl;
    // Create threads for different users posting updates
    thread user1(postUpdate, "Alice", "Hello, world!");
    thread user2(postUpdate, "Bob", "Loving the weather today!");
    thread user3(postUpdate, "Charlie", "Check out my latest photo!");

    // Wait for all threads to finish
    user1.join();
    user2.join();
    user3.join();

    cout << "\nAll users have finished posting their updates!" << endl;
}
```

```
    return 0;  
}
```

OUTPUT

```
C:\Users\AAFREEN ZAHRA KAZMI\OneDrive\Desktop\multithreadin.exe  
Multi-threaded social media updates simulation starting...  
  
Alice is posting: Hello, world!  
Charlie is posting: Check out my latest photo!  
Bob is posting: Loving the weather today!  
Alice finished posting: Hello, world!  
Charlie finished posting: Check out my latest photo!  
Bob finished posting: Loving the weather today!  
  
All users have finished posting their updates!  
-----  
Process exited after 2.148 seconds with return value 0  
Press any key to continue . . .
```

Figure 13.2

14. POSIX-THREAD

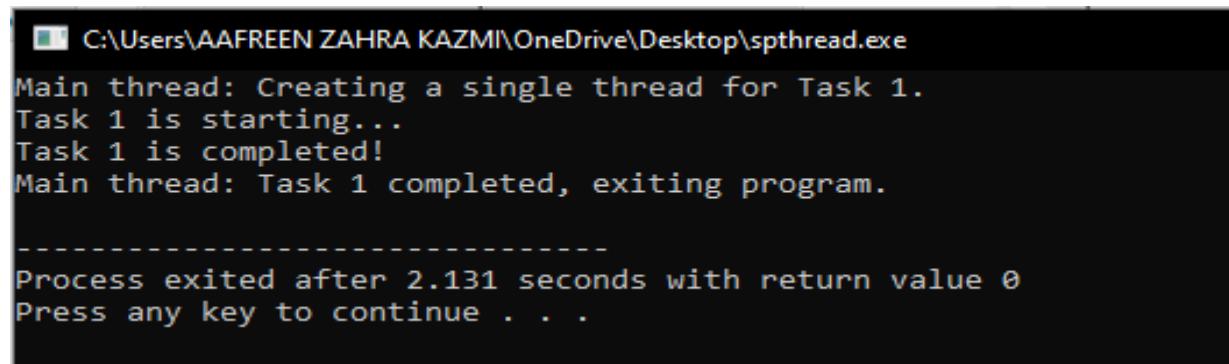
14.1. Write a C++ program to demonstrate basic single threading using P-Thread CODE

```
#include <iostream>
#include <pthread.h>
#include <unistd.h> // For sleep function
using namespace std;

// Function to be executed by the thread
void* task(void* arg) {
    const char* taskName = (const char*)arg;
    cout << taskName << " is starting..." << endl;
    sleep(2); // Simulate task delay
    cout << taskName << " is completed!" << endl;
    return nullptr;
}

int main() {
    pthread_t thread; // Declare a thread identifier
    const char* taskName = "Task 1";
    cout << "Main thread: Creating a single thread for Task 1.\n";
    // Create a thread to execute the task function
    if (pthread_create(&thread, nullptr, task, (void*)taskName) != 0) {
        cerr << "Error: Failed to create the thread." << endl;
        return 1;
    }
    // Wait for the created thread to complete
    pthread_join(thread, nullptr);
    cout << "Main thread: Task 1 completed, exiting program." << endl;
    return 0;
}
```

OUTPUT



```
C:\Users\AAFREEN ZAHRA KAZMI\OneDrive\Desktop\spthread.exe
Main thread: Creating a single thread for Task 1.
Task 1 is starting...
Task 1 is completed!
Main thread: Task 1 completed, exiting program.

-----
Process exited after 2.131 seconds with return value 0
Press any key to continue . . .
```

Figure 14.1

14.2. Write a C++ program to demonstrate basic multi-threading using P-Thread.

CODE

```
#include <iostream>
#include <pthread.h>
#include <unistd.h> // For sleep function
using namespace std;

// Function to simulate sending a message
void* sendMessage(void* arg) {
    const char* user = (const char*)arg;
    cout << user << " is sending a message..." << endl;
    sleep(2); // Simulate sending delay
    cout << user << " has sent the message!" << endl;
    return nullptr;
}

int main() {
    pthread_t threads[3]; // Array to hold thread identifiers
    const char* users[] = {"Alice", "Bob", "Charlie"};
    cout << "Simulating multi-threaded message sending...\n" << endl;
    // Create a thread for each user
    for (int i = 0; i < 3; i++) {
        if (pthread_create(&threads[i], nullptr, sendMessage, (void*)users[i]) != 0) {
            cerr << "Error: Failed to create thread for " << users[i] << endl;
            return 1;
        }
    }
}
```

