Page No.
Date: / /

Practical No:- 1

Aim :- Execute the disk operating system (DOS) system command

Theory :-

What is DOS ?

DOS stands for Disk Operating System. It is one of the most commonly used OS in single user enviornment. It runs mainly on personal computers made by different companies such as IBM, HP.

DOS can refer to a computer operating system that is loaded from a disk drive or to an operating system based on microsoft disk operating system there are two O.S reffered as DOS :-

1) MS - DOS (Microsoft disk operating System)

2) PC - DOS (Personal Computer Disk operating System)

DOS Command :-

① Date Command :-
This Command is used to Change or / and display current system date

② TIME Command :-
This Command is used to change or / and display current system time.

③ VER command :-
This Command is used to ~~Change or / and~~ display the disk volume level or serial number.

④ TITLE :-
This Command is used to change the ~~display~~ title of the MS-DOS. Screen.

⑤ COPY CON :-
Copy Con Command is used to create a text file.

⑥ TYPE Command :-

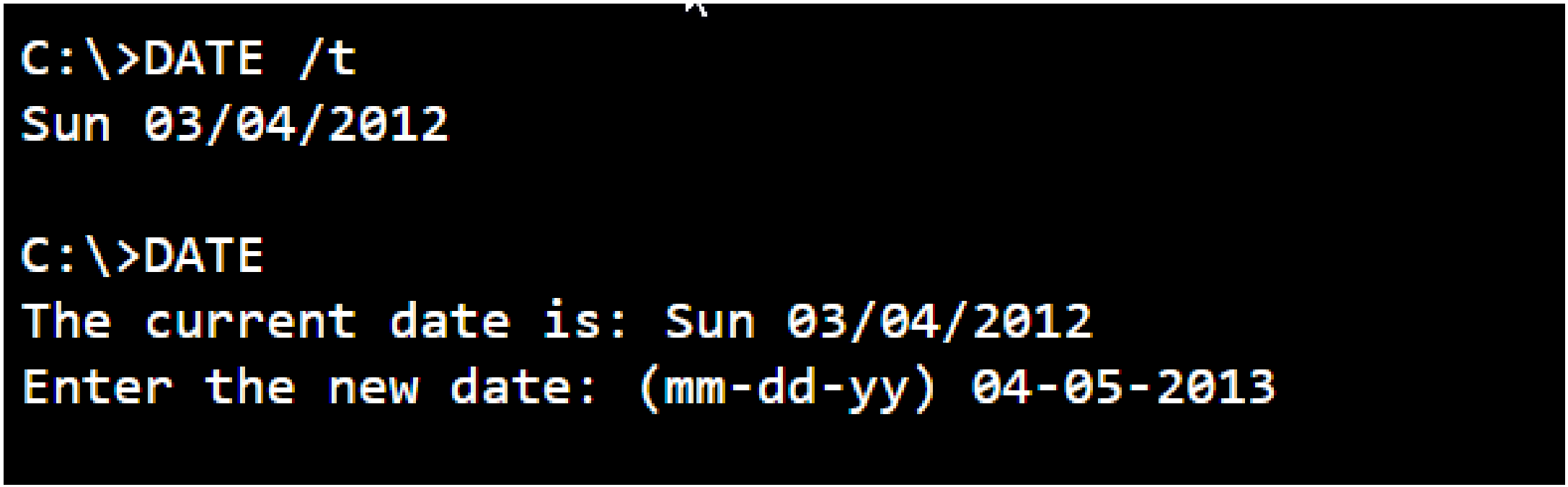
This command is used to display file.

⑦ DEL / ERASE Command :-

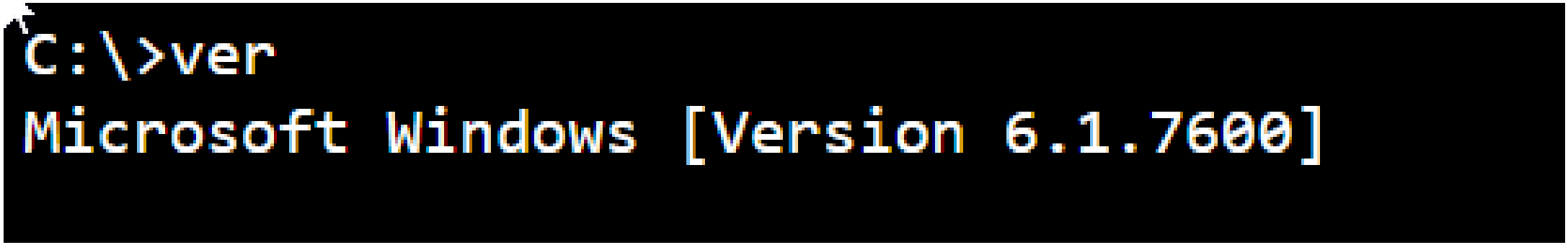
This command is used delete one or more than one file at a time.

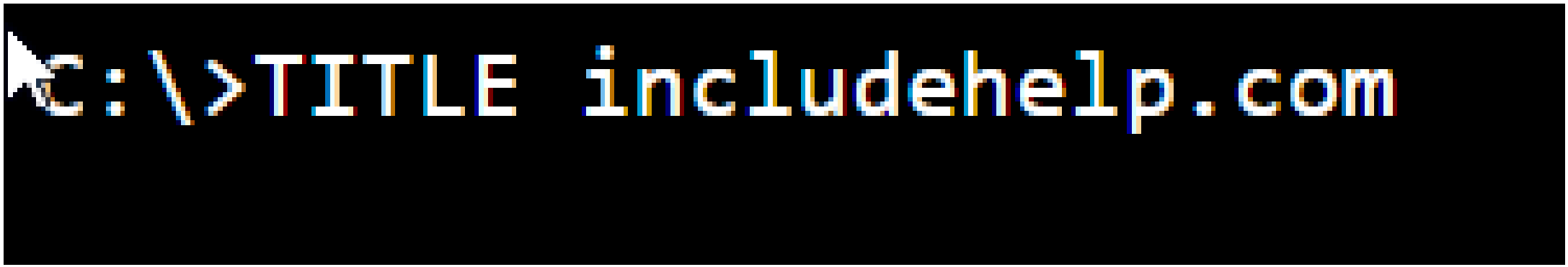
1] /P : [display the confirmation]
2] /F : [force deleting of read only file]

Conclusion :- Hence, we successfully executed dos command.


C:\>DATE /t
Sun 03/04/2012
C:\>DATE
The current date is: Sun 03/04/2012
Enter the new date: (mm-dd-yy) 04-05-2013

```
C:\>TIME /t
02:48 AM

C:\>TIME
The current time is:  2:48:47.32
Enter the new time: 06:10:00
```


C:\>ver
Microsoft Windows [Version 6.1.7600]


C:\>TITLE includehelp.com

```
C:\>COPY CON about_us.txt
includehelp.com provides online tutorials .
^Z
C:\>
```

```
C:\>TYPE about_us.txt
includehelp.com provides online tutorials .
C:\>
```

```
C:\> Del MyFile.Txt
C:\> Del/AH *.jpg
C:\> Del D:\Ram\Shyam\F1.Doc
```

Page No.
Date : / /

Practical :- 2

Aim :- Install and configure windows 9x, windows NT, Windows 2000 & windows XP Operating System.

Theory :-

Install and configure Windows 9x

Minimum Requirement :-

The minimum hardware requirement include

1. 486 60 x 66 Mhz or faster processor (Pentium requirement)
2. 16 megabytes (MB) of memory (24 MB)
3. 195 mb of free hard disk space (the requirement space may vary from 120 Mb to 295 mb depending on your computer configuration and the option you chose to install.)
4. CD - ROM
5. 3.5 Pnch high destiny floppy disk.

Page No.
Date : / /

Installation of windows 9x | 98

After your partitions and format your hard disk your can install windows 98.

① Insert the windows 98 startup disk in the floppy disk drive and then restart your computer

② When the 98 windows 98 Startup menu is displayed choose the start computer with CD-ROM support option and then. Press ENTER.

③ If CD-ROM Support is provided by the generic driver on the startup disk. You receive one of the following messages, where x is the drive letter that is assigned to your CD-ROM drive.

Drive x : = Driver MSCD001

Drive x : = Driver OEMSCD001

④ Insert the windows 98 CD-ROM in the CD-ROM. Type the following comand at CMD and press ENTER X:\setup
[where x is the drive letter that is assigned to your CD-ROM drive]

Date : / /

⑤ When you receive the following message Press ENTER and then follow the instruction on the screen on the computer to Complete the setup procedure :

Please wait while the setup initialize
Setup is now going to perform a routine check on your system to continue Press ENTER.

Windows NT

① Insert your windows NT CD and wait for the boot.

② Once this boot is done

The following Hardware configuration will appear and windows NT will detect your hardware configuration

Soon you will be promoted to insert NT setup disk 2, does this and hit enter.

Welcome to setup >> Enter

you will promoted to search for mass Storage device (CD-ROM in most cases) >> Enter

Page No.
Date: / /

Insert disk 3 of your winNT workstation setup.

Windows has found a CD-ROM Press ENTER to Continue. If you have SCSI device you will have to hit 's' to specify.

Now you have to agree the windows NT Licence.

Components >> standard setting >> Enter

Win NT has now detected your hard drive, if it has been setup before will have to be deleted, if you are looking to install a fresh Operating System Press "D" To delete the partition that is highlighted.

partition menu appears press C To Create a partition.

you will now be asked size of hard drive default is max space on your hard disk.

As we going to use FAT16 you will setup drive to any size below 2048 mb!

You will be asked where you wish NT to be installed hit enter.

Enter to continue.

Setup will then copy files.

After removing your floppy disk hit enter to reboot your system.

Configuration:-

- Setup is now completed and now you remove the floppy disk or CD from the drive. Hit Enter.

- NT will copy more files.

Windows NT setup →> Next

setup >> normal user >> laptop users
⇓ ⇓
Typical portable

fill your name >> organization name >> next

Enter the certificate of Authenticity >> enter

Page No.
Date: / /

Now your OEM Number .

Enter the name of the computer to be known

Admin password >> enter >> confirm >> next

Windows NT will now prompt you to make emergency . Repair disk.

Leave the NT Component at default >> next

NT network >> next

"NT network" menu will appear >> do not Connect >> next >> finish

Select your time zone >> apply >> close.

Now your display adapter, dont install until get into windows >> OK.

NT will copy files >> restart Computer to continue remove any CD and floppy disk first .

After rebooting you will arrive at login screen Ctrl + Alt + del to enter the pass word section.

Page No.
Date :

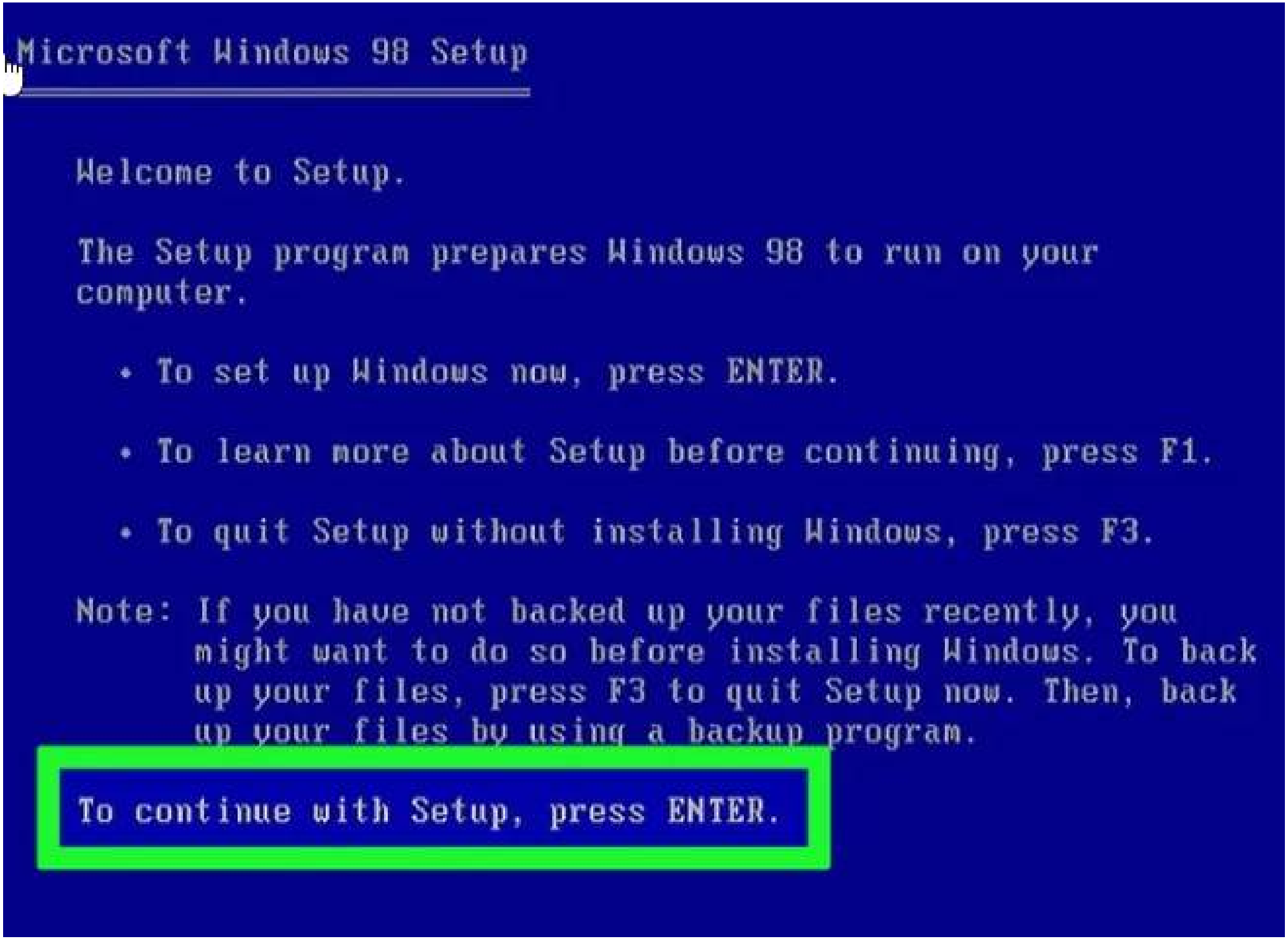
Enter your password

At last the desktop.... NT installed.

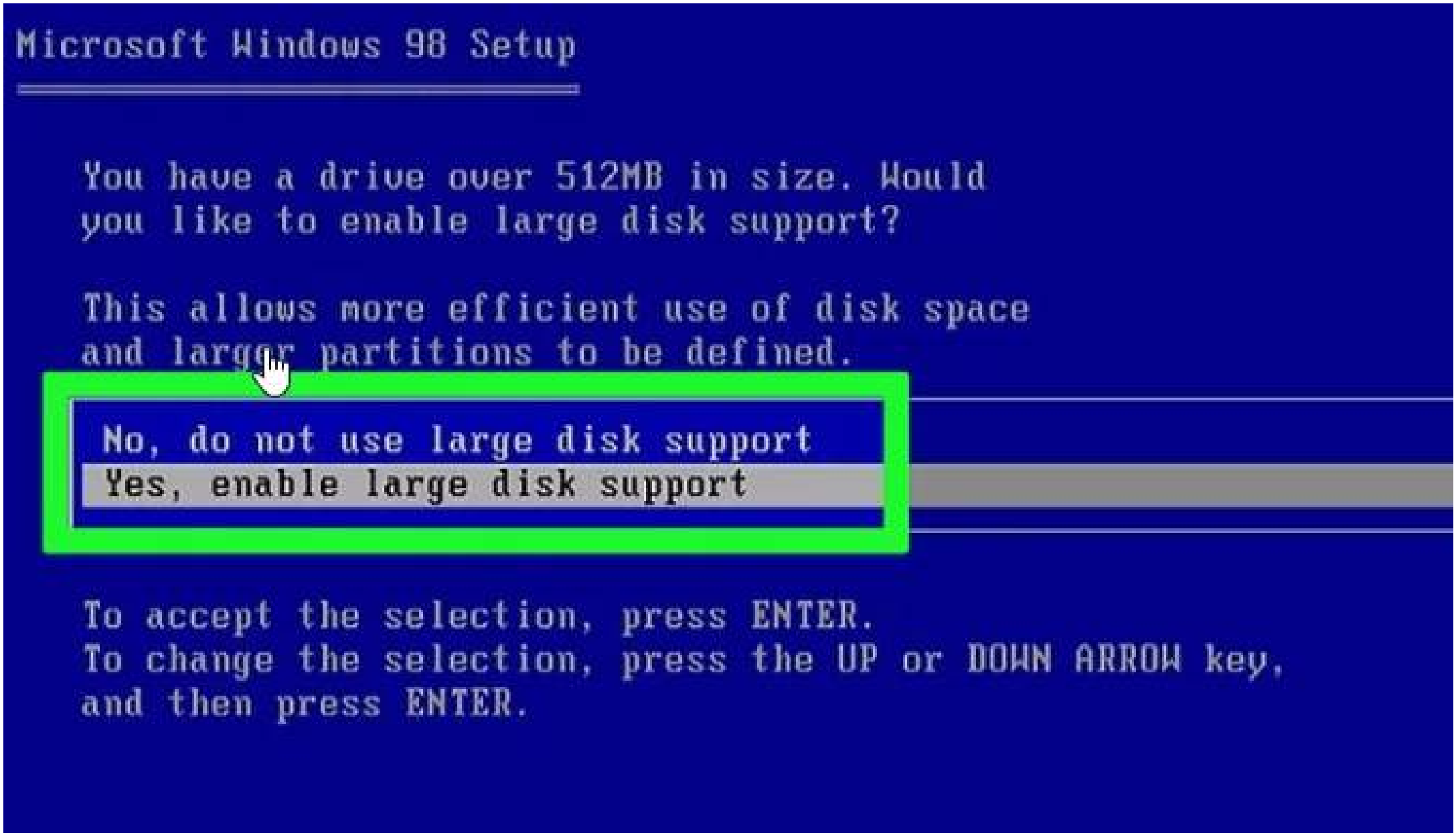
Conclusion:- Hence, we successfully installed Windows 98, 9x, xp and NT.


Microsoft Windows 98 CD-ROM Startup Menu
1. Boot from Hard Disk
2. Boot from CD-ROM
Enter your choice: _

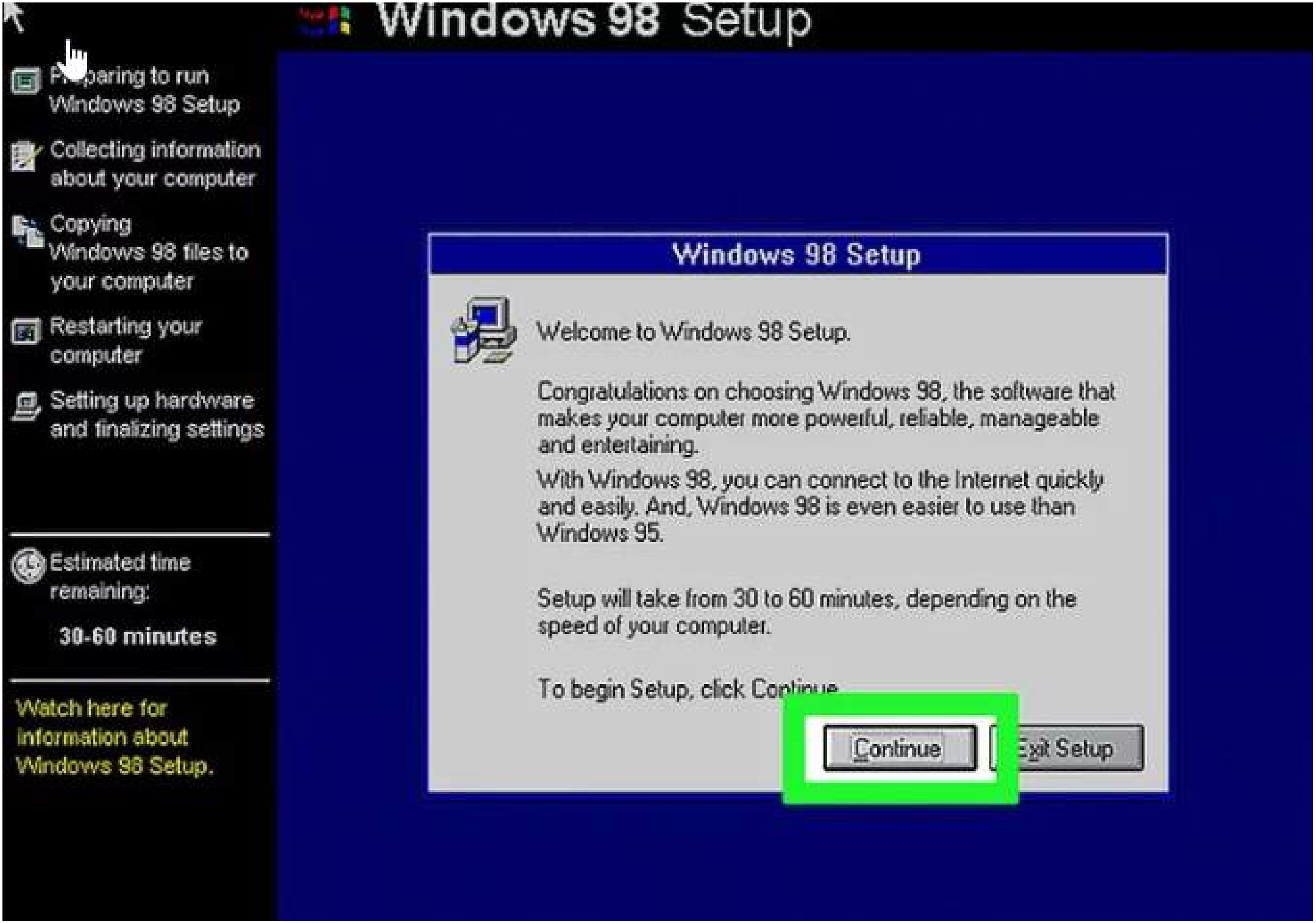

Microsoft Windows 98 Startup Menu
1. Start Windows 98 Setup from CD-ROM.
2. Start computer with CD-ROM support.
3. Start computer without CD-ROM support.
Enter a choice: 1
Time remaining: 26


Microsoft Windows 98 Setup
Welcome to Setup.
The Setup program prepares Windows 98 to run on your computer.
• To set up Windows now, press ENTER.
• To learn more about Setup before continuing, press F1.
• To quit Setup without installing Windows, press F3.
Note: If you have not backed up your files recently, you might want to do so before installing Windows. To back up your files, press F3 to quit Setup now. Then, back up your files by using a backup program.
To continue with Setup, press ENTER.


