**Batch:** A3 **Roll No.:** 1411039

**Experiment / assignment / tutorial No. 1**

**Grade: AA / AB / BB / BC / CC / CD /DD**

**Signature of the Staff In-charge with date**

|  |
| --- |
| **TITLE:** Commands for handling File system. |

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**AIM:** To Explore basic commands for handling File system under Unix/Linux using shell scripts.(Creating groups, chown , chmod , directory name, tty , diff, umask).

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Expected Outcome of Experiment:**

**CO 1.** Explain the fundamental concepts of operating system with extension to Unix and Mobile OS.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Books/ Journals/ Websites referred:**

1. **Silberschatz A., Galvin P., Gagne G. “Operating Systems Principles”, Willey Eight edition.**
2. **Achyut S. Godbole , Atul Kahate “Operating Systems”, McGraw Hill Third Edition.**
3. **Sumitabha Das “ UNIX Concepts & Applications”, McGraw Hill Second**

**Edition.**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Pre Lab/ Prior Concepts:**

An operating system (OS) is a resource manager. It takes the form of a set of software routines that allow users and application programs to access system resources (e.g. the CPU, memory, disks, modems, printers network cards etc.) in safe efficient and abstract way.

* The operating system kernel is in direct control of the underlying hardware. The kernel provides low-level device, memory and processor management functions (e.g. dealing with interrupts from hardware devices, sharing the processor among multiple programs, allocating memory for programs etc.)
* Basic hardware-independent kernel services are exposed to higher-level programs through a library of system calls (e.g. services to create a file, begin execution of a program, or open a logical network connection to another computer).
* Application programs (e.g. word processors, spreadsheets) and system utility programs (simple but useful application programs that come with the operating system, e.g. programs which find text inside a group of files) make use of system calls. Applications and system utilities are launched using a shell (a textual command line interface) or a graphical user interface that provides direct user interaction.

Operating systems can be distinguished from one another by the system calls, system utilities and user interface they provide, as well as by the resource scheduling policies implemented by the kernel.

UNIX has been a popular OS for more than two decades because of its multi-user, multi-tasking environment, stability, portability and powerful networking capabilities.

Linux is a free open source UNIX OS for PCs.

Linux has all of the components of a typical OS :

* **Kernel**

The Linux kernel includes device driver support for a large number of PC hardware devices (graphics cards, network cards, hard disks etc.), advanced processor and memory management features, and support for many different types of file systems. In terms of the services that it provides to application programs and system utilities, the kernel implements most BSD and SYSV system calls, as well as the system calls described in the POSIX.1 specification.

The kernel (in raw binary form that is loaded directly into memory at system startup time) is typically found in the file /boot/vmlinuz, while the source files can usually be found in /usr/src/linux.

* **Shells and GUIs**

Linux supports two forms of command input: through textual command line shells similar to those found on most UNIX systems (e.g. sh - the Bourne shell, bash - the Bourne again shell and csh - the C shell) and through graphical interfaces (GUIs) such as the KDE and GNOME window managers.

* **System Utilities**

Virtually every system utility that you would expect to find on standard implementations of UNIX has been ported to Linux. This includes commands such as ls, cp, grep, awk, sed, bc, wc, more, and so on. These system utilities are designed to be powerful tools that do a single task extremely well (e.g. grep finds text inside files while wc counts the number of words, lines and bytes inside a file). Users can often solve problems by interconnecting these tools instead of writing a large monolithic application program.

* **Application programs**

Linux distributions typically come with several useful application programs as standard. Examples include the emacseditor, xv (an image viewer), gcc (a C compiler), g++ (a C++ compiler), xfig (a drawing package), latex (a powerful typesetting language) and soffice (StarOffice, which is an MS-Office style clone that can read and write Word, Excel and PowerPoint files).

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Description of the application to be implemented**:

**pwd**

Syntax: pwd

Description: Displays the absolute path of the current directory.

**cd**

Syntax: cd [directory]

Description: The current working directory to the directory specified by "directory".

Example: enter the directory / usr / bin /:

cd / usr / bin

**cd ..** .

Syntax: cd ..

Description: Change to parent directory.

**ls**

Syntax: ls [options] [pathname-list]

Description: display the file name within the directory and file name specified in the

"pathname-list"

Example: List all names in the current working directory is s at the beginning of the file:

ls s \*

**chmod 755 file**

Syntax: chmod 755 [filename]

Description: When you perform the chmod 755 filename command you allow everyone

to read and execute the file, and the file owner is allowed to write to the file as well.

You may need this for Perl and other scripts that should to be run via a webserver. If

you apply 755 to a directory, it means that everyone can go to it and get its file listing.

**chgrp**

Syntax: chgrp [Options]... {Group | --reference=File} File...