```
}

// Wait for all threads to complete

for (int i = 0; i < 3; i++) {

    pthread_join(threads[i], nullptr);

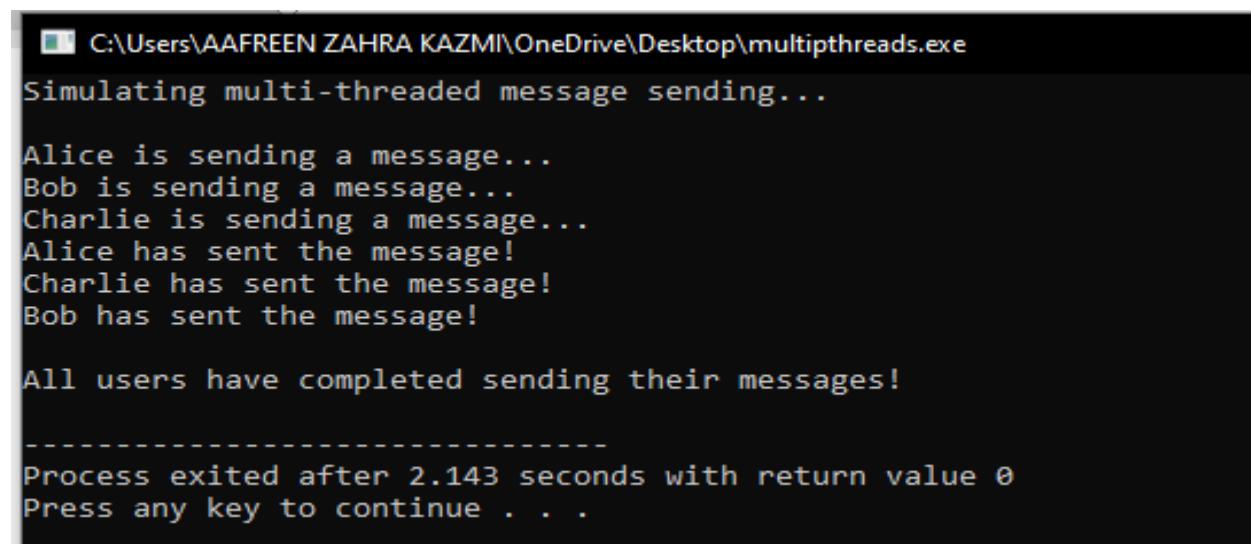
}

cout << "\nAll users have completed sending their messages!" << endl;

return 0;

}
```

OUTPUT



The screenshot shows a terminal window with the following text output:

```
C:\Users\AAFREEN ZAHRA KAZMI\OneDrive\Desktop\multipthreads.exe
Simulating multi-threaded message sending...

Alice is sending a message...
Bob is sending a message...
Charlie is sending a message...
Alice has sent the message!
Charlie has sent the message!
Bob has sent the message!

All users have completed sending their messages!
-----
Process exited after 2.143 seconds with return value 0
Press any key to continue . . .
```

Figure 14.2

WEEK # 10

15. PAGE REPLACEMENT ALGORITHM

15.1. FIFO (FIRST IN FIRST OUT)

Code

```
#include <iostream>
#include <queue> // For FIFO management
using namespace std;

void fifoPageReplacement(int pages[], int n, int capacity) {

    int frames[capacity]; // Array to hold the pages in memory
    int currentSize = 0; // Current size of the frames array
    int pageFaults = 0; // Count of page faults
    int index = 0; // Pointer for FIFO replacement

    // Initialize frames to -1 to indicate they are empty
    for (int i = 0; i < capacity; i++) {
        frames[i] = -1;
    }

    cout << "Page Reference | Frames Content | Page Fault" << endl;
    for (int i = 0; i < n; i++) {
        bool pageFound = false;
        // Check if the page is already in frames
        for (int j = 0; j < currentSize; j++) {
            if (frames[j] == pages[i]) {
                pageFound = true;
                break;
            }
        }
        // If the page is not found, replace using FIFO
        if (!pageFound) {
            frames[index] = pages[i]; // Replace the oldest page
            index = (index + 1) % capacity;
            pageFaults++;
        }
    }
}
```

```
index = (index + 1) % capacity; // Move FIFO pointer
pageFaults++;

// Increment currentSize until frames are full

if (currentSize < capacity) {

    currentSize++;