Microsoft Windows 98 Setup
Setup needs to configure the unallocated space on your hard disk to prepare it for use with Windows. None of your existing files will be affected.
To have Setup configure the space on your hard disk for you, choose the recommended option.
Configure unallocated disk space (recommended).
Exit Setup.
To accept the selection, press ENTER.
To change the selection, press the UP or DOWN ARROW key, and then press ENTER.


Microsoft Windows 98 Setup
You have a drive over 512MB in size. Would you like to enable large disk support?
This allows more efficient use of disk space and larger partitions to be defined.
No, do not use large disk support
Yes, enable large disk support
To accept the selection, press ENTER.
To change the selection, press the UP or DOWN ARROW key, and then press ENTER.


Microsoft Windows 98 Setup
Setup will restart your computer now.
Please make sure Windows 98 Boot Disk is in drive A.
• To continue, press ENTER.


Windows 98 Setup
Preparing to run Windows 98 Setup
Collecting information about your computer
Copying Windows 98 files to your computer
Restarting your computer
Setting up hardware and finalizing settings
Estimated time remaining:
30-60 minutes
Watch here for information about Windows 98 Setup.
Windows 98 Setup
Welcome to Windows 98 Setup.
Congratulations on choosing Windows 98, the software that makes your computer more powerful, reliable, manageable and entertaining.
With Windows 98, you can connect to the Internet quickly and easily. And, Windows 98 is even easier to use than Windows 95.
Setup will take from 30 to 60 minutes, depending on the speed of your computer.
To begin Setup, click Continue.
Continue
Exit Setup

Windows 98 Setup

Preparing to run Windows 98 Setup

Collecting information about your computer

Copying Windows 98 files to your computer

Restarting your computer

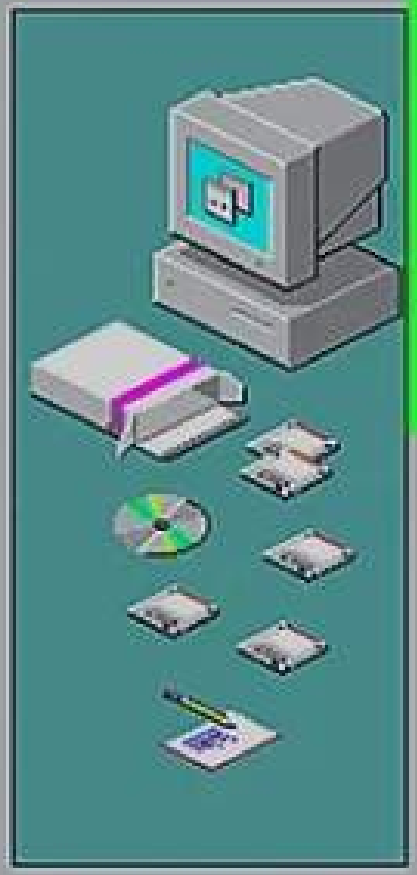
Setting up hardware and finalizing settings

Estimated time remaining:

35 minutes

You can install Windows 98 in a folder other than the default (C:\Windows).

Windows 98 Setup Wizard



Select Directory

Select the directory where you want to install Windows 98.

- (•) C:\WINDOWS
- ( ) Other directory

< Back | Next > | Cancel

Windows 98 Setup

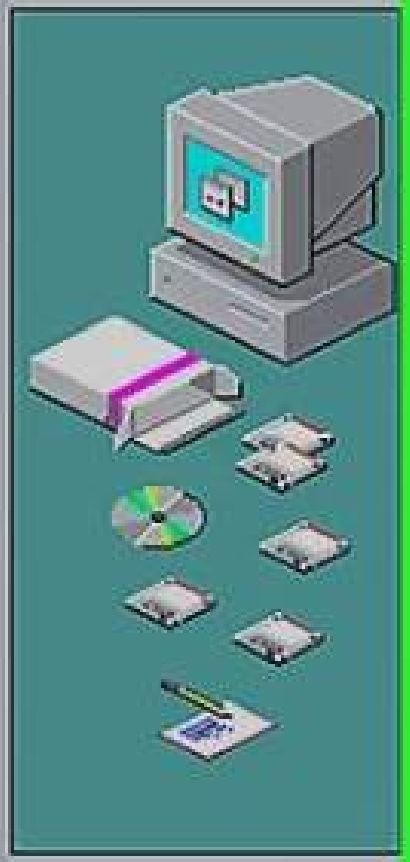
- Preparing to run Windows 98 Setup
- **Collecting information about your computer**
- Copying Windows 98 files to your computer
- Restarting your computer
- Setting up hardware and finalizing settings

Estimated time remaining:

31 minutes

Select a Setup option that best reflects your computer's environment or your user level.

Windows 98 Setup Wizard



Setup Options

Click the kind of Setup you prefer, and then click Next.

◉	Typical	This option is recommended for most computers.
○	Portable	Windows 98 will be set up with options that are useful for portable computers.
○	Compact	To save disk space, none of the optional components will be installed.
○	Custom	This option is for advanced users and system administrators only. You can customize all available Setup options.

< Back | Next > | Cancel

Windows 98 Setup

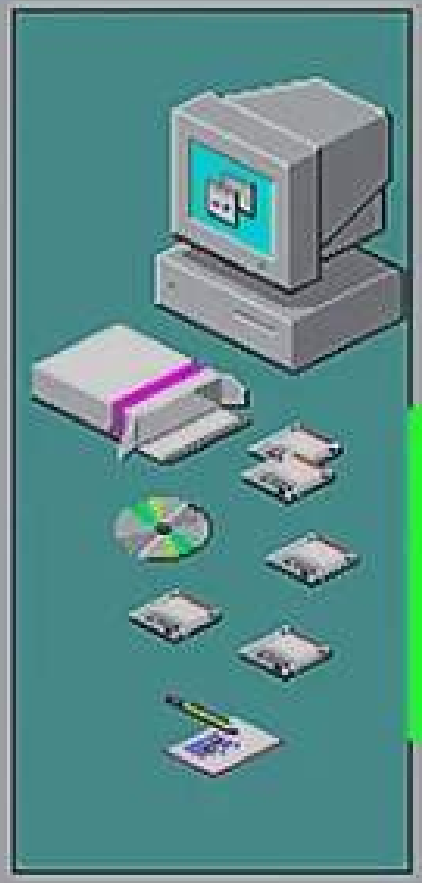
- Preparing to run Windows 98 Setup
- **Collecting information about your computer**
- Copying Windows 98 files to your computer
- Restarting your computer
- Setting up hardware and finalizing settings

Estimated time remaining:

29 minutes

Windows 98 comes with a variety of optional components. Setup can install these now, or you can install them later by clicking the Add/Remove Programs icon in Control Panel.

Windows 98 Setup Wizard



Windows Components

Windows 98 Setup has selected appropriate components to install, based on the kind of Setup you selected.

Some components require extra disk space and will not be installed unless you select them from the list.

I want Setup to:

- (•) Install the most common components (Recommended).
- ( ) Show me the list of components so I can choose.

< Back | Next > | Cancel

Windows 98 Setup

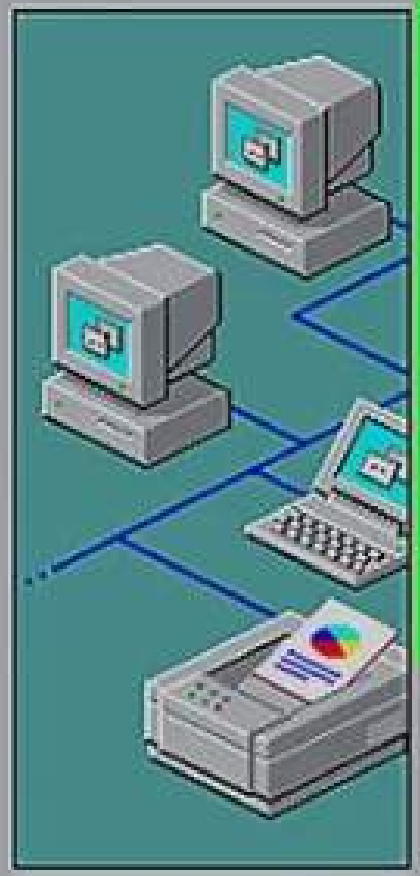
- Preparing to run Windows 98 Setup
- **Collecting information about your computer**
- Copying Windows 98 files to your computer
- Restarting your computer
- Setting up hardware and finalizing settings

Estimated time remaining:

29 minutes

Windows 98 comes with a variety of optional components. Setup can install these now, or you can install them later by clicking the Add/Remove Programs icon in Control Panel.

Identification



Windows uses the following information to identify your computer on the network. Please type a name for this computer, the workgroup it will appear in, and a short description of the computer.

Computer name: wikihow

Workgroup: WORKGROUP

Computer Description:

< Back | Next > | Cancel

Windows 98 Setup

- Preparing to run Windows 98 Setup
- **Collecting information about your computer**
- Copying Windows 98 files to your computer
- Restarting your computer
- Setting up hardware and finalizing settings

Estimated time remaining:

26 minutes

Select the country or region where you will use this computer.

Windows 98 Setup Wizard

Establishing Your Location

Windows 98 makes it easy to get region-specific news and information.

Selecting the correct location establishes the basic settings for receiving this information.

Select your country or region from the list below.

- Thailand
- Togo
- Trinidad
- Tunisia
- Turkey
- Turkmenistan
- Uganda
- Ukraine
- United Arab Emirates
- United Kingdom
- **United States**

< Back | Next > | Cancel

Windows 98 Setup

Preparing to run Windows 98 Setup

Collecting information about your computer

Copying Windows 98 files to your computer

Restarting your computer

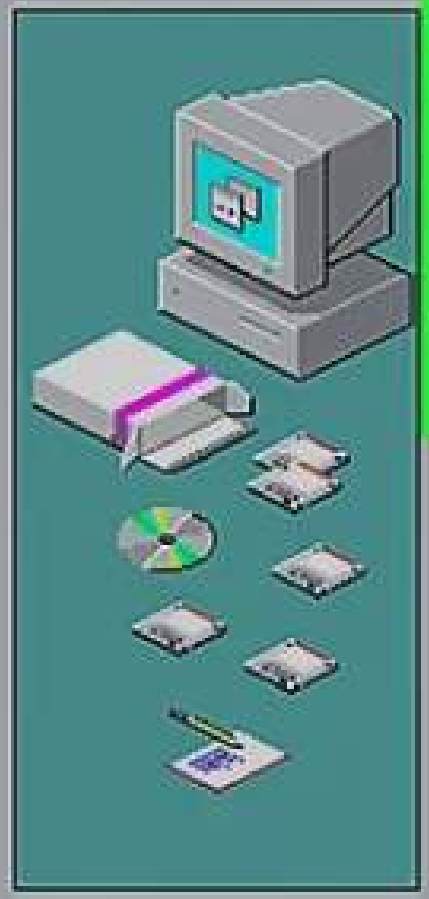
Setting up hardware and finalizing settings

Estimated time remaining:

23 minutes

The rest of Windows 98 Setup does not normally require your assistance.

Windows 98 Setup Wizard



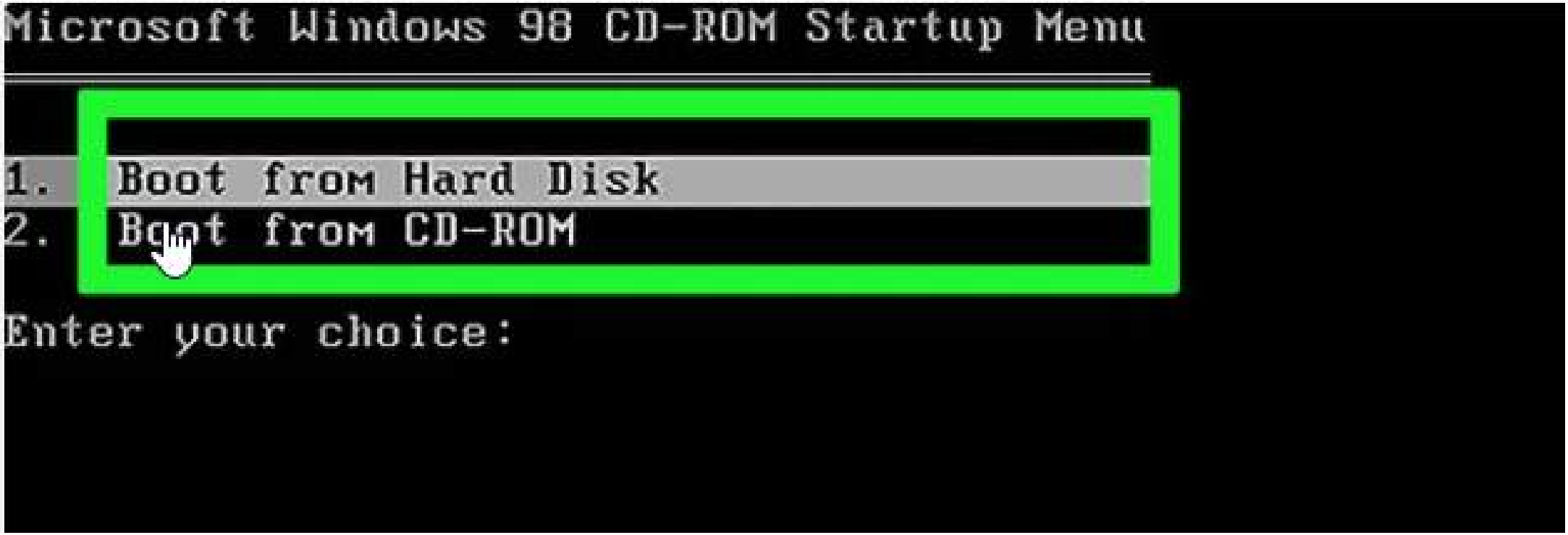
Start Copying Files

The Windows 98 Setup wizard now has enough information to start copying Windows 98 files to your computer. If you want to review or change any settings, click Back.

To start copying Windows 98 files, click Next.

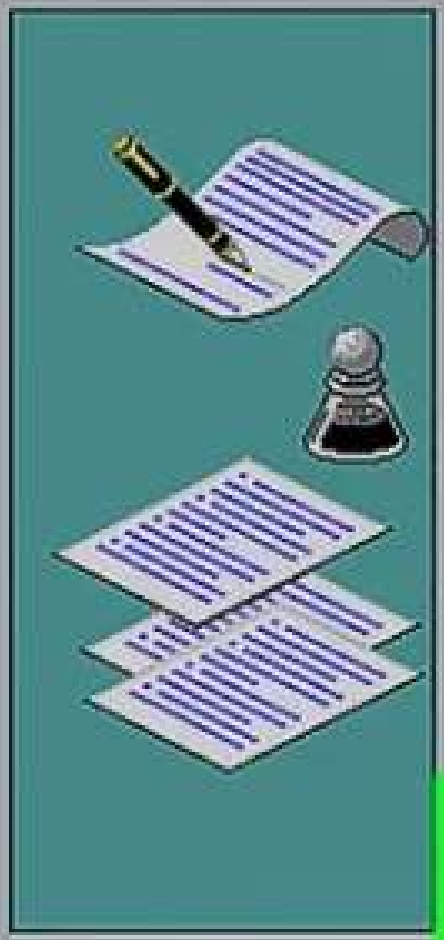
< Back | Next > | Cancel


Windows 98 Setup
Preparing to run Windows 98 Setup
Collecting information about your computer
Copying Windows 98 files to your computer
Restarting your computer
Setting up hardware and finalizing settings
Estimated time remaining:
20 minutes
Windows 98 Setup is now trying to restart your computer. If your computer does not respond for a long time, turn it off and then back on.
Restart Computer
Setup needs to restart your computer to continue. Your computer will automatically restart in 15 seconds or when you click 'Restart Now'.
Restart Now


Microsoft Windows 98 CD-ROM Startup Menu
1. Boot from Hard Disk
2. Boot from CD-ROM
Enter your choice:


Windows 98 Setup Wizard
User Information
Type your name below. If you want, you can also type the name of the company you work for.
Name:
wikihow.com
Company:
wikihow.com
< Back
Next >
Cancel

Windows 98 Setup Wizard



License Agreement

Please read the following License Agreement. You must accept the Agreement to continue Setup.

MICROSOFT WINDOWS 98 SECOND EDITION

END-USER LICENSE AGREEMENT FOR
MICROSOFT DESKTOP OPERATING SYSTEMS

IMPORTANT-READ CAREFULLY: This End-User License Agreement ("EULA") is a legal agreement between you

(•) I accept the Agreement
( ) I don't accept the Agreement

Press the PAGE DOWN key to see more text...

< Back | Next > | Cancel

Windows 98 Setup

Windows Product Key

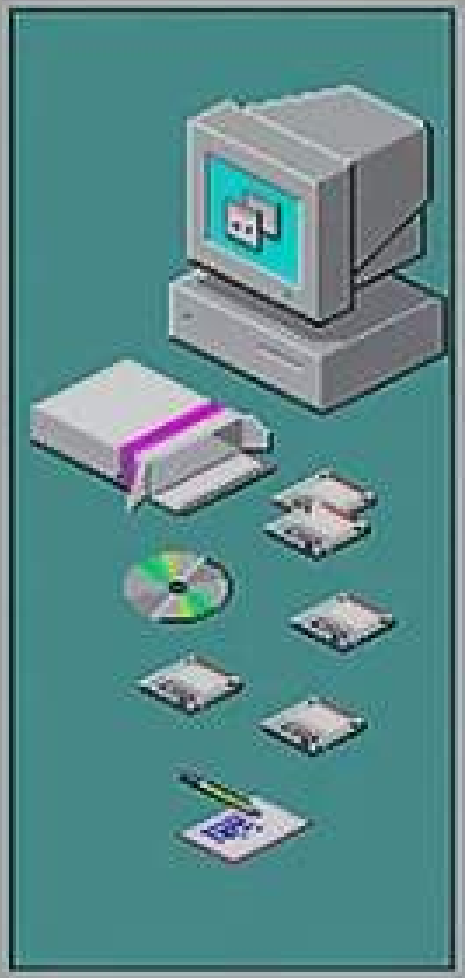
Your Microsoft Windows 98 Product Key is included with your computer documentation or with your Windows 98 software. This Windows Product Key can be found on the lower right of the Certificate of Authenticity, immediately above the bar code as is shown in the picture on the left.

Type the Product Key, excluding the dashes, in the boxes below.

Help...

