Document Number:	*Copy Number:*

TATA CONSULTANCY SERVICES

PRE ILP – Unix LOUNGE

Content Manual Version 1.0

March 2014

DOCUMENT RELEASE NOTICE

Notice No.
Client
Project
Document details

Name	Version No.	Author	Description

Revision details:

Action taken (add/del/chg)	Preceding page No.	New page No.	Revision description

Change Register serial numbers covered:
The documents or revised pages are subject to document control.
Please keep them up-to-date using the release notices from the distributor of the document.
These are confidential documents. Unauthorised access or copying is prohibited.

Approved by : _____ Date: _____

Authorised by: _____ Date: _____

DOCUMENT REVISION LIST

Client
Project
Document Name
Release Notice Reference (for release)

Rev. No.	Revision date	Revision description	Page No.	Prev page No.	Action taken	Addenda /New page	Release Notice reference

CONTENTS

CHAPTER 3 - UNIX USERS AND FILE PERMISSIONS

3.1 Objective

To understand about the different users in UNIX and their access rights.

3.2 Course Content

- ⇧ Type of users
- ⇧ File permissions
- ⇧ Changing permissions using symbolic mode
- ⇧ Changing permissions using octal mode

3.2.1 Types of users in Unix

There are three types of accounts on a Unix system:
- Root Account
- System Account
- User Account

Root	System	User
• This is also called superuser and would have complete and unfettered control of the system. • A superuser can run any commands without any restriction. This user should be assumed as a system administrator.	• System accounts are those needed for the operation of system-specific components for example mail accounts and the sshd accounts. • These accounts are usually needed for some specific function on your system, and any modifications to them could adversely affect the system.	• User accounts provide interactive access to the system for users and groups of users. • General users are typically assigned to these accounts and usually have limited access to critical system files and directories.

3.2.2 Managing users and groups

- Unix supports a concept of Group which logically groups a number of accounts.
- Every account would be a part of any group.
- Unix groups plays important role in handling file permissions and process management.
- Grouping of users allows to grant and revoke file permissions collectively.

/etc/passwd: Keeps user account and password information.

/etc/shadow: Holds the encrypted password of the corresponding account. Not all the system support this file.

/etc/group: This file contains the group information for each account.

/etc/gshadow: This file contains secure group account information.

3.2.3 File Permission: Ownerships level

File ownership is an important component of Unix that provides a secure method for storing files. Every file in Unix has the following attributes:

Permission	Symbol	Description
Owner Permissions	**u**	The owner's permissions determine what actions the owner of the file can perform on the file.
Group Permissions	**g**	The group's permissions determine what actions a user, who is a member of the group that a file belongs to, can perform on the file.
Other Permissions	**o**	The permissions for others indicate what action all other users can perform on the file.

3.2.4 File Permission Types (Mode)

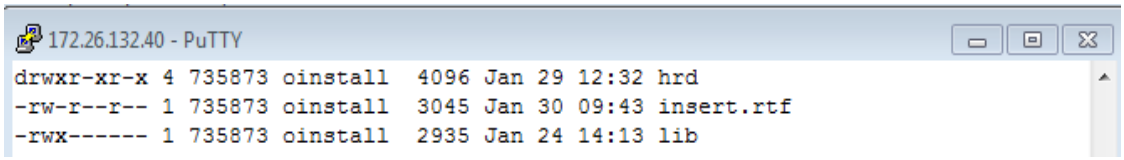
Three type of permissions can be set any ownership level:
Read, Write and Execute

⇧ Three modes or permissions have different meaning for file and directory;