}

// Print page fault info

cout << " " << pages[i] << " | ";

for (int k = 0; k < currentSize; k++) {

    cout << frames[k] << " ";

}

for (int k = currentSize; k < capacity; k++) {

    cout << "- ";

}

cout << " | Yes" << endl;

} else {

    // Print no page fault info

    cout << " " << pages[i] << " | ";

    for (int k = 0; k < currentSize; k++) {

        cout << frames[k] << " ";

    }

    for (int k = currentSize; k < capacity; k++) {

        cout << "- ";

    }

    cout << " | No" << endl;

}

cout << "\nTotal Page Faults: " << pageFaults << endl;
```

```
}
```

```
int main() {
```

```
    int pages[] = {7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2}; // Page reference string
```

```
    int n = sizeof(pages) / sizeof(pages[0]); // Number of pages
```

```
    int capacity = 3; // Number of frames
```

```
    fifoPageReplacement(pages, n, capacity);
```

```
    return 0;
```

```
}
```

OUTPUT

Page Reference	Frames	Content	Page Fault
7	7 - -		Yes
0	7 0 -		Yes
1	7 0 1		Yes
2	2 0 1		Yes
0	2 0 1		No
3	2 3 1		Yes
0	2 3 0		Yes
4	4 3 0		Yes
2	4 2 0		Yes
3	4 2 3		Yes
0	0 2 3		Yes
3	0 2 3		No
2	0 2 3		No

```
Total Page Faults: 10
```

```
-----
```

```
Process exited after 0.2028 seconds with return value 0
```

```
Press any key to continue . . .
```

Figure 15.1

15.2. LRU (LEAST RECENTLY USED)

CODE

```
#include <iostream>

#include <unordered_map>

using namespace std;

int findLRU(int time[], int n) {

    int min_time = time[0], pos = 0;

    for (int i = 1; i < n; i++) {

        if (time[i] < min_time) {

            min_time = time[i];

            pos = i;

        }

    }

    return pos;

}

void lruPageReplacement(int pages[], int n, int capacity) {

    int frames[10], time[10];

    int page_faults = 0, count = 0;

    unordered_map<int, int> page_map; // To store page and its index

    for (int i = 0; i < capacity; i++) {

        frames[i] = -1; // Initialize frame slots to -1

    }

    cout << "Page Frames Status:\n";

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

        int found = 0;

        // Check if the page is already in the frames

        for (int j = 0; j < capacity; j++) {

            if (frames[j] == pages[i]) {
```

```
        found = 1;

        time[j] = count++; // Update access time

        break;

    }

}

if (!found) {

    // Find position to replace

    int replace_idx = (i < capacity) ? i : findLRU(time, capacity);

    page_map.erase(frames[replace_idx]); // Remove old page from map

    frames[replace_idx] = pages[i]; // Replace with the new page

    time[replace_idx] = count++; // Update access time

    page_map[pages[i]] = replace_idx; // Add new page to map

    page_faults++;

}

// Display current frame state

for (int j = 0; j < capacity; j++) {

    if (frames[j] == -1)

        cout << "- ";

    else

        cout << frames[j] << " ";

}

cout << endl;