< Back | Next > | Cancel

Windows 98 Setup Wizard



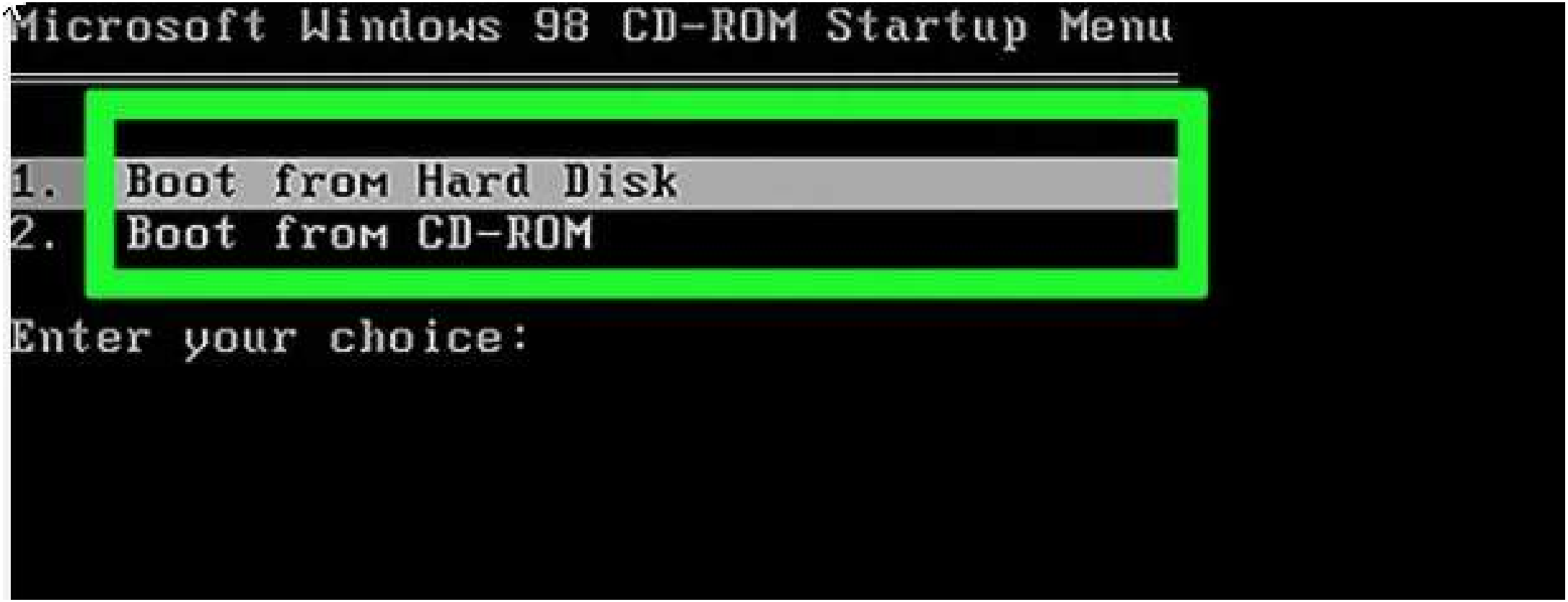
Start Wizard

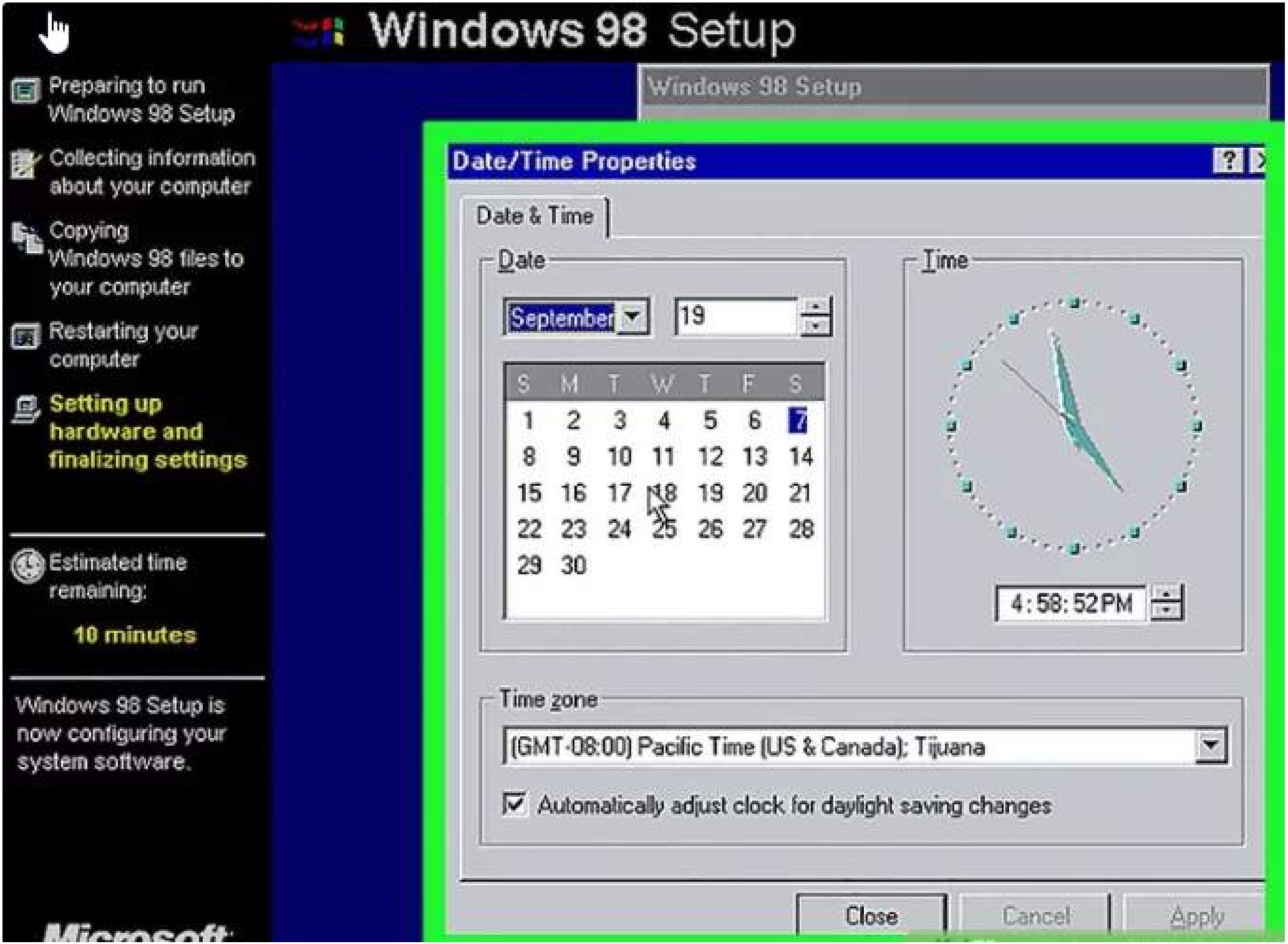
Windows 98 saved all information.

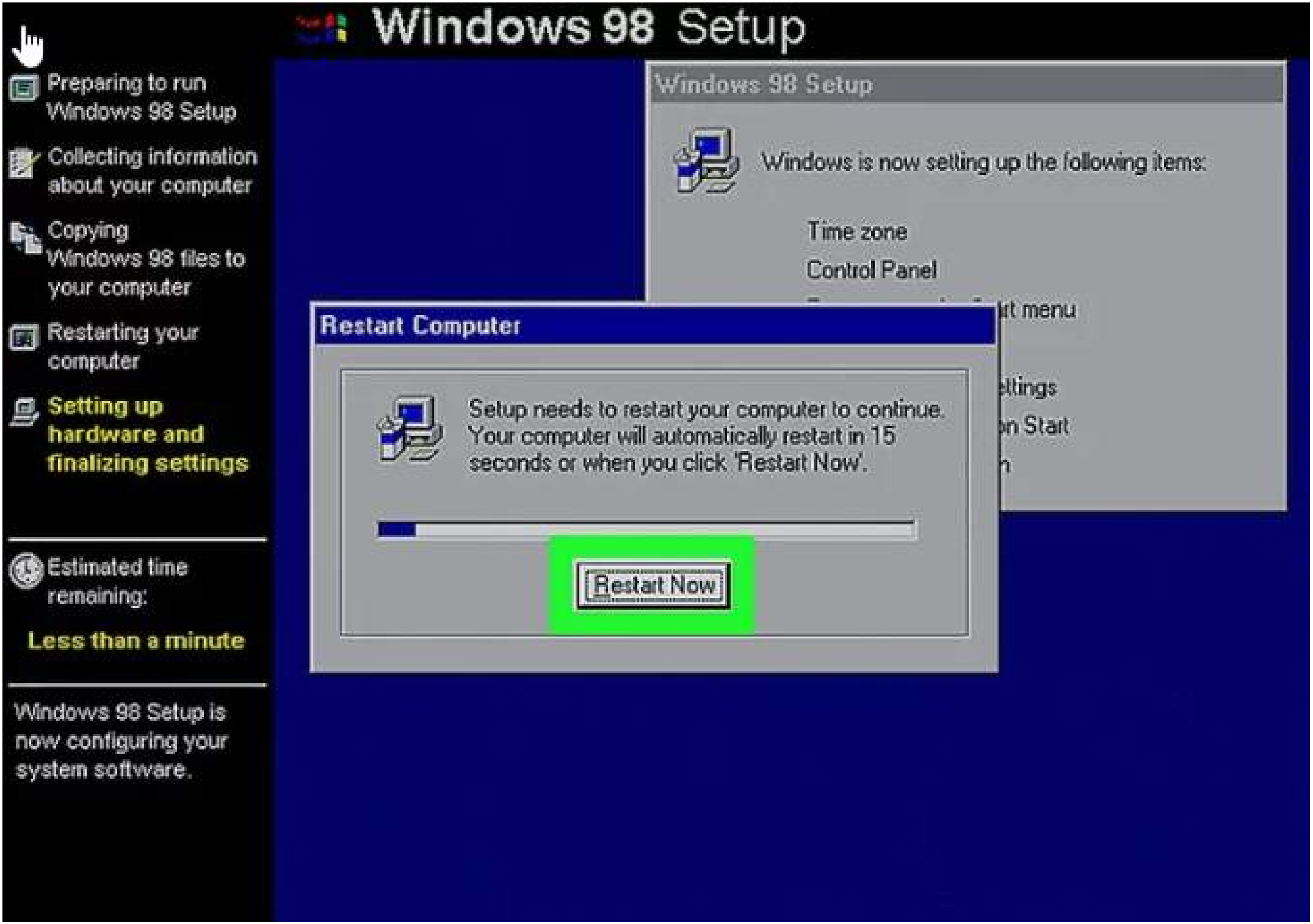
Click Finish to continue starting Windows 98.

< Back

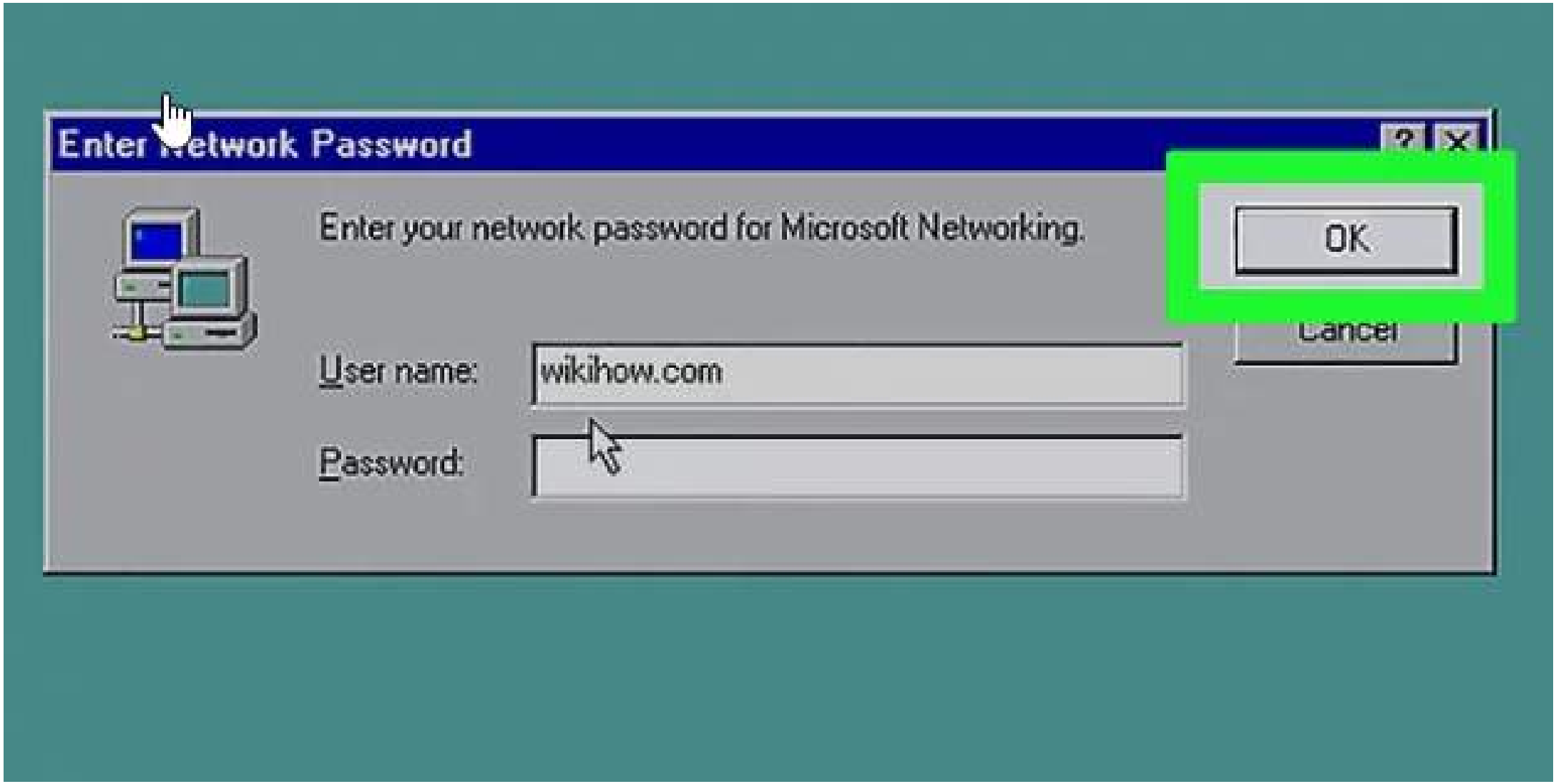
Finish


Microsoft Windows 98 CD-ROM Startup Menu
1. Boot from Hard Disk
2. Boot from CD-ROM
Enter your choice:


Windows 98 Setup
Preparing to run Windows 98 Setup
Collecting information about your computer
Copying Windows 98 files to your computer
Restarting your computer
Setting up hardware and finalizing settings
Estimated time remaining:
10 minutes
Windows 98 Setup is now configuring your system software.
Windows 98 Setup
Date/Time Properties
Date & Time
Date
September
19
S M T W T F S
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
Time
4:58:52 PM
Time zone
(GMT-08:00) Pacific Time (US & Canada); Tijuana
Automatically adjust clock for daylight saving changes
Close
Cancel
Apply


Windows 98 Setup
Preparing to run Windows 98 Setup
Collecting information about your computer
Copying Windows 98 files to your computer
Restarting your computer
Setting up hardware and finalizing settings
Estimated time remaining:
Less than a minute
Windows 98 Setup is now configuring your system software.
Windows 98 Setup
Windows is now setting up the following items:
Time zone
Control Panel
Restart Computer
Setup needs to restart your computer to continue. Your computer will automatically restart in 15 seconds or when you click 'Restart Now'.
Restart Now


Microsoft Windows 98 CD-ROM Startup Menu
1. Boot from Hard Disk
2. Boot from CD-ROM
Enter your choice:


Enter Network Password
Enter your network password for Microsoft Networking.
OK
Cancel
User name:
wikihow.com
Password:


My Computer
Online Services
My Documents
Internet Explorer
Network Neighborhood
Recycle Bin
Setup MSN Internet A...
Welcome to Windows 98
Microsoft Windows98
CONTENTS
Register Now
Connect to the Internet
Discover Windows 98
Maintain Your Computer
Welcome
Welcome to the exciting new world of Windows 98, where your computer desktop meets the Internet!
Sit back and relax as you take a brief tour of the options available on this screen.
If you want to explore an option, just click it.
Show this screen each time Windows 98 starts.
Close

Practical No :- 03

Aim :- Execute linux command . Man, apropos, clear, ls, MKdir, cd, rmdir, pwd, rm, touch, mv, tr, wc, sort, grep, wall, write, who, chmod, useradd, usermod, Kill, ssh, ftp, telnet.

Theory :-

The function & syntax of each command are mentioned below :-

(1) man :- This command is used to display user manual of any command that can run on terminal.

Syntax :- $ man

(2) apropos :- This command is used to find exact command from the key words.

Syntax :- $ apropos [Keyword]

(3) Clear :- This command is used to clear the terminal screen.

Syntax :- $ clear.

(4) ls :- This command is used to list directo -ries.

Syntax :- $ ls

Page No.
Date : / /

(5) mkdir :- This command helps the user to create file or multiple files and set permission for them.
Syntax :- $ mkdir [option] [directories]

(6) cd :- This linux command is used to change current working directories.
Syntax :- $ cd [directory path]

(7) rmdir :- The command is used to remove empty directories.
Syntax :- $ rmdir ~~Everbo~~ [directory]

(8) Pwd :- It prints the path of working directory starting from root.
Syntax :- $ pwd

(9) rm :- The command removes the files from the filesystem. Once the file is deleted, it can't be recovered.
Syntax :- $ rm [file-name]

(10) touch :- This command is used to create, change, or modify timespan of a file

Syntax :- $ touch [filename]

(11) mv :- This command is used to move directory or files from one location to another.

syntax :- $ mv [filename] [path]

(12) tr :- This command is used to translate or delete characters.

Syntax :- $ tr [option 1] set 1 [set 2]

(13) WC :- This command is used to count number of lines of given file or standard input or print result.

syntax :- wc [file_name]

(14) Sort :- This command is used to arrange record in particular order

syntax :- $ sort [filename]

(15) grep :- This command is used by user to search specific string of character in a specific file.

syntax :- $ grep [string] [file]

(16) Wall :- This command line in linux is used to write message to all other use

syntax :- $ wall [-n] [-t timeout] [message | file]

(17) write :- The linux command is used to send message to another user.
Syntax :- $ write user [tty]

(18) who :- This command is used to find following information :-
① time to last system boot.
② Current run level of system.
③ List of logged in user & more.
④ Remote host name of user.
Syntax :- $ who [option] [filename]

(19)

(19) Chmod :- This command is used to change access of the file.
Syntax :- $ chmod [reference] [operator] [mode] file

(20) useradd :- This command allow admin user of ~~to~~ ~~add~~ the system to add users.
Syntax :- $ useradd [option] [username]

(21) Usermod :- it is used to modify user.
Syntax :- $ usermod [option] [user]

(22) Kill :- The command sends signal to processor which terminates the process.
Syntax :- $ kill

(23) SSH :- SSH stands of Secure Shell and it is used to securly Connect to the remote user, server or system.

Syntax :- $ ssh [user]@[IP]

(24) ftp :- FTP stands for File Transfer protocol it used to Transfer file Is and from the remote server.

(25) telnet :- It is used to Connect remote system over TCP/IP. It is very old protocol.

Syntax :- $ telnet [Server-IP-address]

Conclusion :- Here, we have executed all the 25 linux operating system command with different functions & understood there use respectively.

blackstorm@BLACKSTORM: ~

MAN(1) Manual pager utils MAN(1)

NAME

man - an interface to the on-line reference manuals

SYNOPSIS

man [-C file] [-d] [-D] [--warnings[=warnings]] [-R encoding] [-L locale] [-m system[,...]] [-M path] [-S list] [-e extension] [-i|-I] [--regex|--wildcard] [--names-only] [-a] [-u] [--no-subpages] [-P pager] [-r prompt] [-7] [-E encoding] [--no-hyphenation] [--no-justification] [-p string] [-t] [-T[device]] [-H[browser]] [-X[dpi]] [-Z] [[section] page[.section] ...] ...
man -k [apropos options] regexp ...
man -K [-w|-W] [-S list] [-i|-I] [--regex] [section] term ...
man -f [whatis options] page ...
man -l [-C file] [-d] [-D] [--warnings[=warnings]] [-R encoding] [-L locale] [-P pager] [-r prompt] [-7] [-E encoding] [-p string] [-t] [-T[device]] [-H[browser]] [-X[dpi]] [-Z] file ...
man -w|-W [-C file] [-d] [-D] page ...
man -c [-C file] [-d] [-D] page ...
man [-?V]

DESCRIPTION

man is the system's manual pager. Each page argument given to man is normally the name of a program, utility or function. The manual page associated with each of these arguments is then found and displayed. A section, if provided, will direct man to look only in that section of the manual. The default action is to search in all of the available sections following a pre-defined order ("1 n l 8 3 2 3posix 3pm 3perl 3am 5 4 9 6 7" by default, unless overridden by the SECTION directive in /etc/manpath.config), and to show only the first page found, even if page exists in several sections.

The table below shows the section numbers of the manual followed by the types of pages they contain.

1 Executable programs or shell commands
2 System calls (functions provided by the kernel)
3 Library calls (functions within program libraries)
4 Special files (usually found in /dev)
5 File formats and conventions eg /etc/passwd
6 Games
7 Miscellaneous (including macro packages and conventions), e.g. man(7), groff(7)
8 System administration commands (usually only for root)
9 Kernel routines [Non standard]

A manual page consists of several sections.

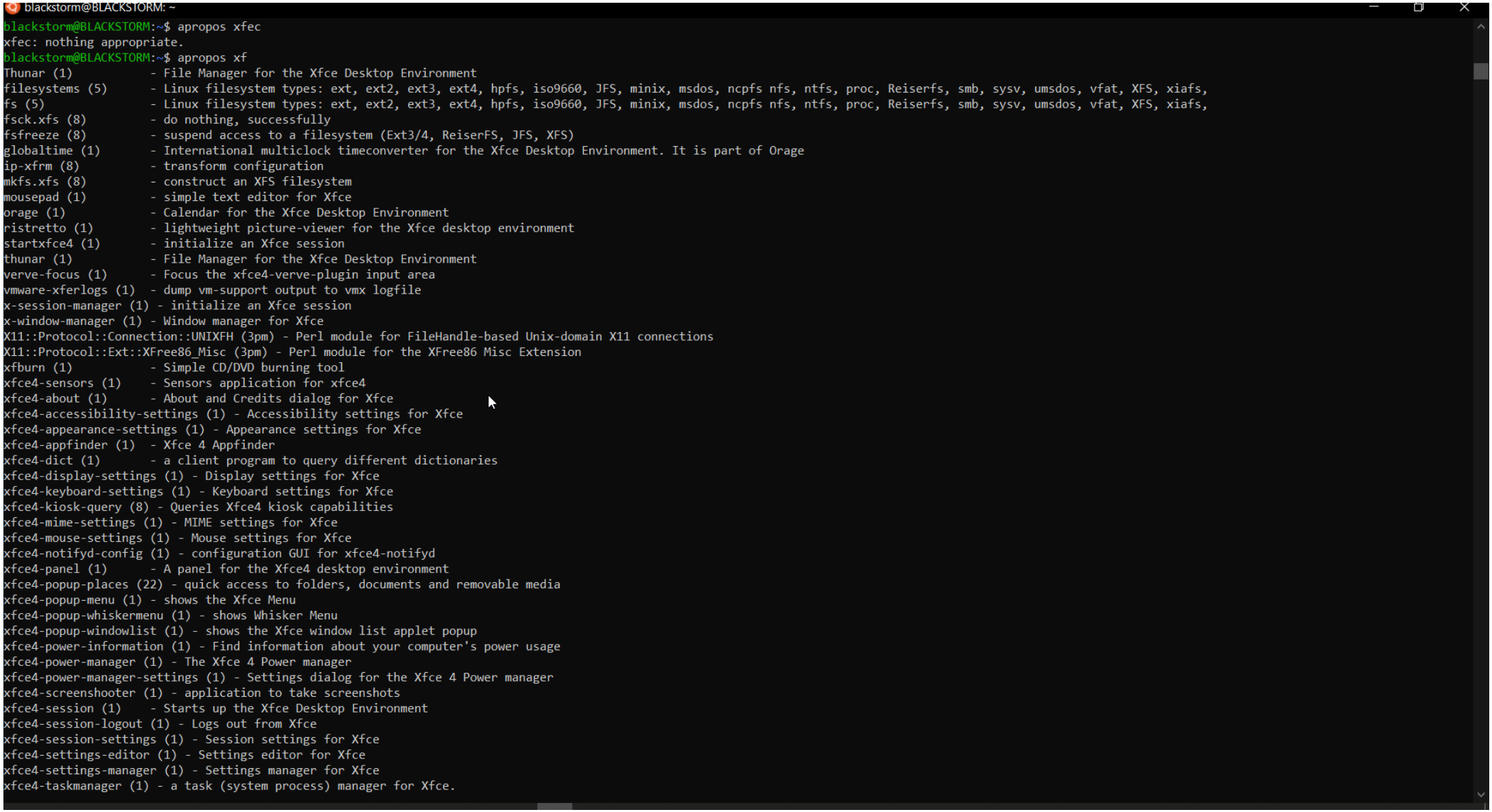
Conventional section names include NAME, SYNOPSIS, CONFIGURATION, DESCRIPTION, OPTIONS, EXIT STATUS, RETURN VALUE, ERRORS, ENVIRONMENT, FILES, VERSIONS, CONFORMING TO, NOTES, BUGS, EXAMPLE, AUTHORS, and SEE ALSO.