Description: 'chgrp' command changes the group ownership of each given File to Group (which can be either a group name or a numeric group id) or to match the same group as an existing reference file.

**chown**

chown [OPTION]... [OWNER][:[GROUP]] FILE…

Description: chown changes the user and/or group ownership of each given file. If only an owner (a user name or numeric user ID) is given, that user is made the owner of each given file, and the files' group is not changed.

**cp file1 file2**

Syntax: cp [options] file1 file2

Description: Copy the file file1 to file2.

Common options:-r copy the entire directory

Example: aaa copy (existing), and named bbb:

cp aaa bbb

**mv file1 newname**

Syntax: mv [options] source ... directory

Description: Rename the file, or the number of files to another directory.

Example: aaa renamed as bbb:

mv aaa bbb

**rm file1 [file2 ...]**

Syntax: rm [options] name ...

Description: delete files and directories.

Commonly used options:-f to force delete files

Example: Remove all but the suffix named c file

rm \*. c

**mkdir dir1 [dir2...]**

Syntax: mkdir [options] dirName

Description: create name in dirName subdirectory.

Example: In the working directory, create a subdirectory named AA:

mkdir AA

**mkdir -p dirpath**

Syntax: mkdir [options] dirName

Description: Create the DIRECTORY(ies), if they do not already exist.

Option -p stands for ‘parent’; no error if existing, make parent directories as needed

**rmdir dir1 [dir2...]**

Syntax: rmdir [-p] directory ...

Description: The rmdir utility removes the directory entry specified by each directory argument, provided the directory is empty.

**cat filename**

Syntax: cat [options] [file-list]

Description: standard output connection, display a list of files in the file-list file

Example 1: Displays the contents of file1 and file2

cat file1 file2

Example 2: file1 and file2 merged into file3

cat file1 file2> file3

**more filename**

Syntax: more [-dlfpcsu] [-num lines] [+/pattern] [+linenum] [file ...]

Description: more is a filter for paging through text one screen at a time. It does not provide as many options or enhancements as less, but is nevertheless quite useful and simple to use.

**less filename**

Syntax: less [Options] [filename]...

Description: less is a simple, feature-rich command-line file viewer.

less is a program similar to more, but it has many more features. less does not have to read the entire input file before starting, so with large input files it starts up faster.

**head filename**

Syntax: head [options] [file-list]

Description:

Description: Display the initial part of the file in the list of files in the file-list, the default display, 10 lines;

Example: the initial part of the file AAA

head AAA

**head -n filename**

Syntax: head -n … [FILE]...

Description: Display the initial part of the file in the list of files in the file-list, the default display, n lines;

**tail filename**

Syntax: tail [OPTION]... [FILE]...

Description: tail prints the last 10 lines of each FILE to standard output. With more than one FILE, it precedes each set of output with a header giving the file name. If no FILE is specified, or if FILE is specified as a dash ("-"), tail reads from standard input.

**tail -n filename**

Syntax: tail -n … [FILE]...

Description: tail -n will output the last n lines instead of the default 10 lines of the file.

**groupadd**

Syntax: groupadd [options] group

Description: The groupadd command creates a new group account using the values specified on the command line plus the default values from the system. The new group will be entered into the system files as needed.

**usersadd**

Syntax: useradd -D [options]

Description: useradd creates a new user or sets the default information for new users.

**tty**

Syntax: tty [OPTION]...

Description: Print the file name of the terminal connected to standard input.

**diff**

Syntax: diff [filename1] [filename2]

Description: diff analyzes two files and prints the lines that are different. Essentially, it outputs a set of instructions for how to change one file in order to make it identical to the second file.

**umask**

Syntax: umask [-S] [mask]

Description: On Linux and other Unix-like operating systems, new files are created with a default set of permissions. Specifically, a new file's permissions may be restricted in a specific way by applying a permissions "mask" called the umask. The umask command is used to set this mask, or to show you its current value.

**Implementation details:** (printout of screen shots)

Please find attached a separate PDF file containing all the screenshots along with the write-up.

(Drive link : <http://goo.gl/3tt4uU>)

**Conclusion:**

Thus, basic commands for handling File system under Unix/Linux using shell scripts have been studied and implemented.

**Post Lab Descriptive Questions (Add questions from examination point view)**

1. In Unix, Which system call creates the new process?  
   a) fork  
   b) create  
   c) new  
   d) none of the mentioned

Answer: **a) fork** - This system call creates new child processes.

1. Explain different functions of operating system.

**Answer:**

The OS typically provides services in the following areas:

**• Program development:** The OS provides a variety of facilities and services, such as editors and debuggers, to assist the programmer in creating programs. Typically, these services are in the form of utility programs that, while not strictly part of the core of the OS, are supplied with the OS and are referred to as application program development tools.