}

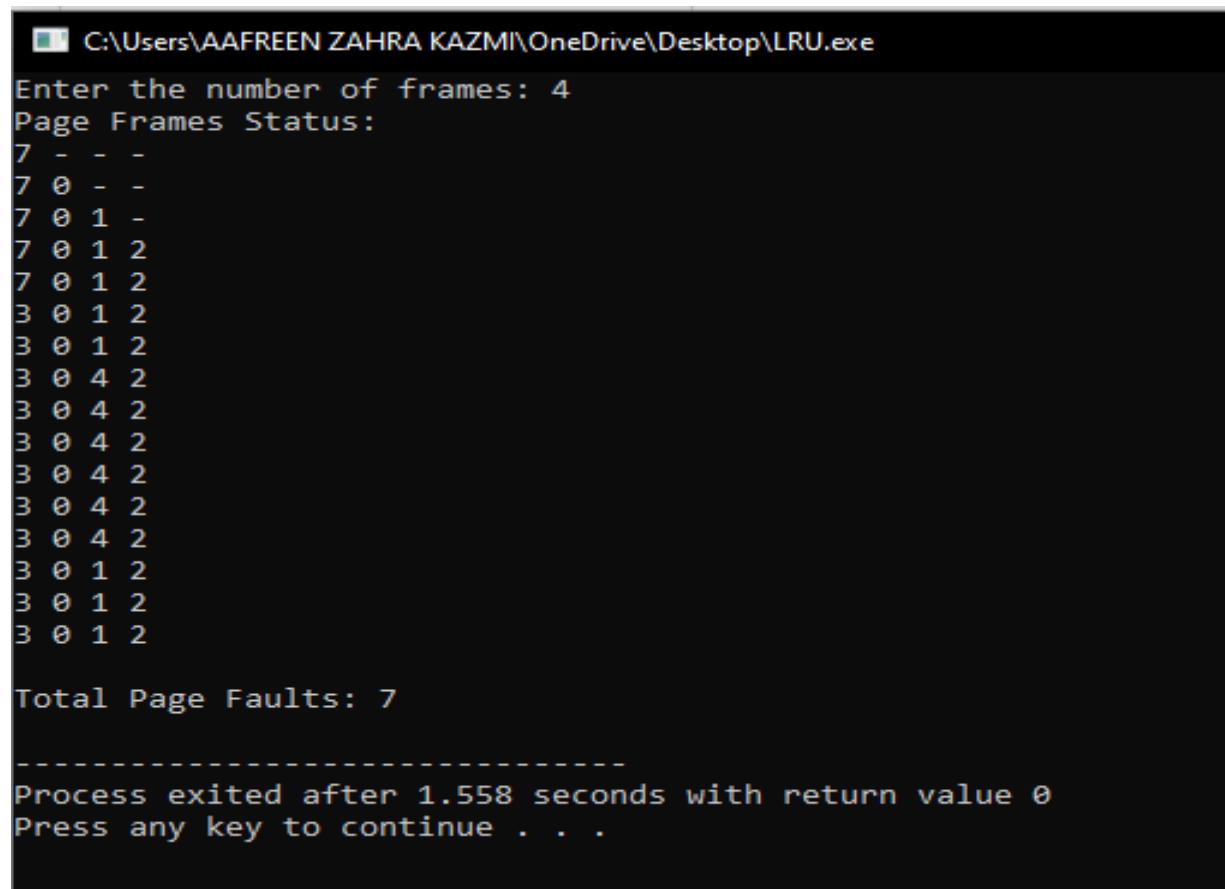
cout << "\nTotal Page Faults: " << page_faults << endl;

}

int main() {
```

```
int pages[] = {7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1, 2, 0};  
int n = sizeof(pages) / sizeof(pages[0]);  
int capacity;  
cout << "Enter the number of frames: ";  
cin >> capacity;  
lruPageReplacement(pages, n, capacity);  
return 0;  
}
```

OUTPUT



```
C:\Users\AAFREEN ZAHRA KAZMI\OneDrive\Desktop\LRU.exe  
Enter the number of frames: 4  
Page Frames Status:  
7 - - -  
7 0 - -  
7 0 1 -  
7 0 1 2  
7 0 1 2  
3 0 1 2  
3 0 1 2  
3 0 4 2  
3 0 4 2  
3 0 4 2  
3 0 4 2  
3 0 4 2  
3 0 1 2  
3 0 1 2  
3 0 1 2  
Total Page Faults: 7  
-----  
Process exited after 1.558 seconds with return value 0  
Press any key to continue . . .
```

Figure 15.2

15.3. OPTIMAL PAGE REPLACEMENT

CODE

```
#include <iostream>

using namespace std;

int predict(int pages[], int frames[], int n, int frame_count, int current_index) {

    int farthest = current_index, pos = -1;

    for (int i = 0; i < frame_count; i++) {

        int j;

        for (j = current_index; j < n; j++) {

            if (frames[i] == pages[j]) {

                if (j > farthest) {

                    farthest = j;

                    pos = i;

                }

                break;

            }

        }

        if (j == n) return i; // If not found in future

    }

    return (pos == -1) ? 0 : pos;

}
```

```
void optimalPageReplacement(int pages[], int n, int frame_count) {

    int frames[10], page_faults = 0;

    for (int i = 0; i < frame_count; i++) frames[i] = -1;

    cout << "Page Frames Status:\n";

    for (int i = 0; i < n; i++) {
```

```
int found = 0;

// Check if the page is already in the frames

for (int j = 0; j < frame_count; j++) {

    if (frames[j] == pages[i]) {

        found = 1;

        break;

    }

}

if (!found) {

    // Find the frame to replace

    int replace_idx = (i < frame_count) ? i : predict(pages, frames, n, frame_count, i);

    frames[replace_idx] = pages[i];

    page_faults++;

}

// Display the current state of the frames

for (int j = 0; j < frame_count; j++) {

    if (frames[j] == -1) cout << "- ";

    else cout << frames[j] << " ";

}

cout << endl;

}

cout << "\nTotal Page Faults: " << page_faults << endl;