The following conventions apply to the SYNOPSIS section and can be used as a guide in other sections.

bold text	type exactly as shown.
italic text	replace with appropriate argument.
[-abc]	any or all arguments within [ ] are optional.
-a|-b	options delimited by | cannot be used together.
argument ...	argument is repeatable.
[expression] ...	entire expression within [ ] is repeatable.

Exact rendering may vary depending on the output device. For instance, man will usually not be able to render italics when running in a terminal, and will typically use underlined or coloured text

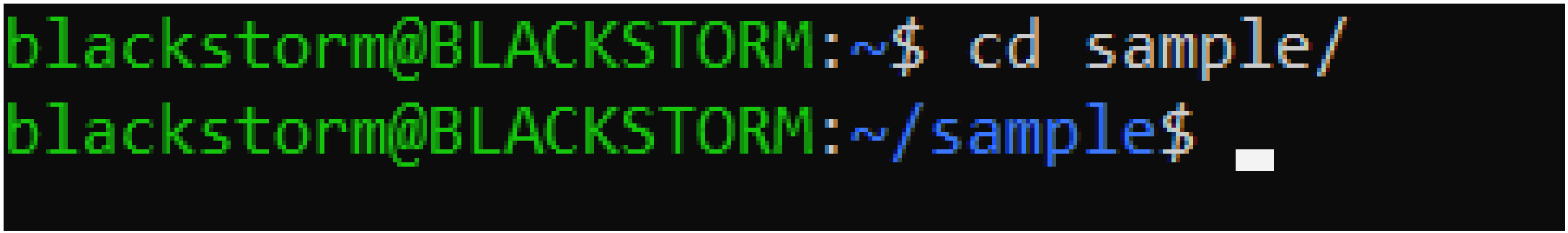
Manual page man(1) line 1 (press h for help or q to quit)

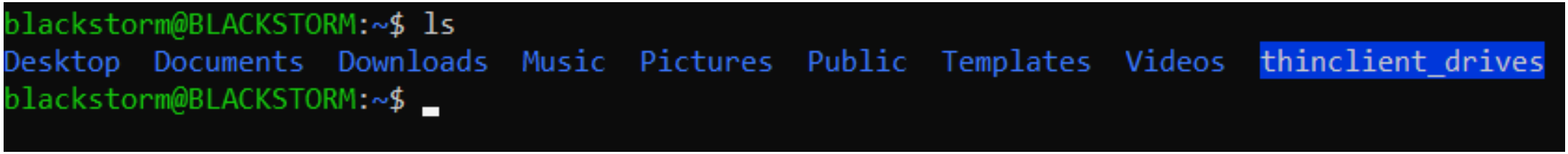

blackstorm@BLACKSTORM: ~
blackstorm@BLACKSTORM:~$ apropos xfec
xfec: nothing appropriate.
blackstorm@BLACKSTORM:~$ apropos xf
Thunar (1)   - File Manager for the Xfce Desktop Environment
filesystems (5)   - Linux filesystem types: ext, ext2, ext3, ext4, hpfs, iso9660, JFS, minix, msdos, ncpfs nfs, ntfs, proc, Reiserfs, smb, sysv, umsdos, vfat, XFS, xiafs,
fs (5)   - Linux filesystem types: ext, ext2, ext3, ext4, hpfs, iso9660, JFS, minix, msdos, ncpfs nfs, ntfs, proc, Reiserfs, smb, sysv, umsdos, vfat, XFS, xiafs,
fsck.xfs (8)   - do nothing, successfully
fsfreeze (8)   - suspend access to a filesystem (Ext3/4, ReiserFS, JFS, XFS)
globaltime (1)   - International multiclock timeconverter for the Xfce Desktop Environment. It is part of Orage
ip-xfrm (8)   - transform configuration
mkfs.xfs (8)   - construct an XFS filesystem
mousepad (1)   - simple text editor for Xfce
orage (1)   - Calendar for the Xfce Desktop Environment
ristretto (1)   - lightweight picture-viewer for the Xfce desktop environment
startxfce4 (1)   - initialize an Xfce session
thunar (1)   - File Manager for the Xfce Desktop Environment
verve-focus (1)   - Focus the xfce4-verve-plugin input area
vmware-xferlogs (1) - dump vm-support output to vmx logfile
x-session-manager (1) - initialize an Xfce session
x-window-manager (1) - Window manager for Xfce
X11::Protocol::Connection::UNIXFH (3pm) - Perl module for FileHandle-based Unix-domain X11 connections
X11::Protocol::Ext::XFree86_Misc (3pm) - Perl module for the XFree86 Misc Extension
xfburn (1)   - Simple CD/DVD burning tool
xfce4-sensors (1)   - Sensors application for xfce4
xfce4-about (1)   - About and Credits dialog for Xfce
xfce4-accessibility-settings (1) - Accessibility settings for Xfce
xfce4-appearance-settings (1) - Appearance settings for Xfce
xfce4-appfinder (1) - Xfce 4 Appfinder
xfce4-dict (1)   - a client program to query different dictionaries
xfce4-display-settings (1) - Display settings for Xfce
xfce4-keyboard-settings (1) - Keyboard settings for Xfce
xfce4-kiosk-query (8) - Queries Xfce4 kiosk capabilities
xfce4-mime-settings (1) - MIME settings for Xfce
xfce4-mouse-settings (1) - Mouse settings for Xfce
xfce4-notifyd-config (1) - configuration GUI for xfce4-notifyd
xfce4-panel (1)   - A panel for the Xfce4 desktop environment
xfce4-popup-places (22) - quick access to folders, documents and removable media
xfce4-popup-menu (1) - shows the Xfce Menu
xfce4-popup-whiskermenu (1) - shows Whisker Menu
xfce4-popup-windowlist (1) - shows the Xfce window list applet popup
xfce4-power-information (1) - Find information about your computer's power usage
xfce4-power-manager (1) - The Xfce 4 Power manager
xfce4-power-manager-settings (1) - Settings dialog for the Xfce 4 Power manager
xfce4-screenshooter (1) - application to take screenshots
xfce4-session (1)   - Starts up the Xfce Desktop Environment
xfce4-session-logout (1) - Logs out from Xfce
xfce4-session-settings (1) - Session settings for Xfce
xfce4-settings-editor (1) - Settings editor for Xfce
xfce4-settings-manager (1) - Settings manager for Xfce
xfce4-taskmanager (1) - a task (system process) manager for Xfce.

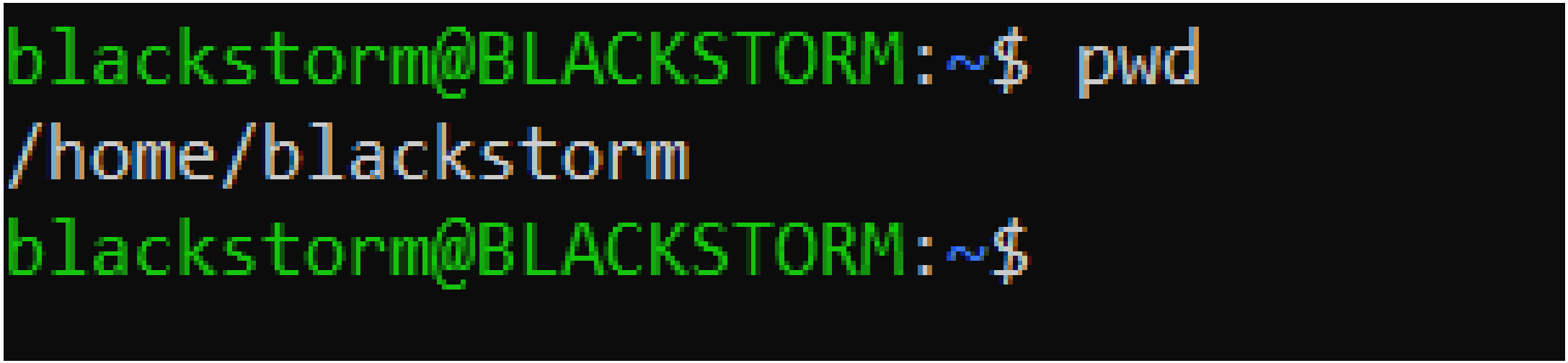

blackstorm@BLACKSTORM:~$ clear

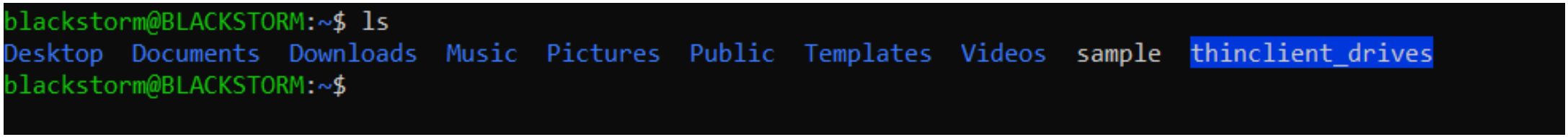

blackstorm@BLACKSTORM:~$ ls
Desktop Documents Downloads Music Pictures Public Templates Videos thinclient_drives

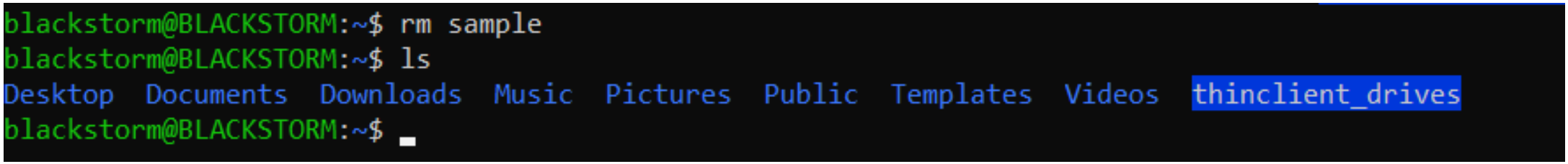

blackstorm@BLACKSTORM:~$ mkdir sample
blackstorm@BLACKSTORM:~$ ls
Desktop Documents Downloads Music Pictures Public Templates Videos sample thinclient_drives

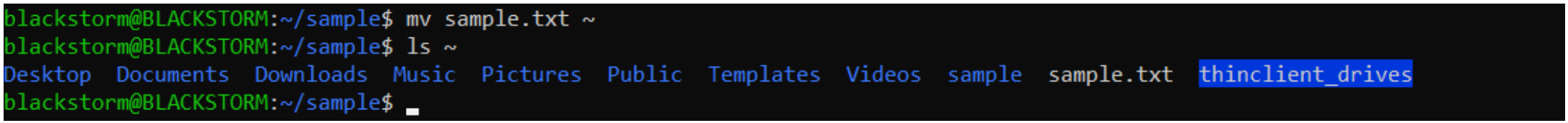

blackstorm@BLACKSTORM:~$ cd sample/
blackstorm@BLACKSTORM:~/sample$


blackstorm@BLACKSTORM:~$ ls
Desktop Documents Downloads Music Pictures Public Templates Videos thinclient_drives
blackstorm@BLACKSTORM:~$

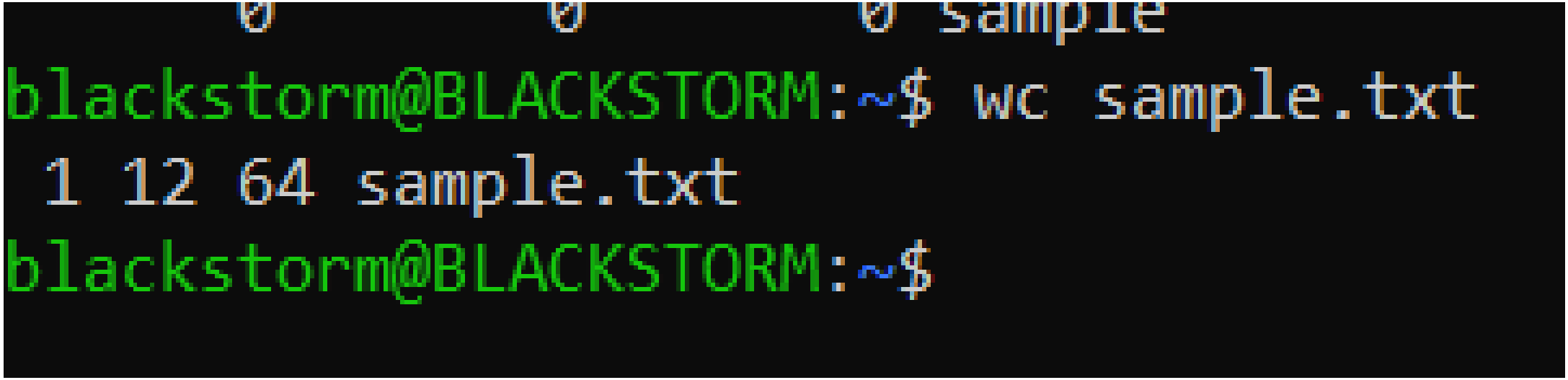

blackstorm@BLACKSTORM:~$ pwd
/home/blackstorm
blackstorm@BLACKSTORM:~$

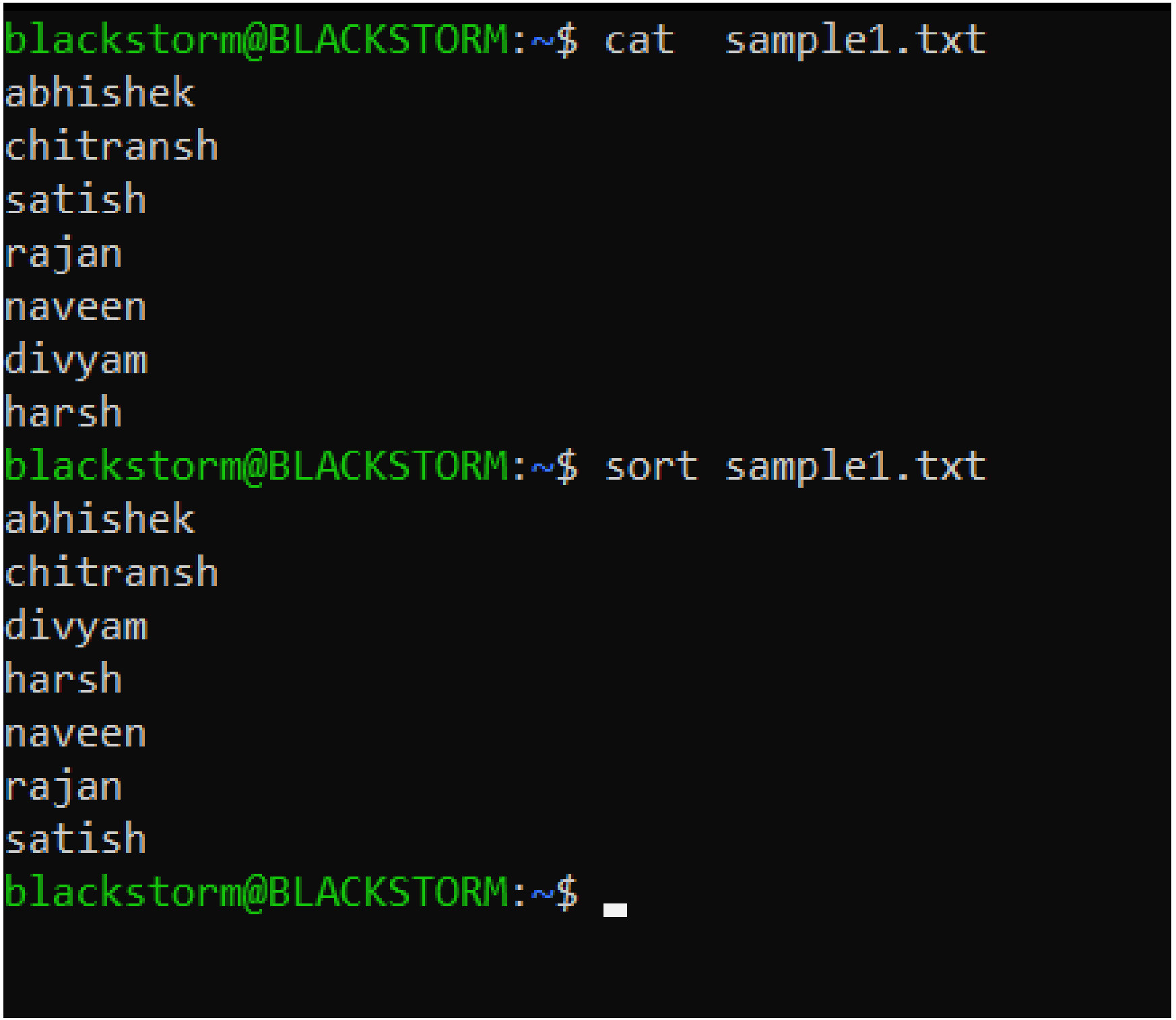

blackstorm@BLACKSTORM:~$ ls
Desktop Documents Downloads Music Pictures Public Templates Videos sample thinclient_drives
blackstorm@BLACKSTORM:~$

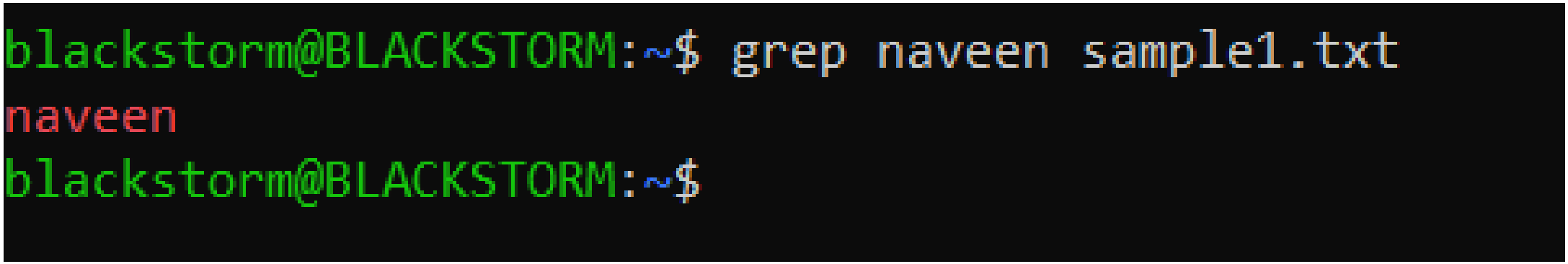

blackstorm@BLACKSTORM:~$ rm sample
blackstorm@BLACKSTORM:~$ ls
Desktop Documents Downloads Music Pictures Public Templates Videos thinclient_drives
blackstorm@BLACKSTORM:~$


blackstorm@BLACKSTORM:~/sample$ mv sample.txt ~
blackstorm@BLACKSTORM:~/sample$ ls ~
Desktop Documents Downloads Music Pictures Public Templates Videos sample sample.txt thinclient_drives
blackstorm@BLACKSTORM:~/sample$

```
blackstorm@BLACKSTORM:~$ cat sample.txt
This is a sample of TR command in LINUX. PLease give Attention
blackstorm@BLACKSTORM:~$ cat sample.txt | tr "[a-z]" "[A-Z]"
THIS IS A SAMPLE OF TR COMMAND IN LINUX. PLEASE GIVE ATTENTION
blackstorm@BLACKSTORM:~$
```

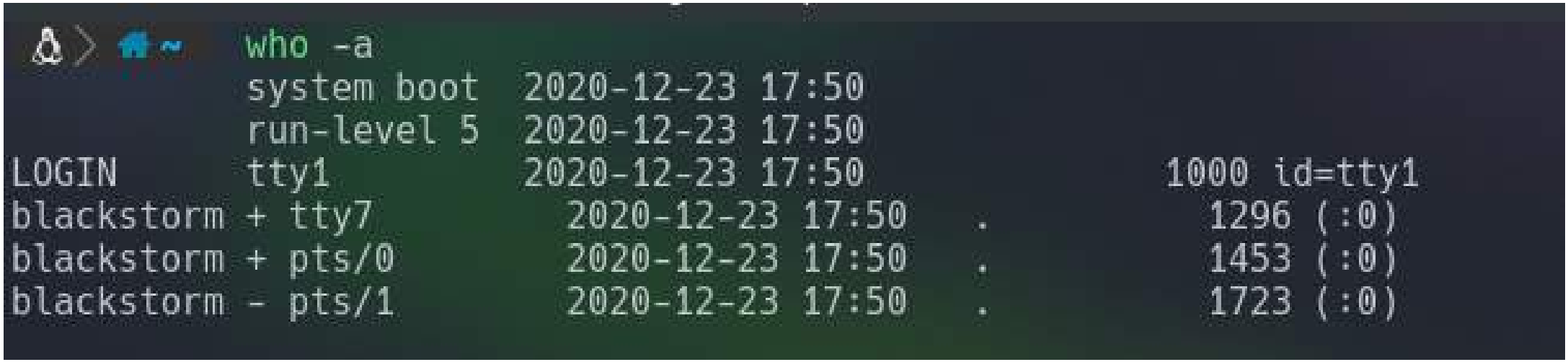

0 0 0 sample
blackstorm@BLACKSTORM:~$ wc sample.txt
1 12 64 sample.txt
blackstorm@BLACKSTORM:~$


blackstorm@BLACKSTORM:~$ cat sample1.txt
abhishek
chitransh
satish
rajan
naveen
divyam
harsh
blackstorm@BLACKSTORM:~$ sort sample1.txt
abhishek
chitransh
divyam
harsh
naveen
rajan
satish
blackstorm@BLACKSTORM:~$

