UNIVERSITY OF ENGINEERING AND TECHNOLOGY,

TAXILA

**FACULTY OF TELECOMMUNICATION AND INFORMATION ENGINEERING**

**SOFTWARE ENGINEERING DEPARTMENT**

# Operating Systems

## *Experiment 2*

***Introduction to Basic LINUX Commands –I Shell, file and directories.***

***CLO-3: Use modern tools and languages.***
***CLO-4: Build the projects of varying complexities.***
***CLO-5: Demonstrate the implementation of problem solving process.***
***CLO-6: Work individually as well as in teams.***
***CLO-7: Propose a unique solution that exhibits originality of idea***

UNIVERSITY OF ENGINEERING AND TECHNOLOGY,

TAXILA

**FACULTY OF TELECOMMUNICATION AND INFORMATION ENGINEERING**

**SOFTWARE ENGINEERING DEPARTMENT**

# Implementing Basic Linux Commands

**Shell** a program that takes the commands you type and translates them into instructions to the operating system.

**File:** Under most operating systems (including Linux), there is the concept of a file, which is just a bundle of information given a name (called a **filename**). Examples of files might be your history term paper, an e-mail message, or an actual program that can be executed. Essentially, anything saved on disk is saved in an individual file.

**Directory:** With the concept of files comes the concept of directories. A **directory** is a collection of files. It can be thought of as a ``folder" that contains many different files. Directories are given names, with which you can identify them. Furthermore, directories are maintained in a tree-like structure; that is, directories may contain other directories.

**Current working directory:** At any moment, commands that you enter are assumed to be relative to your current working directory. You can think of your working directory as the directory in which you are currently ``located". When you first log in, your working directory is set to your home directory--/home/me.

# Introduction to Basic Shell Commands

## Command: cat

### General Syntax

```
cat [OPTION] [FILE]...
```

### 1. Display Contents of File

In the below example, it will show contents of **file1/passwd** file.

```
# cat file1/passwd
```

```
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
narad:x:500:500::/home/narad:/bin/bash
```

### 2. View Contents of Multiple Files in terminal

In below example, it will display contents of **test** and **test1** file in terminal.

```
# cat test test1
```

```
Hello everybody
Hi world,
```

### 3. Create a File with Cat Command

We will create a file called **test2** file with below command using "**>**" overwrite operator.

```
# cat > test2
```

Awaits input from user, type desired text and press **CTRL+D** (hold down **Ctrl Key** and type '**d**') to exit. The text will be written in **test2** file. You can see content of file with following **cat** command.

**Note:** if you write again in this file using "cat > test2" command then previous content will be overwritten. To append at the end of previously written content use "**>>**" append operator.

```
# cat test2
```

```
hello everyone, how do you do?
```

### 4. Use Cat Command with More & Less Options

If file having large number of contents that won't fit in output terminal and screen scrolls up very fast, we can use parameters more and less with **cat** command as show above.

```
# cat song.txt | more
# cat song.txt | less
```

### 5. Display Line Numbers in File

With **-n** option you could see the line numbers of a file **song.txt** in the output terminal.

```
# cat -n song.txt
```

```
1  "Heal The World"
2  There's A Place In
3  Your Heart
4  And I Know That It Is Love
5  And This Place Could
6  Be Much
7  Brighter Than Tomorrow
8  And If You Really Try
9  You'll Find There's No Need
```

```
10  To Cry
11  In This Place You'll Feel
12  There's No Hurt Or Sorrow
```

### 6. Display $ at the End of File

In the below, you can see with **-e** option that '**$**' is shows at the end of line and also in space showing '**$**' if there is any gap between paragraphs. This options is useful to squeeze multiple lines in a single line.

```
# cat -e test
```

```
hello everyone, how do you do?$
$
Hey, am fine.$
How's your training going on?$
$
```

### 7. Display Tab separated Lines in File

In the below output, we could see **TAB** space is filled up with '**^I**' character.

```
# cat -T test
```

```
hello ^Ieveryone, how do you do?

Hey, ^Iam fine.
^I^IHow's your training ^Igoing on?
Let's do ^Isome practice in Linux.
```

### 8. Display Multiple Files at Once

In the below example we have three files **test**, **test1** and **test2** and able to view the contents of those file as shown above. We need to separate each file with **;** (semi colon).

```
# cat test; cat test1; cat test2
```

```
This is test file
This is test1 file.
This is test2 file.
```

### 9. Use Standard Output with Redirection Operator(Overwriting contents of file with another file)

We can redirect standard output of a file into a new file else existing file with '>' (greater than) symbol. Careful, existing contents of **test1** will be overwritten by contents of **test** file.

```
# cat test > test1
```

**10. Appending Standard Output with Redirection Operator(Appending contents of file at end of another file)**

Appends in existing file with **'>>' (double greater than**) symbol. Here, contents of **test** file will be appended at the end of **test1** file.

```
# cat test >> test1
```

**11. Redirecting Standard Input with Redirection Operator**

When you use the redirect with standard input **'<' (less than symbol**), it use file name **test2** as a input for a command and output will be shown in a terminal.

```
# cat < test2
```

```
This is test2 file.
```

**12. Redirecting Multiple Files Contain in a Single File**

This will create a file called **test3** and all output will be redirected in a newly created file.

```
# cat test test1 test2 > test3
```

# 2. Command: pwd

**Syntax:** pwd [*OPTION*]

**Description:** "pwd" stands for print working directory. It displays your current position in the LINIX file system.

**Examples:**

- pwd

It is simply used to report your current working directory.