}

int main() {

    int pages[] = {7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1, 2, 0};

    int n = sizeof(pages) / sizeof(pages[0]);

    int frame_count;
```

```
cout << "Enter the number of frames: ";  
cin >> frame_count;  
  
optimalPageReplacement(pages, n, frame_count);  
  
return 0;  
}
```

OUTPUT

```
C:\Users\AAFREEN ZAHRA KAZMI\OneDrive\Desktop\OPTIMAL.exe  
Enter the number of frames: 4  
Page Frames Status:  
7 - - -  
7 0 - -  
7 0 1 -  
7 0 1 2  
7 0 1 2  
3 0 1 2  
3 0 1 2  
3 0 4 2  
3 0 4 2  
3 0 4 2  
3 0 4 2  
3 0 4 2  
3 0 4 2  
1 0 4 2  
1 0 4 2  
1 0 4 2  
  
Total Page Faults: 7  
  
-----  
Process exited after 1.531 seconds with return value 0  
Press any key to continue . . .
```

Figure 15.3

WEEK # 11

16. FRAGMENTATION

16.1. FIRST-FIT

CODE

```
#include <iostream>

using namespace std;

void firstFit(int blockSizes[], int m, int processSizes[], int n) {

    int allocation[n]; // To store block assigned to each process

    for (int i = 0; i < n; i++) allocation[i] = -1; // Initialize all allocations to -1

    for (int i = 0; i < n; i++) { // Iterate over each process

        for (int j = 0; j < m; j++) { // Find the first suitable block

            if (blockSizes[j] >= processSizes[i]) {

                allocation[i] = j; // Assign block to process

                blockSizes[j] -= processSizes[i]; // Reduce available size of block

                break;
            }
        }
    }

    // Output results

    cout << "Process No.\tProcess Size\tBlock No.\n";

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

        cout << i + 1 << "\t" << processSizes[i] << "\t";

        if (allocation[i] != -1)

            cout << allocation[i] + 1 << endl;

        else

            cout << "Not Allocated" << endl;
    }
}
```

```
int main() {  
  
    int blockSizes[] = {100, 500, 200, 300, 600};  
  
    int processSizes[] = {212, 417, 112, 426};  
  
    int m = sizeof(blockSizes) / sizeof(blockSizes[0]);  
  
    int n = sizeof(processSizes) / sizeof(processSizes[0]);  
  
  
    firstFit(blockSizes, m, processSizes, n);  
  
    return 0;  
}
```

OUTPUT

Process No.	Process Size	Block No.
1	212	2
2	417	5
3	112	2
4	426	Not Allocated

Process exited after 0.2144 seconds with return value 0
Press any key to continue . . .

Figure 16.1

16.2. WORST-FIT

CODE

```
#include <iostream>  
  
using namespace std;  
  
void worstFit(int blockSizes[], int m, int processSizes[], int n) {  
  
    int allocation[n]; // To store block assigned to each process  
  
    for (int i = 0; i < n; i++) allocation[i] = -1; // Initialize all allocations to -1  
  
  
    for (int i = 0; i < n; i++) { // Iterate over each process  
  
        int worstIndex = -1;
```

```
// Find the worst-fit block

for (int j = 0; j < m; j++) {

    if (blockSizes[j] >= processSizes[i]) {

        if (worstIndex == -1 || blockSizes[j] > blockSizes[worstIndex]) {

            worstIndex = j;

        }

    }

}

// If a suitable block was found, allocate it

if (worstIndex != -1) {

    allocation[i] = worstIndex;

    blockSizes[worstIndex] -= processSizes[i];

}

}

// Output results

cout << "Process No.\tProcess Size\tBlock No.\n";

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

    cout << i + 1 << "\t" << processSizes[i] << "\t\t";

    if (allocation[i] != -1)

        cout << allocation[i] + 1 << endl;

    else

        cout << "Not Allocated" << endl;