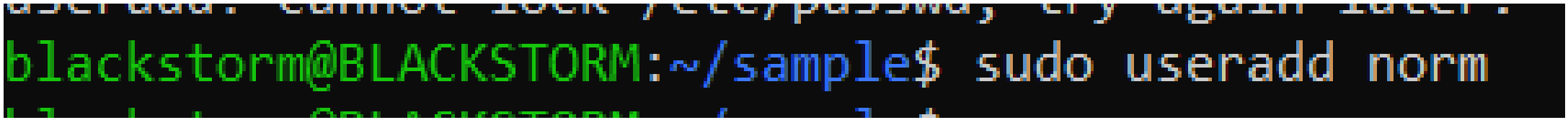

blackstorm@BLACKSTORM:~$ grep naveen sample1.txt
naveen
blackstorm@BLACKSTORM:~$

```
blackstorm@BLACKSTORM:~$ wall "This is a sample text for wall command"
```


blackstorm@BLACKSTORM:~$ write root
write: root is not logged in


who -a
system boot 2020-12-23 17:50
run-level 5 2020-12-23 17:50
LOGIN tty1 2020-12-23 17:50 1000 id=tty1
blackstorm + tty7 2020-12-23 17:50 . 1296 (:0)
blackstorm + pts/0 2020-12-23 17:50 . 1453 (:0)
blackstorm - pts/1 2020-12-23 17:50 . 1723 (:0)

```
blackstorm@BLACKSTORM:~/sample$ ls -l
total 0
-rw-r--r-- 1 blackstorm blackstorm 299 Dec 23 17:39 sample.txt
blackstorm@BLACKSTORM:~/sample$ chmod 777 sample.txt
blackstorm@BLACKSTORM:~/sample$ ls -l
total 0
-rwxrwxrwx 1 blackstorm blackstorm 299 Dec 23 17:39 sample.txt
blackstorm@BLACKSTORM:~/sample$
```


blackstorm@BLACKSTORM:~/sample$ sudo useradd norm

```
blackstorm@BLACKSTORM:~/sample$ sudo useradd --password "norm" norm
blackstorm@BLACKSTORM:~/sample$ su norm
Password:
su: Authentication failure
blackstorm@BLACKSTORM:~/sample$ su norm
Password:
su: Authentication failure
blackstorm@BLACKSTORM:~/sample$ sudo passwd norm
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
blackstorm@BLACKSTORM:~/sample$ su norm
Password:
$ id
uid=1001(norm) gid=1001(norm) groups=1001(norm)
$
```

```
blackstorm@BLACKSTORM:~/sample$ sudo usermod -g blackstorm norm
blackstorm@BLACKSTORM:~/sample$ groups
blackstorm adm dialout cdrom floppy sudo audio dip video plugdev lxd netdev
blackstorm@BLACKSTORM:~/sample$ su norm
Password:
$ id
uid=1001(norm) gid=1000(blackstorm) groups=1000(blackstorm)
$
```

```
blackstorm@BLACKSTORM:~/sample$ dd if=/dev/zero of=/dev/null &
[1] 395
blackstorm@BLACKSTORM:~/sample$ ps
  PID TTY          TIME CMD
   10 tty1     00:00:00 bash
  395 tty1     00:00:03 dd
  396 tty1     00:00:00 ps
blackstorm@BLACKSTORM:~/sample$ kill -n 9 395
[1]+  Killed                  dd if=/dev/zero of=/dev/null
blackstorm@BLACKSTORM:~/sample$ ps
  PID TTY          TIME CMD
   10 tty1     00:00:00 bash
  397 tty1     00:00:00 ps
blackstorm@BLACKSTORM:~/sample$
```

Page No.
Date : / /

Practical No :- 04

Aim :- Develope, debug and execute a C program to simulate the FCFS CPU scheduling algorithm to find turnaround and waiting time.

Theory :-

FCFS scheduling :-

First come first serve is an operating system scheduling algorithm that automatically executes queued requests and process in order of their arrival. It is the easiest & simple CPU scheduling algorithm.

In this type of algorithm, process which requests the CPU first gets the CPU algorithm first. This is managed by first - in - first - out (FIFO) queue.

FCFS follows non-premetive scheduling which mean once the CPU is allocated to a process it does not leave the CPU until the process will not get terminated or may get halted due to some i/o interrupt.

Page No.
Date: / /

Algorithm:-

Start:- Step ① :- fun^c int waitingtime (int proc[], int n, int burst_time[], int wait_time[])

Set wait_time [0] = 0

Loop For i = 1 and i < n and i++

set wait_time [i] = burst_time [i-1] + wait_time[i-1]

End FOR

Step ②:- In function int turnaroundtime (int proc[], int n, int burst_time[], int wait_time[], int tat[])

Loop For i = 0 and i<n and i++

set tat[i] = burst_time [i] + wait_time [i]

End For

Step ③:- In function int avgtime (int proc[], int n, int burst_time[])

Declare and initialize wait_time[n], tat[n], total_wt=0, total_tat = 0;

Call waiting_time (proc, n, burst_time, wait_time, tat)

Call turnaroundtime (proc, n, burst_time, wait_time, tat)

Loop For i=0 and i<n and i++

set total_wt = total_wt + wait_time[i]

Set total_tat = total_tat + tat[i]

print process number, burst time, wait time and turnaround time

End FOR.

print "Average waiting time = i.e total_wt \n

print "Average turn around time = i.e total_tat \n

Step (4):- In int main ()

Declare impu int proc [] = {1, 2, 0}

Declare and initialize n = sizeof proc / sizeof proc[0]

Declare and initialize burst-time [] = {10, 5, 8}

Call avgtime (proc, n, burst-time)

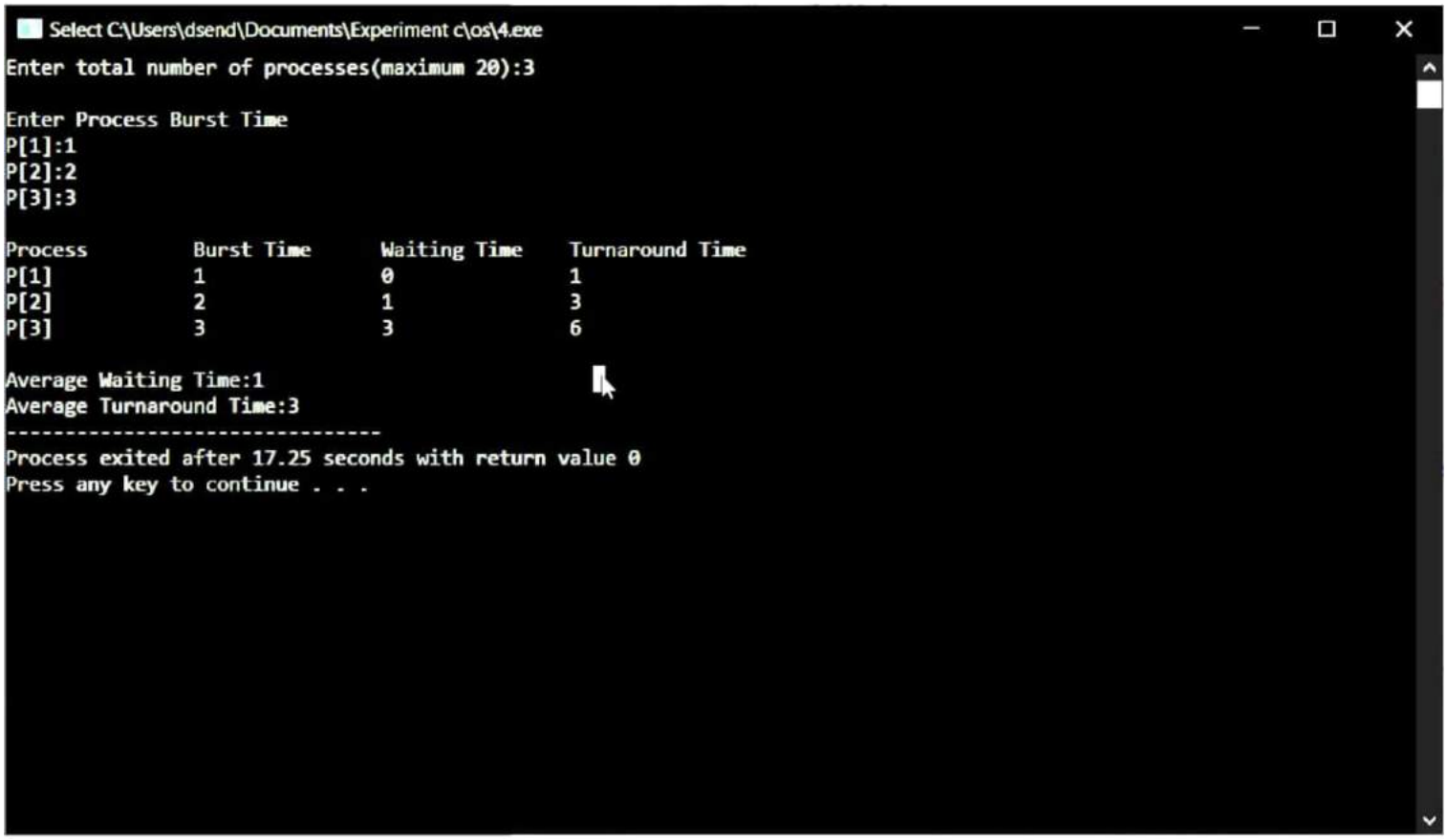
Stop.

Program :-

```
#include <stdio.h>
int main () {
    int n, bt[20], wt[20], tat[20], awwt = 0, awtat = 0
        i, j;
    printf ("Enter total number of processes (maximum
            20) : ");
    scanf ("%d", &n);
    printf ("\n Enter process Burst time \n");
    for (i=0; i<n; i++) {
        printf (" P[%d]: ", i+1);
        scanf ("%d", &bt[i]);
    }
    wt[0] = 0;

    for (i=1; i<n; i++) {
        wt[i] = 0;
        For(j=0; j<i; j++)
            wt[i] += bt[j];
    }
    printf ("\n Process \t\t Burst time \t Waiting Time \t
            Turnaround Time ");

    for (i=0, i<n, i++) {
        tat[i] = bt[i] + wt[i];
        awwt += wt[i];
        awtat += tat[i];
        printf ("\n P[%d] \t\t %d \t\t %d", i+1, bt[i],
            wt[i], tat[i]);
    }
```

```
        awt /= i;
        atat /= i;
        printf ("\n P[%d]\t\t %d\t\t %d\t\t %d", i+1,
                bt[i], wt[i], tat[i]);
    printf ("\n\n Average Waiting Time : %d", awt);
    printf ("\n Average Turnaround Time : %d", atat);
    return 0;
}
```

Conclusion :- Hence, we ~~exe~~ successfully developed and executed FCFS CPU scheduling ~~algorithm~~ program.


Select C:\Users\dsend\Documents\Experiment c\os\4.exe
Enter total number of processes(maximum 20):3
Enter Process Burst Time
P[1]:1
P[2]:2
P[3]:3
Process Burst Time Waiting Time Turnaround Time
P[1] 1 0 1
P[2] 2 1 3
P[3] 3 3 6
Average Waiting Time:1
Average Turnaround Time:3
Process exited after 17.25 seconds with return value 0
Press any key to continue . . .

Page No.
Date : / /

Practical No :- 05

Aim :- Develope debug & execute a C program to simulate the SJF CPU Scheduling algorithm to find turnaround & waiting time.

Theory :-

SJF Schedulling :-

Shortest Job first (SJF) is an algorithm in which the process having the smallest execution time is chosen for the next execution. This schedulling method can be premitive or non-premitive It significantly reduce the average waiting time for other process awaiting execution.

There are two types of SJF ;-

1. Non-premitive SJF.
2. Premitive SJF.

SJF algorithm can be premptive as well as non-preemptive. Premetive schedulling is also known as Short-remaining-time-first schedulling. In preemptive approach. the new process arises when there is already executing process. If the burst of newly arriving process is lesser than the burst time of executing process than scheduler will preempt the execution of the process with lesser burst time.

Page No.
Date : / /

Algorithm :-

Start

Step ① :- Declare a struct process
Declare pid, bt, art

Step ② :- In function findaroundtime (Process proc [], int n
Int wt [], Int tat [])
Loop For i = 0 and i<n and i++
set tat [i] = proc [i] . bt + wt [i]

Step ③ :- In function findWaitingTime (process proc [], int
int wt [])
Declare rt [n]
Loop For i = 0 and i<n and i++
set rt [i] = proc [i]. bt
Set Complete = 0, t = 0, minm = INT_MAX
Set Shortest = 0, finish-time
Set bool check = false
loop while (complete != n)
Loop for j = 0 and j<n and j++
If (proc [j]. art <= t) && (rt [j] < minm)
&& rt [j] > 0 then,
Set minm = rt [j]
Set shortest = j
Set check = true
If check == false then,
Increment t by 1
continue
decrement the value of rt [shortest] by 1

Page No.
Date : / /

Set minm = wt [shortest]
If minm == 0 then
Set minm = INT_MAX
If wt [shortest] == 0 then
Increment complete by 1
Set check = false
Set finish-time = t + 1
Set wt [shortest] = finish-time - proc [shortest].bt
- proc [shortest]. art
If wt [shortest] < 0
Set wt [shortest] = 0
Increment t by 1

Step ④:- In function findavgtime (process proc [], int n)
Declare and set wt [n], tat [n], total-wt = 0,
total-tat = 0
Call findWaitingTime (proc, n, wt)
Call findturnaroundtime (proc, n, wt, tat)
Loop for i = 0 and i<n and i++
Set total-wt = total-wt + wt [i]
Set total-tat = total-tat + tat [i]
print proc [i].pid, proc [i].bt, wt [i], tat [i]
print Average Waiting Time i.e, total_wt \n
print Average turn around time is, total-tat \n

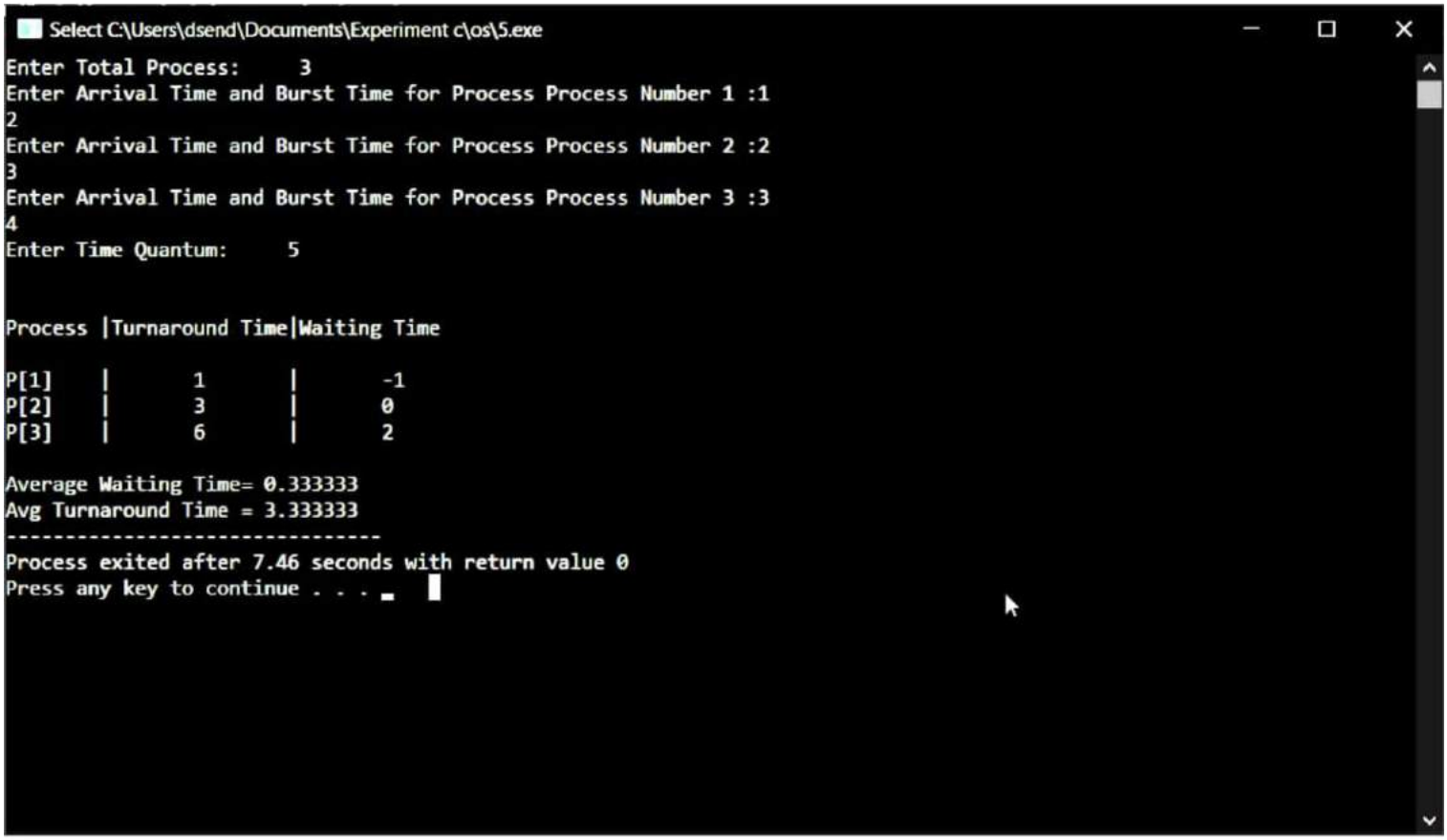
Step ⑤:- Int function int main ()
Declare and set process proc [] = {{1, 5, 1}, {2, 3, 1}
{3, 6, 2}, {4, 5, 3}}
Set n = sizeof (proc) / sizeof (proc [0])
Call find avg time (proc, n)
Stop

Program :-

```
#include <stdio.h>
#include <conio.h>
int main () {
    int count, j, n, time, remain, flag = 0, time
        _quantum;
int wait_time = 0, turnaround_time = 0, at[10], bt[10],
        rt[10];
printf (" Enter total process : \t ");
 scanf ("%d", &n);
 remain = n;
 for (count = 0 ; count < n ; count++) {
     printf (" Enter Arrival Time and Burst
Time for process Process number %d : ", count +1);
scanf ("%d", &at[count]);
scanf ("%d", &bt[count]);
 rt[count] = bt[count];
}

printf (" Enter time Quantum : \t ");
scanf ("%d", &time_quantum);
printf ("\n\nProcess \t | Turnaround Time | waiting Time \n"
for (time = 0, count = 0 ; remain != 0) {
    if (rt[count] <= time_quantum && rt[count] > 0
{
    time += rt[count];
    rt[count] = 0;
    flag = 1;
}
```

```
else if (rt[count] > 0) {
    rt[count] -= time_quantum;
    time t = time_quantum;
  }
if (rt[count] == 0 && flag == 1) {
    remain--;
  printf("P[%d]\t|\t%d\t|\t%d\n", count+1,
  time - at[count], time - at[count - bt[count]]);
  wait_time += time - at[count] - bt[count];
  Turnaround_time += time - at[count];
   flag = 0;

}
  if (count == n-1)
      count = 0;
  else if (at[count + 1] <= time)
      Count++;
  else
      Count = 0;
}

printf("In Average Waiting Time = %f \n", wait_time * 1.0/n);
printf("Avg Turnaround Time = %f ", turnaround_time
   * 1.0/n);

return 0;

}
```

Conclusion:- Hence, we successfully developed and executed SJF CPU Scheduling program.


Select C:\Users\dsend\Documents\Experiment c\os\5.exe
Enter Total Process: 3
Enter Arrival Time and Burst Time for Process Process Number 1 :1
2
Enter Arrival Time and Burst Time for Process Process Number 2 :2
3
Enter Arrival Time and Burst Time for Process Process Number 3 :3
4
Enter Time Quantum: 5
Process |Turnaround Time|Waiting Time
P[1] | 1 | -1
P[2] | 3 | 0
P[3] | 6 | 2
Average Waiting Time= 0.333333
Avg Turnaround Time = 3.333333
Process exited after 7.46 seconds with return value 0
Press any key to continue . . .

Page No.
Date :

Practical :- 06

Aim :- Develope and execute a C program to simulate the Round Robin CPU Scheduling algorithm to find turnaround time and waiting time.

Theory :-

Round Robin Scheduling :-

A Round Robin is a CPU Scheduling algorithm that shares equal portions of resource in a circular orders to each process and handle all process without prioritization. In the round-robin, each process gets a fixed time interval of the slice to utilize the resources or executes its task Called time quantum or time slice. Some of the round-robin processes are pre-empted if it executed in a given time slot, while the rest of the processes go back to the ready queue and wait to run in a circular order with the scheduled time slot until they complete their task. It removes the starvation for each process to achieve CPU Scheduling by proper partitioning of the CPU.

Page No.
Date : / /

Algorithm :-

Step ① :- Organize all process according to their arrival time in the ready queue. The queue structure of the ready queue is based on the FIFO structure to execute all CPU process.