**• Program execution:** A number of steps need to be performed to execute a program. Instructions and data must be loaded into main memory, I/O devices and files must be initialised, and other resources must be prepared. The OS handles these scheduling duties for the user.

**• Access to I/O devices:** Each I/O device requires its own peculiar set of instructions or control signals for operation. The OS provides a uniform interface that hides these details so that programmers can access such devices using simple reads and writes.

**• Controlled access to files:** For file access, the OS must reflect a detailed under- standing of not only the nature of the I/O device (disk drive, tape drive) but also the structure of the data contained in the files on the storage medium. In the case of a system with multiple users, the OS may provide protection mechanisms to control access to the files.

**• System access:** For shared or public systems, the OS controls access to the system as a whole and to specific system resources. The access function must provide protection of resources and data from unauthorised users and must resolve conflicts for resource contention.

**• Error detection and response:** A variety of errors can occur while a computer system is running. These include internal and external hardware errors, such as a memory error, or a device failure or malfunction; and various software errors, such as division by zero, attempt to access forbidden memory location, and inability of the OS to grant the request of an application. In each case, the OS must provide a response that clears the error condition with the least impact on running applications. The response may range from ending the pro- gram that caused the error, to retrying the operation, to simply reporting the error to the application.

**• Accounting:** A good OS will collect usage statistics for various resources and monitor performance parameters such as response time. On any system, this information is useful in anticipating the need for future enhancements and in tuning the system to improve performance. On a multiuser system, the information can be used for billing purposes.

1. Give difference between DOS and WINDOWS.

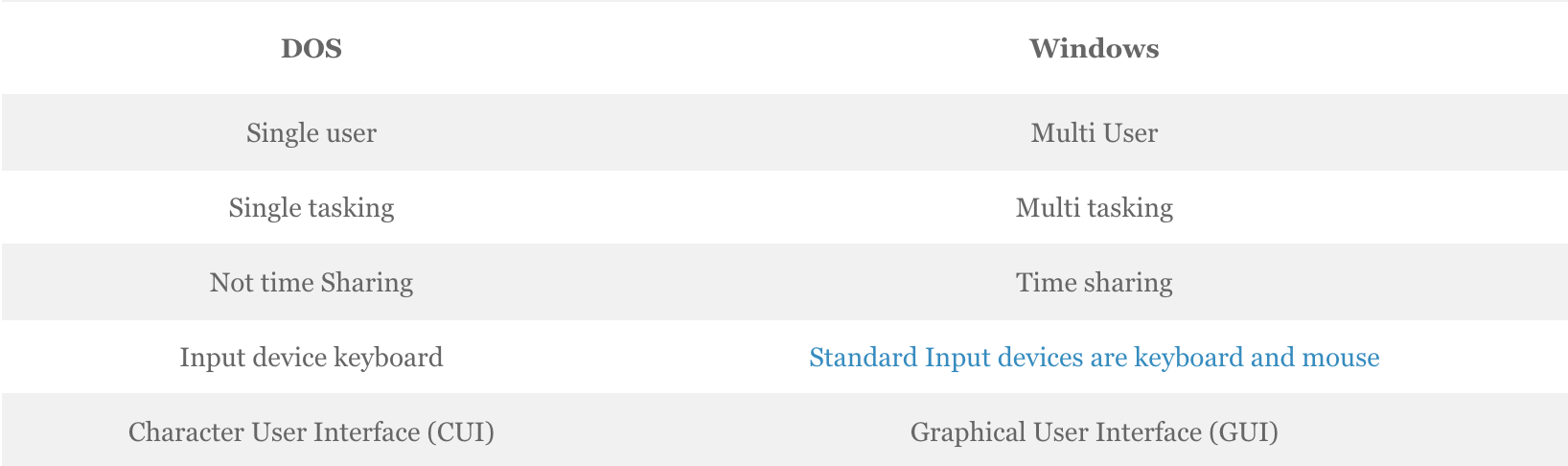
**Answer:**

**FULL ANSWER**

MS DOS is a CLI based OS whereas MS Windows is a GUI based Operating system. A few differentiating points of the two are :

The original MS-DOS operating system, first released in August 1980, used a solely text-based programming language to allow users to work with, or interface with, their PC. Commands were typed into computers at a specific command prompt location on the computer screen using a standard keyboard. Commands had to be precise. Users had to specify what command they wanted, how they wanted it to run, and what program or system on the computer they wished to use. This required users to learn specific language and syntax rules to use their computer properly.

The Windows operating system, released in November 1985, used a graphical user interface instead. Input from the user usually came from using a computer mouse, and commands were run by clicking on representative icons with the virtual pointer controlled by the mouse. There was a small learning curve required to use Windows properly, but it was much easier to interact with graphical representations than text lines and commands, and no special programming language needed to be learned.

Summary:

**Date:** Aug 23, 2016 **Signature of faculty in-charge**