}

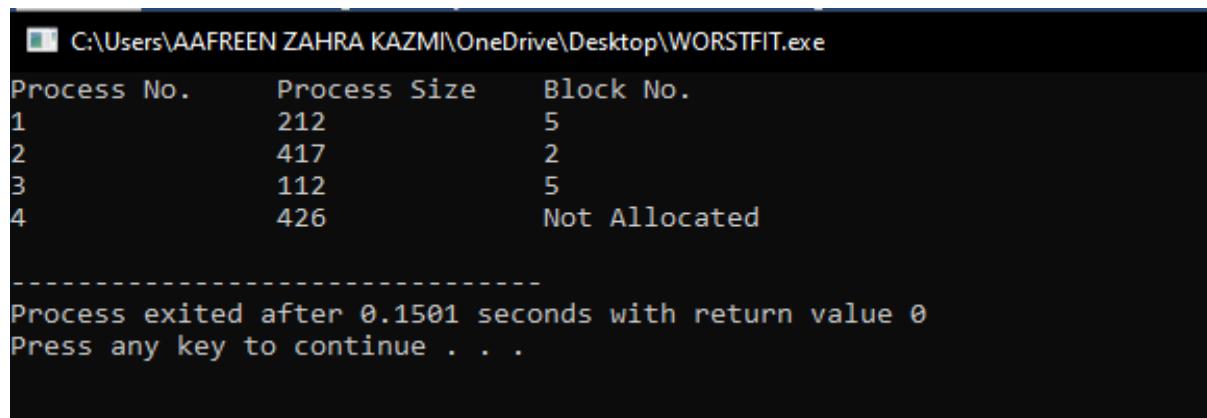
int main() {

    int blockSizes[] = {100, 500, 200, 300, 600};

    int processSizes[] = {212, 417, 112, 426};
```

```
int m = sizeof(blockSizes) / sizeof(blockSizes[0]);  
  
int n = sizeof(processSizes) / sizeof(processSizes[0]);  
  
worstFit(blockSizes, m, processSizes, n);  
  
return 0;  
}
```

OUTPUT



Process No.	Process Size	Block No.
1	212	5
2	417	2
3	112	5
4	426	Not Allocated

Process exited after 0.1501 seconds with return value 0
Press any key to continue . . .

Figure 16.2

16.3. BEST-FIT

CODE

```
#include <iostream>  
  
using namespace std;  
  
void bestFit(int blockSizes[], int m, int processSizes[], int n) {  
  
    int allocation[n]; // To store block assigned to each process  
  
    for (int i = 0; i < n; i++) allocation[i] = -1; // Initialize all allocations to -1  
  
    for (int i = 0; i < n; i++) { // Iterate over each process  
  
        int bestIndex = -1;  
  
        // Find the best-fit block  
  
        for (int j = 0; j < m; j++) {  
  
            if (blockSizes[j] >= processSizes[i]) {  
  
                if (bestIndex == -1 || blockSizes[j] < blockSizes[bestIndex]) {  
  
                    bestIndex = j;  
  
                }  
            }  
        }  
        allocation[i] = bestIndex;  
    }  
}
```

```
    }

}

// If a suitable block was found, allocate it

if (bestIndex != -1) {

    allocation[i] = bestIndex;

    blockSizes[bestIndex] -= processSizes[i];

}

}

// Output results

cout << "Process No.\tProcess Size\tBlock No.\n";

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

    cout << i + 1 << "\t" << processSizes[i] << "\t";

    if (allocation[i] != -1)

        cout << allocation[i] + 1 << endl;

    else

        cout << "Not Allocated" << endl;

}

int main() {

    int blockSizes[] = { 100, 500, 200, 300, 600};

    int processSizes[] = { 212, 417, 112, 426};

    int m = sizeof(blockSizes) / sizeof(blockSizes[0]);

    int n = sizeof(processSizes) / sizeof(processSizes[0]);

    bestFit(blockSizes, m, processSizes, n);

    return 0;
}
```

OUTPUT

```
C:\Users\AAFREEN ZAHRA KAZMI\OneDrive\Desktop\BEST-FIT.exe
Process No.      Process Size      Block No.
1                212              4
2                417              2
3                112              3
4                426              5
-----
Process exited after 0.2996 seconds with return value 0
Press any key to continue . . .
```

Figure 11.3

WEEK # 12

17. IPC(INTER PROCESS COMMUNICATION)

17.1. WRITER PROCESS

CODE

```
#include <iostream>
#include <sys/ipc.h>
#include <sys/msg.h>
#include <cstring>
#include <cstdio>

using namespace std;

// Structure for message queue

struct mesg_buffer {

    long mesg_type;

    char mesg_text[100];

};

int main() {

    key_t key;

    int msgid;

    // Generate unique key using ftok

    key = ftok("progfile", 65);

    // Create or access the message queue

    msgid = msgget(key, 0666 | IPC_CREAT);

    if (msgid == -1) {

        perror("msgget failed");

        return 1;

    }

    mesg_buffer message;

    message.mesg_type = 1;

    // Input data to send
```

```
cout << "Write Data: ";
cin.getline(message.mesg_text, sizeof(message.mesg_text));
// Send message using msgsnd
if (msgsnd(msgid, &message, sizeof(message), 0) == -1) {
    perror("msgsnd failed");
    return 1;
}
cout << "Data sent: " << message.mesg_text << endl;
return 0;
}
```

OUTPUT**Output**