Step ② :- Now, we push the first process from the ready queue to execute its task for a fixed time, allocated by each process that arrives in the queue.

Step ③ :- If the processes can't complete their task within defined time interval or slots because it is stopped by another process that pushes which helps ready queue to execute their task due to arrival time of the next process is reached. Therefore, CPU saved the previous state of the process, which helps to resume from the point where it is interrupted.

Step ④ :- Similarly, the schedular selects another process from the ready queue to execute its tasks. When a process finishes its task within time slot, the process will not go for further execution.

Step ⑤ :- We ~~repeat~~ repeat all the steps to execute the process.

Page No.
Date :

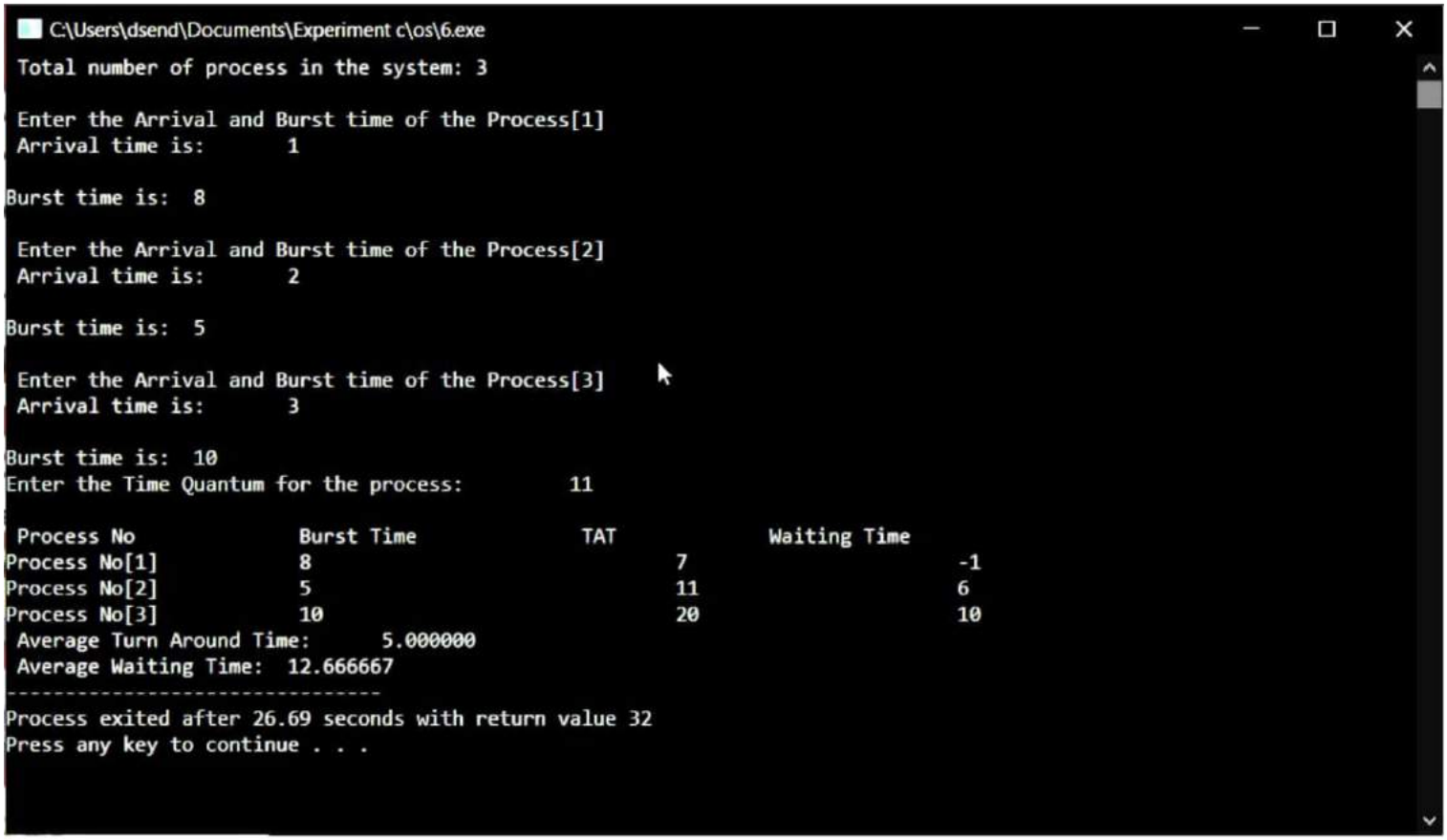
Program :-

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

Void main () {
    int i, NOP, sum = 0, count = 0, y, quant,
        wt = 0, tat = 0, at [10], bt [10], temp [10];
    float avg_wt, avg_tat;
    printf (" Total number of system in the process:"
    Scanf ("%d", &NOP);
    y = NOP;

for (i = 0; i < NOP ; i++) {
    printf ("\n Enter the arrival time and burst time
        of the process [%d] \n ", i+1);
    printf (" Arrival time is: \t ");
    scanf ("%d", &at[i]);
    printf ("\n Burst time is : \t ");
    scanf ("%d", &bt[i]);
    temp (i) = bt [i];
}

printf ("Enter the time quantum for process : \t ");
Scanf ("%d", &quant);

printf (" \n Process No \t\t Burst time \t\t TAT \t\t
        waiting time ");
```

```
for (sum = 0 ; i = 0 ; y != 0) {
    if ( temp [i] <= quant && temp [i] > 0 ) {
            sum += temp [i];
            temp [i] = 0 ;
            count = 1;
    }
    else if ( temp [i] > 0) {
            temp [i] -= quant;
            sum += quant;
    }
    if ( temp [i] == 0 && count == 1) {
        y--;
        printf ("\n Process No [%d] \t\t %d \t\t\t\t %d \t\t\t\t
            %d ", i+1, bt[i] , sum - at [i] , sum - at [i] - bt[i]);
        wt = wt + sum - at[i] - bt [i] ;
        tat = tat + sum - at [i] ;
        count = 0 ;
    }
    if (i == NOP - 1) {
        i=0;
    }
    else if (at [i+1] <= sum ) {
            i++;
    }
    else {
        i=0;
    }
```

```
avg_wt = wt * 1.0/NOP;
avg_tat = tat * 1.0/NOP;

printf("\n Average Turn around time : \t%f", avg_tat);
printf("\n Average Waiting time : \t%f", avg_wt);

getch();

}
```

Conclusion:- Hence, we successfully developed and executed round-robin program.


C:\Users\dsend\Documents\Experiment c\os\6.exe
Total number of process in the system: 3
Enter the Arrival and Burst time of the Process[1]
Arrival time is: 1
Burst time is: 8
Enter the Arrival and Burst time of the Process[2]
Arrival time is: 2
Burst time is: 5
Enter the Arrival and Burst time of the Process[3]
Arrival time is: 3
Burst time is: 10
Enter the Time Quantum for the process: 11
Process No Burst Time TAT Waiting Time
Process No[1] 8 7 -1
Process No[2] 5 11 6
Process No[3] 10 20 10
Average Turn Around Time: 5.000000
Average Waiting Time: 12.666667
Process exited after 26.69 seconds with return value 32
Press any key to continue . . .

Practical No :- 07

Aim :- Developoe, debug and execute a C program to simulate priority CPU Scheduling algorithm. To find turnaround time and waiting time.

Theory :-

Priority Scheduling :-

In priority scheduling, at the time of a arrival of a process in the ready queue, its priority is compared with the priority of the others processes present in the ready queue as well as with the one which is being executed by the CPU at that point of time. The one with the highest priority among all the available processes will be given the CPU next.

The difference between preemptive and non-preemptive priority scheduling is that, the job which is being executed can be stopped at the arrival of a higher priority job.

Once all the jobs are ~~ready~~ available in a ready queue, the alogrith will behave as non-preemptive scheduling, which means the job scheduled will run tiil the completion and no preemption is done.

Page No.
Date : / /

Algorithm :-

Start

Step ①:- Make a Structure Process with variables pid, bt priority.

Step ②:- In function bool compare (process a, process b)
return (a.priority > b.priority)

Step ③:- In function waitingtime (process pro [], int n, int wt [])

Set wt[0] = 0
Loop for i = 1 and i < n and i++
Set wt[i] = pro[i-1].bt + wt[i-1]
End

Step ④:- In function turnaround (process pro [], int n, int wt [], int tat [])

Loop for i=0 and i<n and i++
Set tat[i] = pro[i].bt + wt[i]
END LOOP

Step ⑤:- In function avgtime (process pro [], int n)
Declare and initialize wt[n], tat[n],
total_wt = 0, total_tat = 0
Call function waitingtime (pro, n, wt)
Call function turnaround (pro, n, wt, tat)
print "Processes, Burst time, waiting time, turnaround time"

Loop for i=0 and i<n and i++
Set total_wt = total_wt + wt[i]
total_tat = total_tat + tat[i]
End Loop
print values of " Process, burst time, Waiting time, Turn around time "
print Average time, Average Turnaround time.

Step (6) :- In function Scheduling (process pro[], int n)
Call function sort (pro, pro+n, compare)
Loop for i=0 and i<n and i++
print the order
End loop
Call function avgtime (pro, n)

Step (7) :- In function int main ()
Declare and initialize process pro[] = {{1, 10, 2}, {2, 5, 0}, {3, 8, 1}}
Declare and initialize n = sizeof pro / sizeof pro[0]
Call function scheduling (pro, n)

Stop.

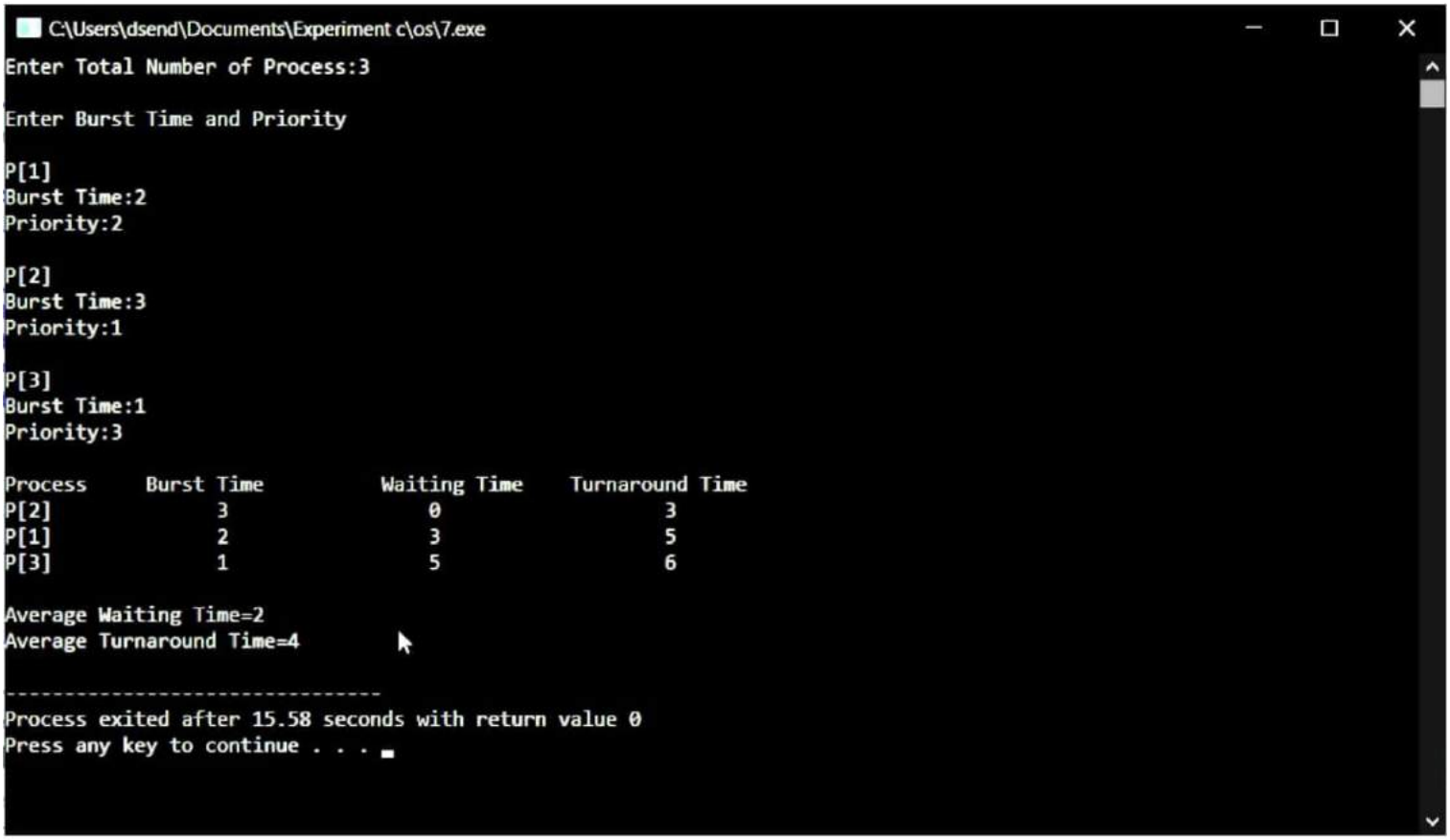
Program :-

```
#include <stdio.h>
int main () {
    int bt[20], p[20], wt[20], tat[20], pr[20], i, j,
    n, total = 0, pos, temp, avg_wt, avg_tat ;
    printf (" Enter Total number of process ");
    scanf ("%d", &n);

    printf ("\n Enter Burst time and priority \n");
    for (i=0; i<n ; i++) {
        printf ("\n P[%d] \n", i+1);
        printf (" Burst time : ");
        scanf ("%d", &bt[i]);
        printf (" Priority : " );
        scanf ("%d", &pr[i]);
        p[i] = i+1;
    }

for (i=0; i<n ; i++ ) {
    pos = i ;
    for (j=i+1; j<n ; j++ ) {
        pos if (pr[j] < pr[pos])
            pos = j ;
    }

    temp = pr[i];
    pr[i] = pr[pos];
    pr[pos] = temp ;
```

```
        temp = bt[i];
        p[i] = p[pos];
        bt[pos] = temp;

        temp = p[i];
        p[i] = p[pos];
        p[pos] = temp;
    }
    wt[0] = 0;

    for(i=1; i<n; i++) {
        wt[i] = 0;
        for(j=0; j<i; j++) {
            wt[i] += bt[j];
        }
        total += wt[i];
    }

    avg_wt = total/n;
    total = 0;

    printf("\n Process\t Burst time \t Waiting time \t
            Turnaround time");
    for(i=0; i<n; i++) {
        tat[i] = bt[i] + wt[i];
        total += tat[i];
        printf("\n P[%d]\t\t %d\t\t %d\t\t\t %d",
            p[i], bt[i], wt[i], tat[i]);
    }
```

```
avg_tat = total / n;
printf ("\n\n Average Waiting Time = %d", avg_wt);
printf ("\n Average Turnaround Time = %d", avg_tat);

return 0;

}
```

Conclusion :- Hence, we successfully developed and executed priority scheduling program.


C:\Users\dsend\Documents\Experiment c\os\7.exe
Enter Total Number of Process:3
Enter Burst Time and Priority
P[1]
Burst Time:2
Priority:2
P[2]
Burst Time:3
Priority:1
P[3]
Burst Time:1
Priority:3
Process Burst Time Waiting Time Turnaround Time
P[2] 3 0 3
P[1] 2 3 5
P[3] 1 5 6
Average Waiting Time=2
Average Turnaround Time=4
Process exited after 15.58 seconds with return value 0
Press any key to continue . . .

Page No.
Date : / /

Practical No :- 08

Aim :- Develope and debug and execute a C program to simulate producer - Consumer problem using Semaphores.

Theory :-

Producer - Consumer problem using Semaphores

The producer consumer problem is a synchronization problem. There is a fixed size buffer and the producer produces items and enter them into the buffer. The consumer removes the items from the buffer and consumes them.

A producer should not produce items into the buffer when the consumer is consuming an item from an buffer and vise-versa. So the buffer should only be accessed by the producer or consumer at a time.

The producer consumer problem can be accessed or resolved using semaphores. The code for the producer and consumer process are as given follows :-

Producer Process :-

The code that defines the producer process is given below :-

```
do {
    .
    .  Produce items
    .
     wait (empty);
     wait (mutex);
    .
    .  Put them in Buffer
    .
     signal (mutex);
     signal (full);
} while (1);
```

In the above code, mutex empty and full are semphores. Here mutex is initialized to 1, Empty is initialized to n and full is initialized to 0.

The mutex semaphores ensures mutual exclusion. The empty and full semaphores count the number of emty and full spaces in the buffer.

Consumer Process :-

The Code that defines the consumer process is given below.

```
do {
    wait (full);
    wait (mutex);
    ..
    .  REMOVE ITEM FROM BUFFER
    .
    signal (mutex);
    signal (empty);
    .
    .  CONSUME ITEM
} while (1);
```

The wait operation is carried out on full. This indicates that items in the buffer have decreased by 1. The wait operation is carried out on mutex so that producer process cannot interfere.

Then the item is removed from buffer. After that signal operation is carried out on mutex and empty. The former indicates that consumer process can now act and the latter shows that the empty space in the buffer increased by 1.

Program :-

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>

int mutex = 1, full = 0, empty = 3, x = 0;
int main () {
    int n;
    void producer ();
    void consumer ();
    int wait (int);
    int signal (int);
    printf ("\n1. Produci\n2. consumer\n3. Exit");
while (1) {
    printf ("\n Enter your choice:");
    scanf ("%d", &n);
    switch (n) {
        case 1 : if ((mutex == 1) && (empty != 0))
                    producer ();
                else
                    printf ("Buffer is full");
                    break;

        case 2 : if ((mutex == 1) && (full != 0))
                    consumer ();
                else
                    printf ("Buffer is Empty");
                break;
```

```
        Case 3:
                exit (0);
                break;
        }
}

    return 0;
}

int wait (int s) {
        return (--s);
    }

int signal (int s) {
        return (++s)
    }

Void producer () {
        mutex = wait (mutex);
        full = signal (full);
        empty = wait (empty);
        x++;
        printf ("\n Producer produces the item %d", x);
        mutex = signal (mutex);
}
```

```
void consumer () {
    mutex = wait (mutex);
    full = wait (full);
    Empty = signal (empty);
    printf("\n Consumer Consumes item %d", x);
    x--;
    mutex = signal (mutex);
}
```

Conclusion :- Hence, we succesfully developed and executed the consumer-producer problem using semaphores in C.


C:\Users\dsend\Documents\Experiment c\os\8.exe
1.Producer
2.Consumer
3.Exit
Enter your choice:1
Producer produces the item 1
Enter your choice:2
Consumer consumes item 1
Enter your choice:
1
Producer produces the item 1
Enter your choice:1
Producer produces the item 2
Enter your choice:1
Producer produces the item 3
Enter your choice:1
Buffer is full!!
Enter your choice:2
Consumer consumes item 3
Enter your choice:2
Consumer consumes item 2
Enter your choice:2
Consumer consumes item 1
Enter your choice:2
Buffer is empty!!
Enter your choice:3
Process exited after 2265 seconds with return value 0
Press any key to continue . . .

Page No.
Date : / /

Practical :- 09

Aim :- Develope, debug and execute C program to simulate FIFO page replacement algorithm.

Theory :-

First In First Out page replacement algorithm :-

This is the simplest page replacement algorithm, operating system keeps tracks of all pages in the memory in a queue, oldest pages is in the front of the queue. When a page needs to be replaced page is the front of the queue is selected for removal.

Example 1 :- Conside page reference string 1, 3, 0, 3, 5, 6 and 3 page slots.

Initially all slots are empty, so when 1, 3, 0 came they are allocated to the empty slots → 3 page faults

When 3 comes, it is already in a memory so → 0 Page faults.

Then 5 comes, it is not available in memory so it replaces the oldest page slots i.e. 1 → 1 page fault.

finally 6 comes, it is also not available in memory so it replaces the oldest page slot i.e. 3 → 1 page fault.

Algorithm :-

1 - Start traversing the pages.

i) If set holds less pages than capacity

a) Insert page into the set one by one until the size of set reaches capacity of all page requests are processed.

b) Simultaneously maintain the pages in the queue to perform FIFO.

c) Increment page fault.

ii) Else

If current page is present in set, do nothing Else

a) Remove the first page from the queue as it was the first to be entered in the memory.

b) Replace the first page in the queue with the current page in the string.

c) Store current page in a queue.

d) Increment page faults.

2) Return page faults.