# 4. Command: ls

**Syntax:** ls [options] [names]

**Description:** "ls" stands for list. It is used to list information about files and directories.

**Examples:**

- ls

This is the basic "ls" command, with no options. It provides a very basic listing of the files in your current working directory. **Filenames beginning with a decimal are considered *hidden* files,** and they are not shown.

```
ls -al
```

This command provides a *long* listing of information about *all* files in the current directory. This is probably the most used version of the ls command.

```
ls -a
```

This command provides a listing of information about *all hidden* files and directories in the current directory whose name begins with a dot.

## 4. Command: mv

**Syntax:** mv [options] source target
**Description:** The "mv" command is used to rename files.

**Examples:**

```
mv Chapter1 garbage
```

This command renames the file "Chapter1" to the new name "garbage".

## 5. Command: rm and rm -r

**Description:** The "rm" command is used to remove files.
The “rm -r” command is used to remove directories..
(Warning - be very careful when removing files!)

**Examples:**

```
rm Chapter1.bad
```

This command deletes the file named "Chapter1.bad"assuming you have permission to delete this file).

```
rm Chapter1 Chapter2 Chapter3
```

This command deletes the files named "Chapter1", "Chapter2", and "Chapter3".

```
rm -i Chapter1 Chapter2 Chapter3
```

This command prompt you before deleting any of the three files specified. **The -i option** stands for ***inquire*.** You must answer y (for yes) for each file you want to delete. This can be a safer way to delete files.

```
rm *.html
```

This command deletes all files in the current directory whose filename ends with the characters ".html".

```
rm index*
```

This command deletes all files in the current directory whose filename begins with the character's "index".

```
rm -r new-novel
```

This **command deletes the directory** named "new-novel". This directory, and all of its contents, are erased from the disk, including any sub-directories and files.

## 6. Command: cp

**Syntax:** cp [options] file1 file2
cp [options] files directory

**Description:** The "cp" command is used to copy files and directories. Note that when using the cp command, you must always specify both the source and destination of the file(s) to be copied.

**Examples:**

```
cp -r file1 Lab2
```

This command copies your file "file1" to a directory named "Lab2".

## 7. Command: mkdir

**Syntax:** mkdir [options] directory name

**Description:** The "mkdir" command is used to create new directories (sub-directories).

**Examples:**

```
mkdir tmp
```

This command creates a new directory named "tmp" in your current directory. (This example assumes that you have the proper permissions to create a new sub-directory in your current working directory.)

```
mkdir memos letters e-mail
```

This command creates three new sub-directories (memos, letters, and e-mail) in the current directory.

| mkdir -p /parent/child/customer/acme |
|---|

This command creates a new directory named / parent/child/customer/acme, and creates any intermediate directories that are needed.

## 8. Command: rmdir

**Syntax:** rmdir [options] directories

**Description:** The "rmdir" command is used to remove directories. (Warning - be very careful when removing directories!)

**Examples:**

| rmdir Lab2 |
|---|

This command deletes the directory named "Lab2" (assuming you have permission to delete this directory).

- **~ files** with this symbol are backup files. We can recover them. To remove these backup files, we use **rm *~.** It will remove all the backup files.

**Command: clear**

| Clear the Window |
|---|

**Command: exit**

| Exit a shell |
|---|

UNIVERSITY OF ENGINEERING AND TECHNOLOGY,

TAXILA

**FACULTY OF TELECOMMUNICATION AND INFORMATION ENGINEERING**

**SOFTWARE ENGINEERING DEPARTMENT**

## Lab Task

# BASIC LINIX COMMAND EXERCISES

1. Verify that you are in your home directory.
2. Change to directory ***Desktop.*** Reside in this Directory.
3. Create 2 more directories in pwd, named “File1” and “File2”.
4. Create a file named as **FisrtFile** in *directory* File1 with some content.
5. Create a file named as **SecondFile** *directory* File2 with some content.
6. **View** Contents of both files **at once** on monitor.
7. **Append** the content of **FisrtFile** in **2nd** File. **Verify** the Change in **2nd** File.
8. **Overwrite** the existing contents of **FisrtFile** by contents of **SecondFile** file. Verify the Change in **SecondFile** File.
9. **Make a copy** of the file **FisrtFile** under the name **SecondFile.**
11. **Delete** the file **FisrtFile.**
12. **Verify** that **FisrtFile.** has been deleted.
13. **Clear** the window.