```
Write Data: HLO I AM AAFREEN
Data sent: HLO I AM AAFREEN
```

```
==== Code Execution Successful ====
```

Figure 17.1

17.2. READER PROCESS

CODE

```
#include <iostream>
#include <sys/ipc.h>
#include <sys/msg.h>
#include <cstring>
#include <cstdio>

using namespace std;

// Structure for message queue

struct mesg_buffer {

    long mesg_type;

    char mesg_text[100];

};

int main() {

    key_t key;

    int msgid;

    // Generate unique key using ftok

    key = ftok("progfile", 65);

    // Get the message queue identifier

    msgid = msgget(key, 0666 | IPC_CREAT);

    if (msgid == -1) {

        perror("msgget failed");

        return 1;

    }

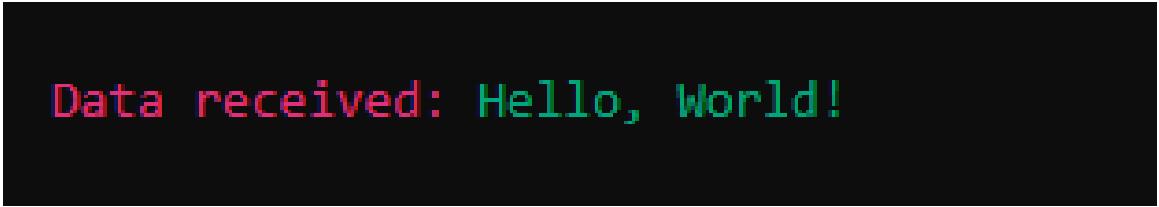
    mesg_buffer message;

    // Receive the message from the queue

    if (msgrcv(msgid, &message, sizeof(message), 1, 0) == -1) {

        perror("msgrcv failed");
```

```
    return 1;  
}  
  
cout << "Data received: " << message.msg_text << endl;  
  
// Remove the message queue  
  
if (msgctl(msqid, IPC_RMID, NULL) == -1) {  
  
    perror("msgctl failed to remove queue");  
  
    return 1;  
}  
  
return 0;  
}
```

OUTPUT

```
Data received: Hello, World!
```

WEEK # 13

18. DINING PHILOSOPHER PROBLEM

CODE

```
#include <iostream>
#include <thread>
#include <mutex>
#include <condition_variable>

#define N 5
#define THINKING 2
#define HUNGRY 1
#define EATING 0

#define LEFT (phnum + 4) % N
#define RIGHT (phnum + 1) % N

int state[N];
int phil[N] = { 0, 1, 2, 3, 4 };

std::mutex mutex;
std::condition_variable S[N];

void test(int phnum)
{
    if (state[phnum] == HUNGRY
        && state[LEFT] != EATING
        && state[RIGHT] != EATING) {
        // state that eating
        state[phnum] = EATING;

        std::this_thread::sleep_for(std::chrono::milliseconds(2000));
```

```
    std::cout << "Philosopher " << phnum + 1 << " takes fork " << LEFT + 1 << " and " << phnum + 1
    << std::endl;

    std::cout << "Philosopher " << phnum + 1 << " is Eating" << std::endl;

    // notify the philosopher that he can start eating
    S[phnum].notify_all();

}

}

// take up chopsticks

void take_fork(int phnum)
{
    std::unique_lock<std::mutex> lock(mutex);

    // state that hungry
    state[phnum] = HUNGRY;

    std::cout << "Philosopher " << phnum + 1 << " is Hungry" << std::endl;

    // eat if neighbours are not eating
    test(phnum);

    // if unable to eat wait to be signalled
    S[phnum].wait(lock);

    std::this_thread::sleep_for(std::chrono::milliseconds(1000));

}

// put down chopsticks

void put_fork(int phnum)
{
```

```
std::unique_lock<std::mutex> lock(mutex);

// state that thinking
state[phnum] = THINKING;

std::cout << "Philosopher " << phnum + 1 << " putting fork " << LEFT + 1 << " and " << phnum + 1
<< " down" << std::endl;
std::cout << "Philosopher " << phnum + 1 << " is thinking" << std::endl;
test(LEFT);
test(RIGHT);
}

void philosopher(int num)
{
    while (true) {
        take_fork(num);
        put_fork(num);
    }
}

int main()
{
    std::thread threads[N];

    for (int i = 0; i < N; i++) {
        threads[i] = std::thread(phiolosopher, i);
        std::cout << "Philosopher " << i + 1 << " is thinking" << std::endl;
    }

    for (int i = 0; i < N; i++)
        threads[i].join();