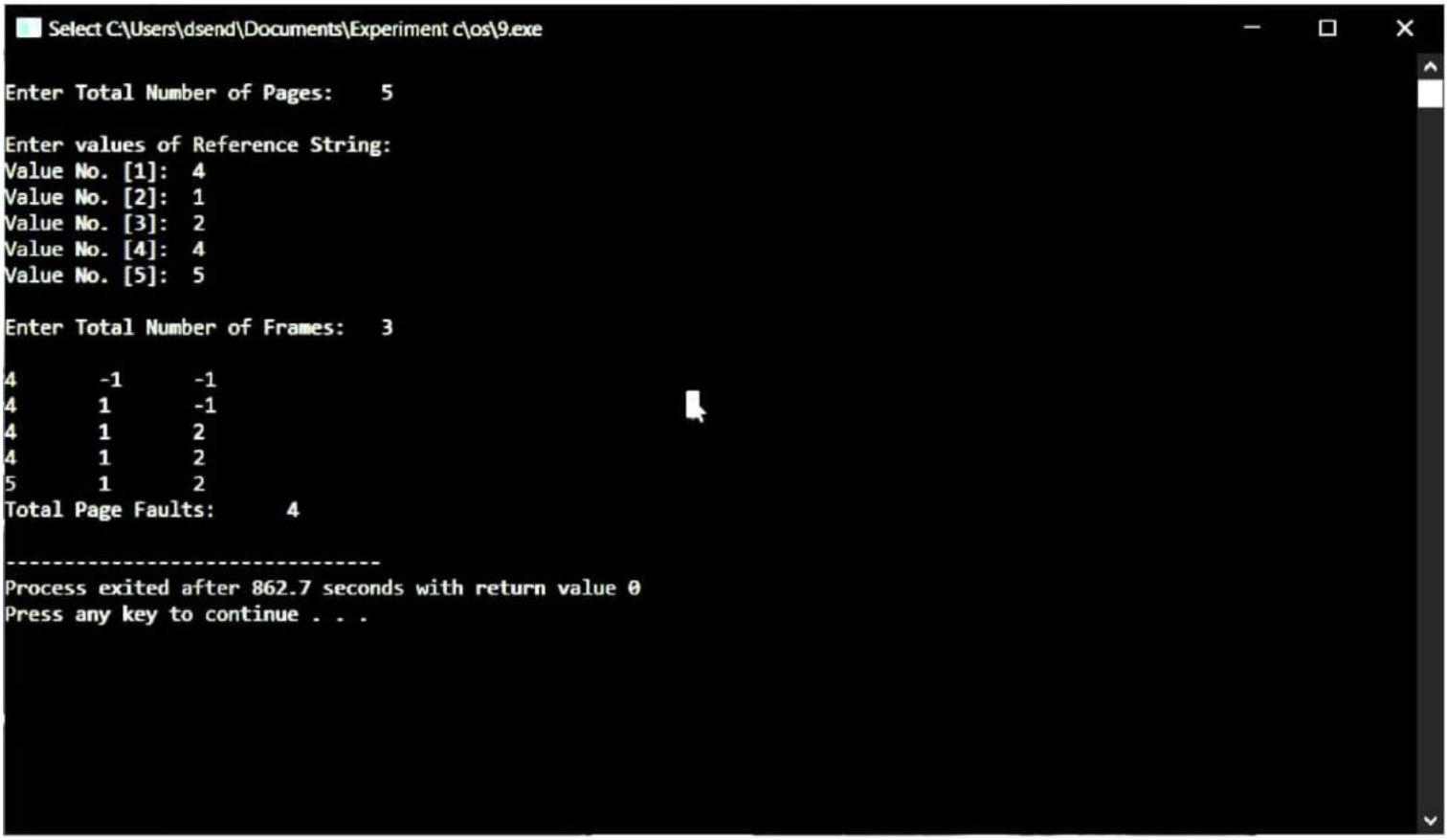
Page No.
Date: / /

Program :-

```
#include <stdio.h>
Int main () {
    Int reference_string [10], page_faults = 0, m, n, s,
    pages, frames ;
printf ("\n Enter Total number of Pages : \t ");
Scanf ("%d", & pages);
printf ("\n Enter Values of Reference string : \n");
for (m=0 ; m<pages ; m++) {
    printf (" Values of No. [%d] : \t ", m+1);
    Scanf ("%d", & reference_string [m];
}
printf ("\n Enter total Number of frames : \t ");
{
    scanf ("%d", & frames);
}

int temp [frames];
for (m=0; m< frames ; m++) {
    temp [m] = -1;
}

for (m=0; m<pages ; m++) {
    s = 0;
    for (n=0 ; n< frames ; n++) {
        if (reference_string[m] == temp [n]) {
            s++;
            page_faults -- ;
        }
    }
}
```

```
        page_faults ++;
        if ((page_faults <= frames) && (s==0)) {
            temp [m] = reference_string [m];
        }

        else if (s==0) {
            temp [(page_faults - 1) % frames] = reference_string
                                                [m]
        }
        printf ("\n");
        for (n=0; n< frames ; n++) {
            printf ("%d\t", temp [n]);
        }

    }
    printf ("\n Total page faults : \t%d\n", page_faults);
    return 0;

}
```

Conclusion :- Hence, we successfully developed and executed FIFO page replacement algorithm.


Select C:\Users\dsend\Documents\Experiment c\os\9.exe
Enter Total Number of Pages: 5
Enter values of Reference String:
Value No. [1]: 4
Value No. [2]: 1
Value No. [3]: 2
Value No. [4]: 4
Value No. [5]: 5
Enter Total Number of Frames: 3
4 -1 -1
4 1 -1
4 1 2
4 1 2
5 1 2
Total Page Faults: 4
Process exited after 862.7 seconds with return value 0
Press any key to continue . . .

Practical No :- 10

Aim :- Develope, debug and execute a C program to simulate LRU Page replacement algorithm.

Theory :-

LRU Page replacement algorithm :-

The page replacement algorithm help an operating system in deciding the memory pages that needs to be swapped out, written to the disk when a page of memory needs to be allocated in the system.

The LRU page replacement method is a marking algorithm. It keeps a track of the page usage in a given period of time. The LRU algorithm effort.

The LRU algorithm offers optimum performance but is the costly in its implementation. The LRU page replacement technique is modified for implementation and its successor are.

The LRU Page replacement technique is modified for implementation, and its successor are LRU-K and ARC algorithm.

Page No.
Date : / /

Algorithm :-

1- Start traversing pages.
i) If Sets hold less pages than capacity.
a) Insert page into the set one by one until the size of set reaches capacity or all.
b) Simultaneously maintain the recent occured index of each page in a map called indexes.
c) Increment page fault.

ii) Else
If Current page is present in set, do nothing
Else
a) find the page in the set that was least recently used. We find it using index array. We basically need to replace the page with minimum index.
b) Replace the found page with current page.
c) Increment page faults.
d) upade index of current page

2. Return page faults.

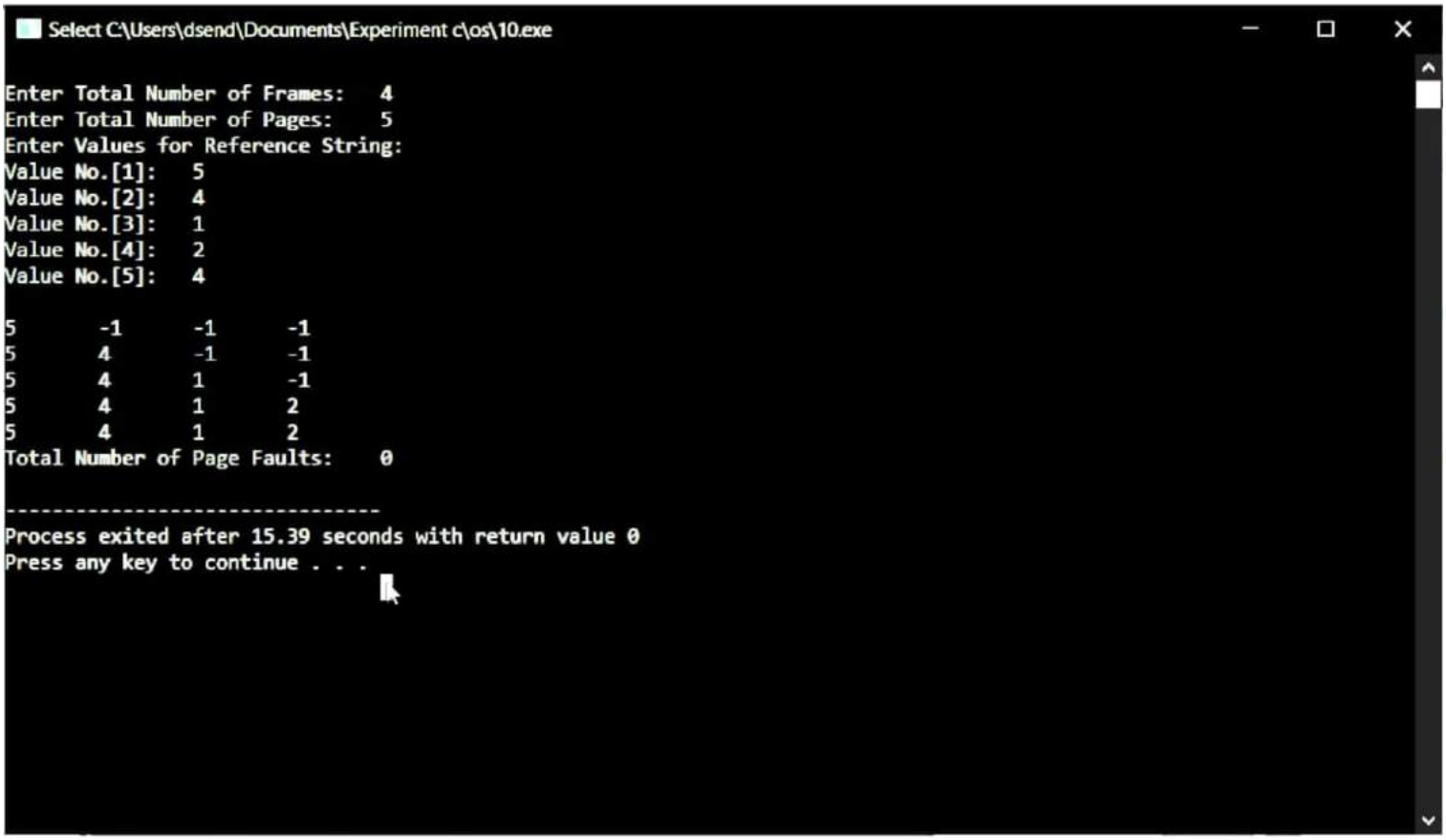
Program :-

```
#include <stdio.h>
int main () {
    int frames [10], temp [10], pages [10];
    int total_pages, m, n, position, k, l,
        total_frames;
    int a=0, b=0, page_fault = 0;
    printf ("\n Enter Total Number of Frames: \t");
    scanf ("%d", &total_frames);

    for (m=0; m< total_frames; m++) {
        frames [m] = -1;
    }

    printf ("Enter Total Number of Pages : \t");
    scanf ("%d", & total_pages);
    printf (" Enter Values for Refrence String : \n");

    for (m=0; m< total_pages; m++) {
        printf ("Values No. [%d] : \t", m+1);
        scanf ("%d", &pages [m]);
    }

    for (n=0; n< total_pages; n++) {
        a=b, b=0;
        for (m=0; m < total_frames; m++) {
            if (frames [m] == pages [n])
            {   a= 1;
                b=1;
                break;
```

```
if (a==0) {
    for (m=0; m<total_frames; m++) {
        if (frames[m] == -1) {
            frames[m] = pages[n];
            b = 1;
            break;
        }
    }
}
if (b==0) {
    for (m=0; m<total_frames; m++) {
        temp[m] = 0;
    }
    for (k=n-1, l=1; l<=total_frames-1; l++, k--) {
        for (m=0; m<total_frames; m++) {
            if (frames[m] == pages[k]) {
                temp[m] = 1;
            }
        }
    }
}

for (m=0; m<total_frames; m++) {
    if (temp[m] == 0)
        position = m;
}
frames[position] = pages[n];
page_fault++;
}
```

```
        printf ("\n");
    for (m=0; m< total_frames ; m++) {
        printf (" %d \t", frames[m]);
    }
}

printf ("\nTotal Number of Page faults : \t %d \n",
        page_fault);

}
```

Conclusion :- Hence, we successfully developed and executed LRU page replacement algorithm.


Select C:\Users\dsend\Documents\Experiment c\os\10.exe
Enter Total Number of Frames: 4
Enter Total Number of Pages: 5
Enter Values for Reference String:
Value No.[1]: 5
Value No.[2]: 4
Value No.[3]: 1
Value No.[4]: 2
Value No.[5]: 4
5 -1 -1 -1
5 4 -1 -1
5 4 1 -1
5 4 1 2
5 4 1 2
Total Number of Page Faults: 0
Process exited after 15.39 seconds with return value 0
Press any key to continue . . .

Page No.
Date :

Practical No :- 11

Aim:- Develope and execute a C program to simulate optimal replacement algorithm.

Theory :-

Optimal Page Replacement Algorithm :-

The page replacement algorithms are used in operating systems that use virtual memory management.

When a page of memory needs to be allocated to the CPU, these page replacement algorithm decide which pages should be written to the disk and which algorithms should be swapped out of memory.

The algorithm is also known as Clairvoyent Replacement Algorithm. As per the optimal page replacement technique, the page with the highest label should be removed first.

When a page needs to be swapped into the memory, the OS will swap out the page which is not required to be used in the near future.

This page replacement algorithm is a little unreliable to implement and therefore it cannot be implemented in a general purpose operating System.

Algorithm :-

① If referred page is already present, increment hit count.

② If not present, if a page that is never referenced in future. If such a page exists, replace this page with new page. If no such page exists, find a page that is referenced farthest in future. Replace this page with new page.

Page No.
Date: / /

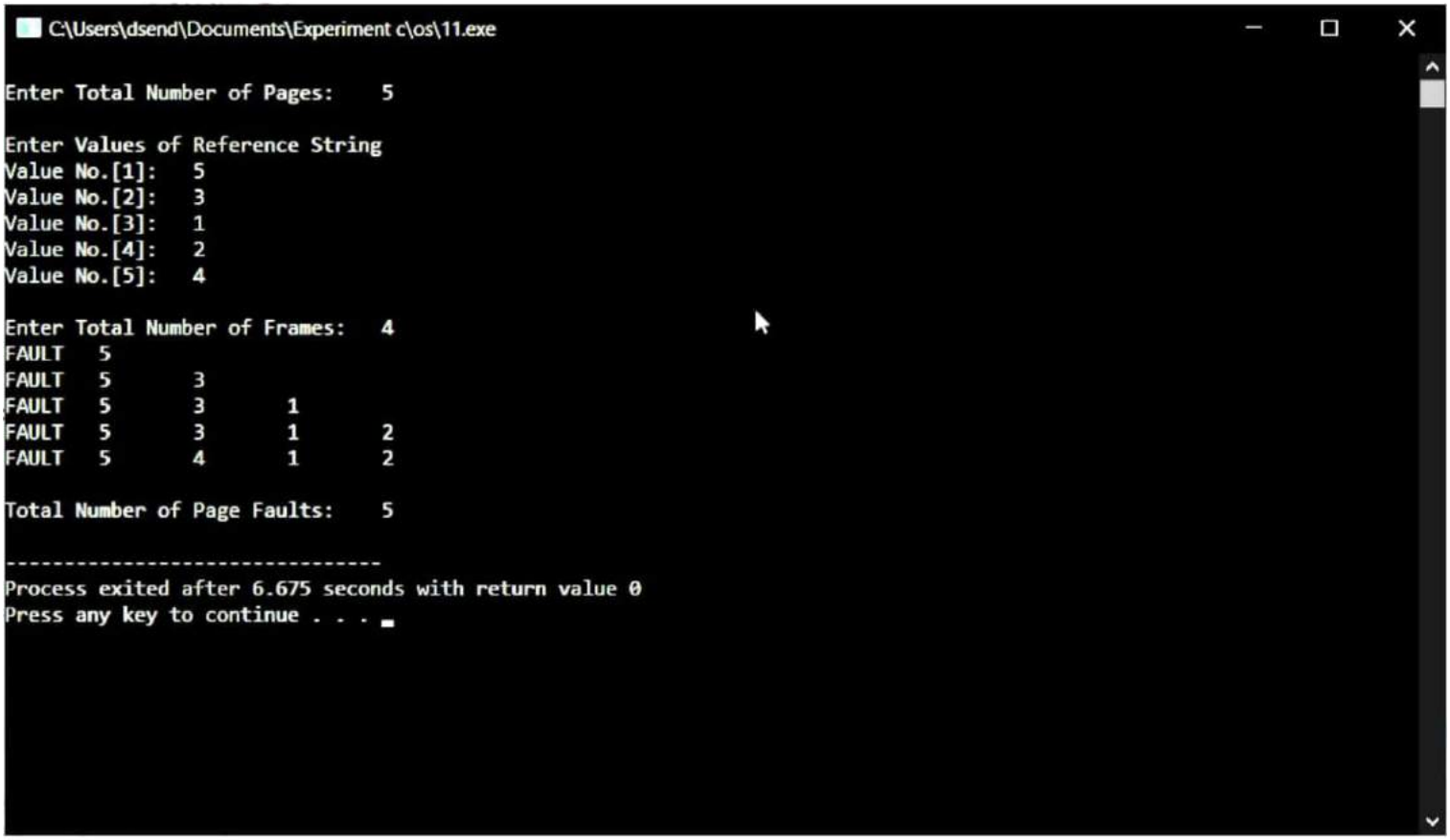
Program :-

```
#include <stdio.h>
int main () {
    int reference_string [25], frames [25], interval[
    int pages, total_frames, page_faults = 0;
    int m, n, temp, flag, found;
    int position, maximum_interval, previous_frame =
    printf ("\nEnter Total Number of Pages: \t");
    scanf ("%d", &pages);
    printf ("\n Enter Value of reference string \n");
    for (m=0; m<pages; m++) {
        printf (" Value No. [%d]: \t", m+1);
        scanf ("%d", &reference_string [m]);
    }

    printf ("\n Enter Total Number of frames: \t");
    scanf ("%d", &total_frames);

    for (m=0; m<total_frames; m++) {
        frames [m] = -1;
    }
    for (m=0; m<pages; m++) {
        flag = 0;
        for (n=0; n<total_frames; n++) {
            if (frames [n] == reference_string [m]) {
                flag = 1;
                printf ("\t");
                break;
            }
    }
```

```
if (flag == 0) {
 if (previous_frame == total_frame - 1) {
  for (n=0; n < total_frames; n++) {
  for (temp = m+1; temp < pages; temp++) {
   interval [n] = 0;
  if (frames [n] == reference_string [temp])
  {
     interval [n] = temp - m;
     break;
  }
  }
 }
}
found = 0;
for (n = 0; n < total_frames ; m++) {
   if (interval [n] == 0) {
     position = n;
     found = 1;
     break;
   }
  }
 }
else {
   position = ++previous_frame;
     found = 1;
 }
if (found == 0) {
   maximum_interval = interval [0];
   position = 0;
```

```
            for (n=1; n < total_frames ; n++ ){
              if (maximum_interval < interval [n]){
                maximum_interval = interval [n];
                position = n;
              }
            }
          }

          frames [position] = reference_string [m];
          printf ("FAULT\t");
          page_faults++;
        }
        for (n=0 ; n < total_frames ; n++){
            if (frames [n] != -1)
              {
                printf ("%d\t", frames [n]);
              }
            printf ("\n");
        }
        printf ("\n Total Number of page Fault : \t:%d\n",
                page_faults);
}
```

Conclusion :- Hence, we successfully developed and executed optimal page replacement algorithm


C:\Users\dsend\Documents\Experiment c\os\11.exe
Enter Total Number of Pages: 5
Enter Values of Reference String
Value No.[1]: 5
Value No.[2]: 3
Value No.[3]: 1
Value No.[4]: 2
Value No.[5]: 4
Enter Total Number of Frames: 4
FAULT 5
FAULT 5 3
FAULT 5 3 1
FAULT 5 3 1 2
FAULT 5 4 1 2
Total Number of Page Faults: 5
Process exited after 6.675 seconds with return value 0
Press any key to continue . . .

Page No.
Date : / /

Practical No :- 12

Aim :- Develope, debug and execute C program to simulate LFU page replacement algorithms.

Theory :-

LFU Page replacement algorithms :-

In LFU page Replacement method, the page with the minimum count is selected for replacement with the page that needs to enter into the system.

LFU is a cache algorithm which is used to mange computers memory. A counter is assigned to every block of memory that is loaded into the cache memory.

However, the LFU technique is hardly implemented these days but this algorithm is normally combined with other algorithm which make it a hybrid algorithm, then it is implemented.

LFU algorithm is sometimes also combined with LRU algorithm, and them implemented.