    return 0;
}
```

OUTPUT

```
C:\Users\AAFREEN ZAHRA KAZMI\OneDrive\Desktop\dinin.exe
Philosopher Philosopher 1 is Hungry
1 is thinking
Philosopher 2 is thinking
Philosopher 2 is Hungry
Philosopher 3 is thinking
Philosopher 3 is Hungry
Philosopher Philosopher 4 is Hungry
4 is thinking
Philosopher 5 is thinking
Philosopher 5 is Hungry
Philosopher 5 takes fork 4 and 5
Philosopher 5 is Eating
```

Figure 18

WEEK # 14

19. ZOMBIE PROCESS

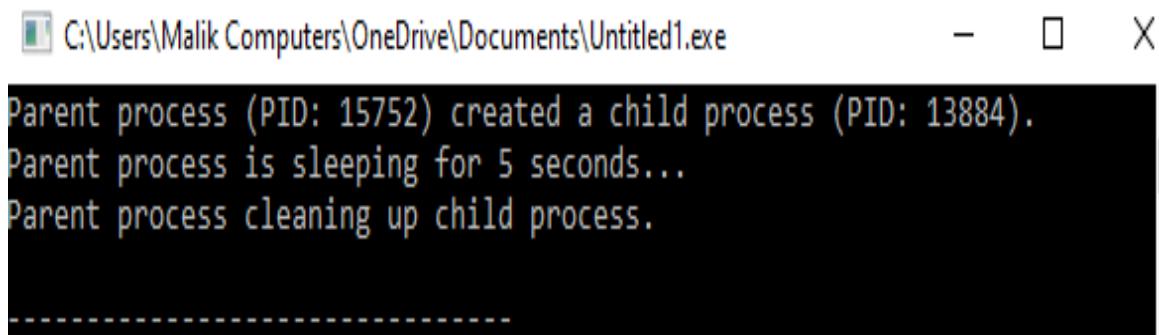
CODE

```
#include <iostream>
#include <windows.h> // For Windows API functions
using namespace std;

int main() {
    // Create a child process
    PROCESS_INFORMATION processInfo;
    STARTUPINFO startupInfo;
    ZeroMemory(&startupInfo, sizeof(startupInfo));
    startupInfo.cb = sizeof(startupInfo);
    ZeroMemory(&processInfo, sizeof(processInfo));
    // Simulate creating a child process
    if (CreateProcess(NULL, (LPSTR)"notepad.exe", NULL, NULL, FALSE, 0, NULL,
        NULL, &startupInfo, &processInfo)) {
        cout << "Parent process (PID: " << GetCurrentProcessId() << ") created a child process
(PID: " << processInfo.dwProcessId << ")." << endl;

        // Simulate not cleaning up the child immediately (zombie-like state)
        cout << "Parent process is sleeping for 5 seconds..." << endl;
        Sleep(5000);
        // Now parent cleans up the child process
        cout << "Parent process cleaning up child process." << endl;
        CloseHandle(processInfo.hProcess);
        CloseHandle(processInfo.hThread);
    } else {
        cerr << "Failed to create child process." << endl;
    }
    return 0;
}
```

OUTPUT



A screenshot of a terminal window titled 'C:\Users\Malik Computers\OneDrive\Documents\Untitled1.exe'. The window contains the following text:

```
Parent process (PID: 15752) created a child process (PID: 13884).
Parent process is sleeping for 5 seconds...
Parent process cleaning up child process.
```

Figure 19

20. ORPHAN PROCESS

CODE

```
#include <iostream>
#include <windows.h> // For Windows API functions
using namespace std;

int main() {
    // Create a child process
    PROCESS_INFORMATION processInfo;
    STARTUPINFO startupInfo;
    ZeroMemory(&startupInfo, sizeof(startupInfo));
    startupInfo.cb = sizeof(startupInfo);
    ZeroMemory(&processInfo, sizeof(processInfo));

    // Simulate creating a child process
    if (CreateProcess(NULL, (LPSTR)"notepad.exe", NULL, NULL, FALSE, 0, NULL,
        NULL, &startupInfo, &processInfo)) {
        cout << "Parent process (PID: " << GetCurrentProcessId() << ") created a child process
(PID: " << processInfo.dwProcessId << ")." << endl;

        // Simulate parent exiting immediately (orphan-like behavior)
        cout << "Parent process exiting now, leaving the child process running." << endl;
        ExitProcess(0); // Parent exits
    } else {
        cerr << "Failed to create child process." << endl;
    }
    return 0;
}
```

OUTPUT



C:\Users\Malik Computers\OneDrive\Documents\Untitled1.exe

```
Parent process (PID: 13972) created a child process (PID: 5648).  
Parent process exiting now, leaving the child process running.
```

Figure 20