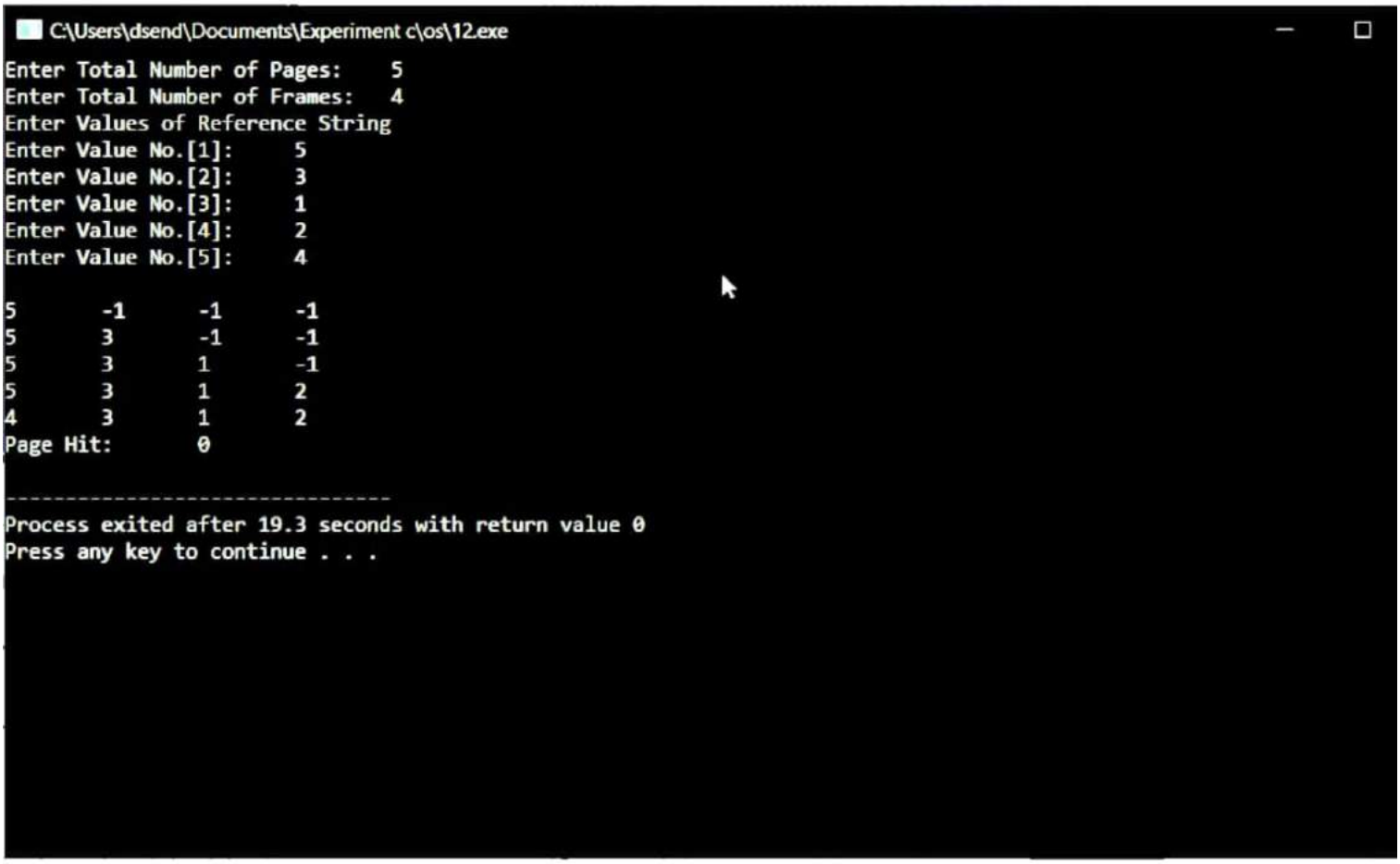
Program :-

```
#include <stdio.h>
int main () {

    int total_frames , total_pages , hit = 0;
    int pages [25] , frame [10] , arr [25] , time [25];
    int m, n , page flag , k , minimum_time , temp ;

    printf (" Enter total number of pages : \t ");
    scanf (" %d ", &total_pages);
    printf (" Enter total number of frame : \t ");
    scanf (" %d ", &total_frames );

    for (m=0; m<total_frames ; m++) {
            frame [m] = -1;
    }
    for ( m=0; m<25 ; m++) {
        arr [m] =0;
    }

    printf (" Enter values of reference String \n ");
    for (m =0 ; m< total_pages ; m++) {
            printf (" Enter value No. [%d] : \t ", m+1);
            scanf (" %d ", &pages [m] );
    }
    printf ("\n");
```

```
for (m=0; m<total_pages; m++) {
    arr[pages[m]]++;
    time[pages[m]] = m;
    flag = 1;
    k = frame[0];
    for (n=0; n<total_frames; n++) {
        if (frame[n]==-1 || frame[n] == pages[m]) {
            if (frame[n] != -1) {
                hit ++;
            }
            flag = 0;
            frame[n] = pages[m];
            break;
        }
        if (arr[k] > arr[frame[n]])
            { k = frame[n]; }
    }
    if (flag) {
        minimum_time = 25;
        for (n=0; n<total_frame; n++) {
            if (arr[frame[n]] == arr[k] && time[
                frame[n]] < minimum_time) {
                temp = n;
                minimum_time = time[frame[n]];
            }
        arr[frame[temp]] = 0;
        frame[temp] = pages[m];
    }
```

```
        for (n=0; n<total_frames; n++) {
            printf ("%d\t", frame[n]);
        }
        printf ("\n");
    }
    printf ("page hit : \t%d\n", hit);
    return 0;

}
```

Conclusion: Hence, we successfully developed and executed the LFU Page replacement algorithm.


C:\Users\dsend\Documents\Experiment c\os\12.exe
Enter Total Number of Pages: 5
Enter Total Number of Frames: 4
Enter Values of Reference String
Enter Value No.[1]: 5
Enter Value No.[2]: 3
Enter Value No.[3]: 1
Enter Value No.[4]: 2
Enter Value No.[5]: 4
5 -1 -1 -1
5 3 -1 -1
5 3 1 -1
5 3 1 2
4 3 1 2
Page Hit: 0
Process exited after 19.3 seconds with return value 0
Press any key to continue . . .

Page No.
Date: / /

Practical No:- 13

Aim :- Develope, debug and execute a C program to simulate the following contiguous memory allocation techniques.
a) Worst-fit b) Best-fit c) first-fit.

Theory :-

Contiguous Memory Allocation :-

The main memory must accommodate both the operating system and the various user processes. We therefore need to allocate main memory in the most efficient way possible. This section explains once early method, contiguous memory allocation.

First-Fit :- Allocate the first hole that is big enough. Searching can start either at the beginning of the set of holes or at the location where the previous first-fit search ended. We can stop searching as soon as we find a free hole that is large enough.

Best fit :- Allocate the smallest hole that is big enough. We must search the entire list, unless the list is ordered by size. This strategy produces the smallest leftover hole.

Page No.
Date :

Worst fit :- Allocate the largest hole. Again, we must search the entire list, unless it is sorted by size. This strategy produces the largest leftover hole, which may be more useful than the smaller leftover hole from a best-fit approach.

Program :-

a) ~~First~~ ~~Worst~~ - fit :-

```
#include<stdio.h>
#include<conio.h>
#define max 25
Void main (){

int frag[max], b[max], f[max], i, j, nb, nf, temp;
static int bf[max], ff[max];

printf("\n\tMemory Managem Scheme - First Fit");
printf("\n Enter the number of blocks : ");
scanf("%d", &nb);
printf(" Enter the number of files");
scanf("%d", &nf);
printf("\n Enter the size of the blocks :- \n");

for(i= 1; i<=nb ; i++ {
    printf(" Block %d : ", i);
    scanf("%d", &b[i]);
}
```

```
printf ("Enter the size of files :- \n");
for (i=1 ; i<nf ; i++) {
    printf ("File %d : ", i);
    scanf ("%d", &f[i]);
}

for (i=1 ; i<=nf ; i++) {
    for (j=1; j<=nb; j++) {
        if (bf[i] != 1) {
            temp = b[j] - f[i];
            if (temp >= 0) {
                ff[i] = j;
                break;
            }
        }
    }
    frag[i] = temp;
    bf[ff[i]] = 1;
}
printf ("\n File_no: \t File_Size \tBlock_no : \t Block_size:
        \t Fragement ");
for (i=1 ; i<=nf ; i++) {
    printf ("\n%d\t\t%d\t\t%d\t\t%d\t\t%d", i,
        f[i], ff[i], b[ff[i]], frag[i]);

}
```

b) Best - fit :-

```
#include <stdio.h>
#include <conio.h>
# define 25

Void main (){
Int frag [max], b [max], f [max], i, j, nb, nf,
temp, lowest = 10000;
Static int bf [max], ff [max];
printf ("\n Enter the number of blocks: ");
scanf ("%d", &nb);
printf (" Enter the number of files:" );
scanf ("%d", &nf);
printf ("\n Enter the size of the blocks :-\n" );

for (i=1; i<=nb ; i++) {
      printf (" Block %d : ", i);
      scanf ("%d", &b[i] );
 }
printf (" Enter the size of the files :- \n");
for (i=1 ; i<=nf ; i++) {
      printf (" File %d : ", i );
      scanf ("%d", &f [i]);
}
```

```
for (i=1; i<=nf; i++) {
    for (j=1; j<=nb; j++) {
        if (bf[j] != 1) {
            temp = b[j] - f[i];
            if (temp >= 0)
                if (lowest > temp) {
                    ff[i] = j;
                    lowest = temp;
                }
        }
    }

frag[i] = lowest;
bf[ff[i]] = 1;
lowest = 10000;
}
printf("\n File No \t File Size \t Block No \t Block size
        \t Fragment");
for (i=1; i<=nf && ff[i] != 0; i++) {
    printf("\n%d\t\t%d\t\t%d\t\t%d\t\t%d",
        i, f[i], ff[i], b[ff[i]], frag[i]);
}
```

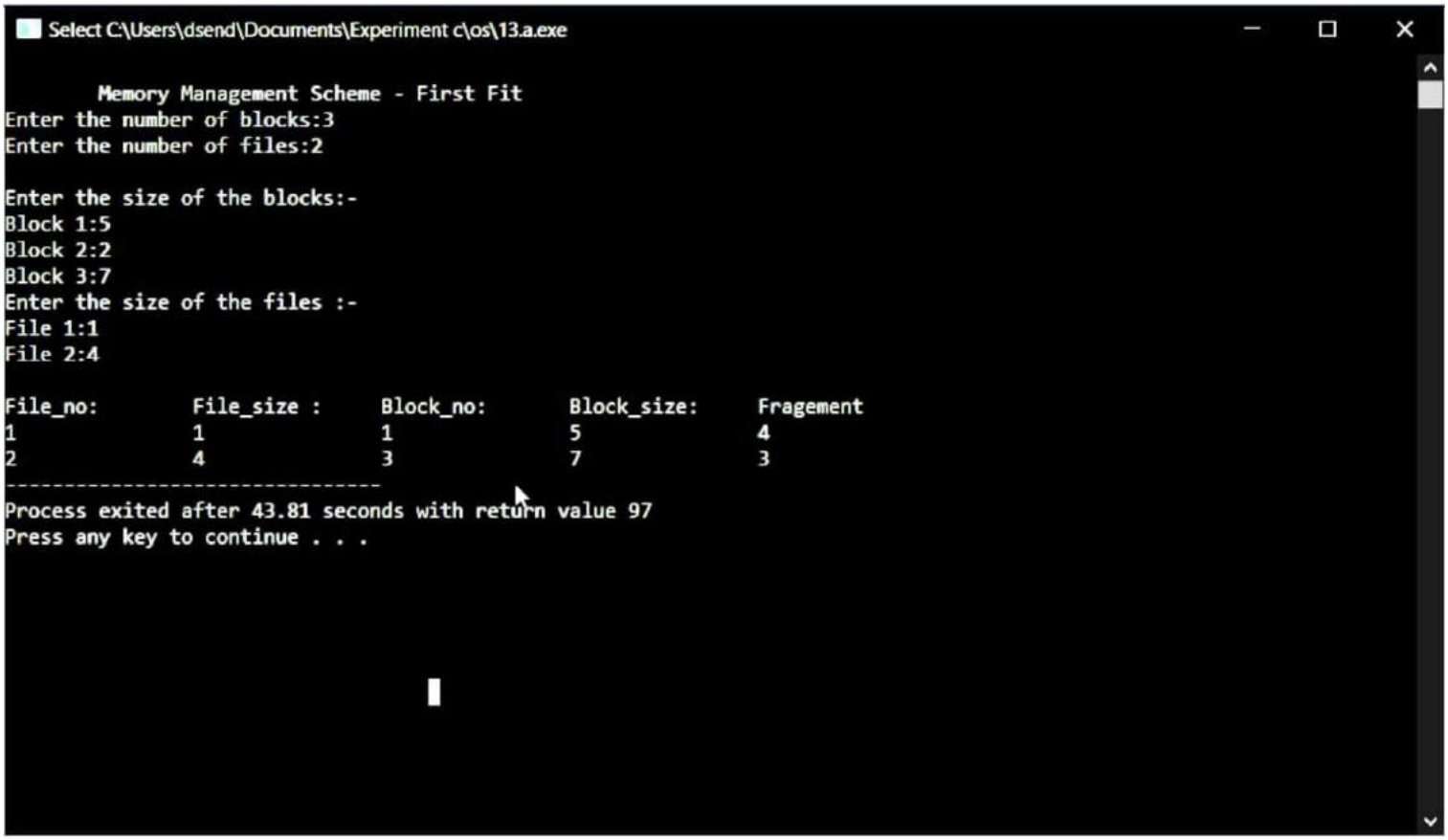
Worst

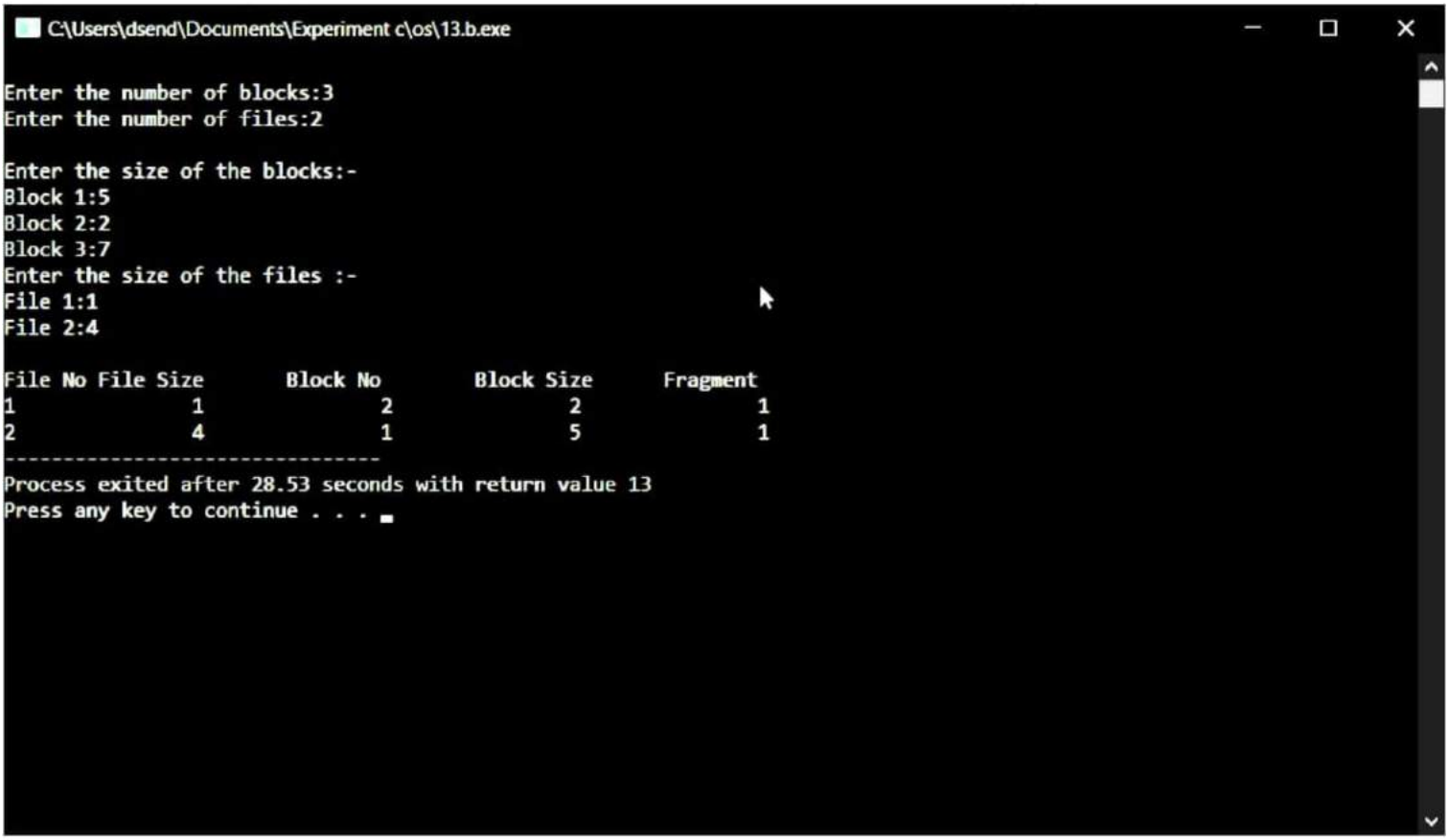
c) ~~First~~ - Fit :-

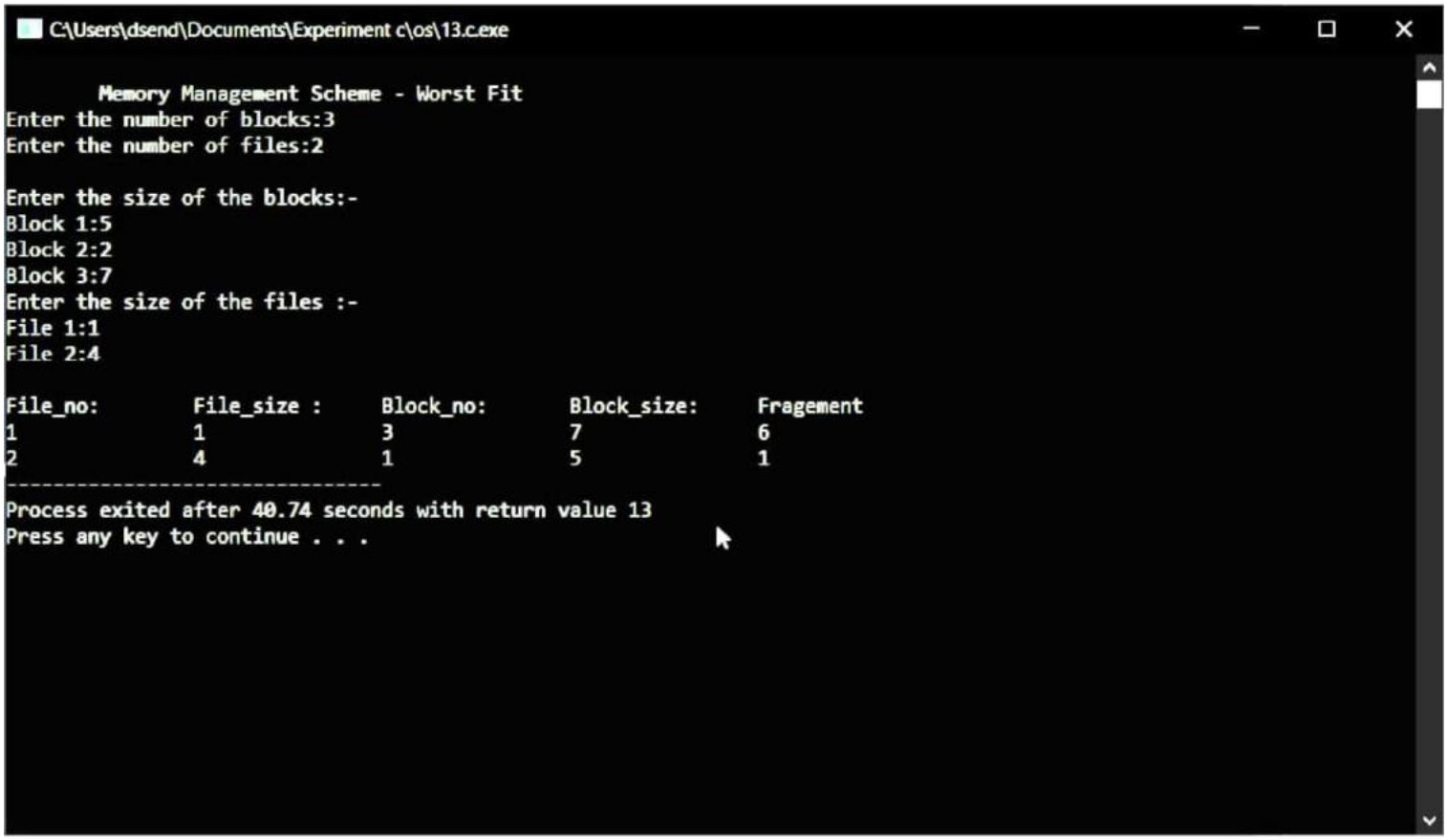
```
#include <stdio.h>
#include <conio.h>
# define max 25
void main () {

int frag [max], b [max], f [max], i, j, nb, nf, temp,
      highest = 0;
static int bf [max], ff [max];
printf ("\n\t Memory Management Scheme - Worst Fit");
printf ("\n Enter the number of blocks :");
scanf ("%d", &nb);
printf ("Enter the number of files :");
scanf ("%d", &nf);
printf ("\n Enter the size of the blocks :- \n");
for (i = 1 ; i <= nb ; i++ ) {
      printf (" Block %d :", i);
      scanf ("%d", &b [i]);
}

printf ("Enter the size of the files :- \n");
for (i = 1 ; i <= nf ; i++ ) {
      printf ("File %d :", i);
      scanf ("%d", &f [i]);
}
```

```
for (i=1 ; i<=nf ; i++) {
    for (j=1 ; j<=nb ; j++) {
        if (bf[j] != 1) {
            temp = b[j] - f[i];
            if (temp >= 0)
            if (highest < temp) {
                ff[i] = j;
                highest = temp;
            }
        }
    }
frag[i] = highest;
bf[ff[i]] = 1;
highest = 0;
}
printf ("\n File_no : \t File_size : \t Block_no : \t Block_size:
        \t Fragement ");
for (i=1 ; i<=nf ; i++) {
    printf ("\n%d\t\t%d\t\t%d\t\t%d\t\t%d", i, f[i]
    ff[i], b[ff[i]], frag[i]);

}
```

Conclusion :- Hence, we successfully developed and executed contiguous memory allocation technique.


Select C:\Users\dsend\Documents\Experiment c\os\13.a.exe
Memory Management Scheme - First Fit
Enter the number of blocks:3
Enter the number of files:2
Enter the size of the blocks:-
Block 1:5
Block 2:2
Block 3:7
Enter the size of the files :-
File 1:1
File 2:4
File_no: File_size : Block_no: Block_size: Fragement
1 1 1 5 4
2 4 3 7 3
Process exited after 43.81 seconds with return value 97
Press any key to continue . . .


C:\Users\dsend\Documents\Experiment c\os\13.b.exe
Enter the number of blocks:3
Enter the number of files:2
Enter the size of the blocks:-
Block 1:5
Block 2:2
Block 3:7
Enter the size of the files :-
File 1:1
File 2:4
File No File Size Block No Block Size Fragment
1 1 2 2 1
2 4 1 5 1
Process exited after 28.53 seconds with return value 13
Press any key to continue . . .


C:\Users\dsend\Documents\Experiment c\os\13.c.exe
Memory Management Scheme - Worst Fit
Enter the number of blocks:3
Enter the number of files:2
Enter the size of the blocks:-
Block 1:5
Block 2:2
Block 3:7
Enter the size of the files :-
File 1:1
File 2:4
File_no: File_size : Block_no: Block_size: Fragement
1 1 3 7 6
2 4 1 5 1
Process exited after 40.74 seconds with return value 13
Press any key to continue . . .