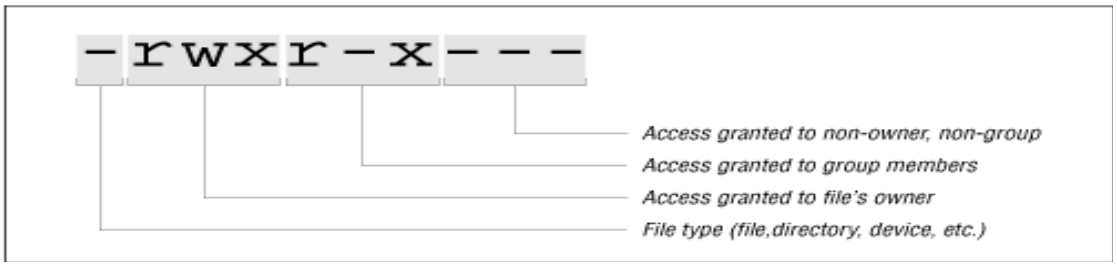
Mode	Symbol	For File	For Directory
Read	**r**	Allows to view the content	Allows to view the directory content using **ls** command
Write	**w**	Allows edit and save changes.	Create and delete files in the directory
Execute	**x**	Allows to run program or script	Allows to access the files in the directory

3.2.5 Unix - File Permission / Access Modes

Consider the output of the command ls -l



- Out of 9 columns in output, 1st column indicates the file type and its permissions.
- This Column contains 10 characters.
- Character 1 : Represents File Type ('d' : Directory; '-' : Regular File)
- Character 2-4 : Represents Permission for owner(user) of file
- Character 5-7 : Represents Permission for group user belongs to.
- Character 8-10 : Represents Permission for other



3.2.6 Changing Permissions (chmod)

To change file or directory permissions, the chmod (change mode) command is used.

There are two ways to use chmod:

- Symbolic mode
- Absolute mode

Syntax:

```
$ chmod [OPTION] MODE          FILENAME
$ chmod [OPTION] OCTAL-MODE FILENAME
```

chmod command options:

-f, --silent, --quiet suppress most error messages

-R, --recursive change files and directories recursively

--help display this help and exit

3.2.6.1 Using chmod in Symbolic Mode

With symbolic representation, permission set can be added, deleted, or specified using the operators in the following table:

chmod operators	Description
+	Adds the designated permission(s) to a file or directory.
-	Removes the designated permission(s) from a file or directory.
=	Sets the designated permission(s).

For example, to give everyone execute permission for a file, you can use :

$chmod ugo+x filename

or

$chmod a+x filename

For example, to remove execute permission from others and retain the

existing permissions for all for a file, you can use :

$chmod o-x filename

```
-bash-3.2$ ls -l abc
-r--r--rwx 1 735873 oinstall 0 Feb  7 12:37 abc
-bash-3.2$ chmod u+wx,g+wx abc
-bash-3.2$ ls -l abc
-rwxrwxrwx 1 735873 oinstall 0 Feb  7 12:37 abc
-bash-3.2$ chmod g-x abc
-bash-3.2$ ls -l abc
-rwxrw-rwx 1 735873 oinstall 0 Feb  7 12:37 abc
-bash-3.2$ chmod o=r-- abc
-bash-3.2$ ls -l abc
-rwxrw-r-- 1 735873 oinstall 0 Feb  7 12:37 abc
-bash-3.2$ chmod a+rwx abc
-bash-3.2$ ls -l abc
-rwxrwxrwx 1 735873 oinstall 0 Feb  7 12:37 abc
```

In the above example,

$ chmod u+wx,g+wx abc , adds execute permission to user and group to the file abc

$ chmod g-x abc, removes execute permission from group from the file abc.

$ chmod o=r-- abc, provides read permission and removes write and execute permission from others for file abc

3.2.6.2 Using chmod in Absolute(octal) Mode

- The second way to modify permissions with the chmod command is to use a number to specify each set of permissions for the file.
- Each permission is assigned a value, as the following table shows, and the total of each set of permissions provides a number for that set.

Number	Octal Permission Representation	Ref
0	No permission	---
1	Execute permission	--x
2	Write permission	-w-
3	Execute and write permission: 1 (execute) + 2 (write) = 3	-wx
4	Read permission	r--
5	Read and execute permission: 4 (read) + 1 (execute) = 5	r-x
6	Read and write permission: 4 (read) + 2 (write) = 6	rw-
7	All permissions: 4 (read) + 2 (write) + 1 (execute) = 7	rwx

Examples:

```
-bash-3.2$ ls -l abc
-rw-r--r-- 1 735873 oinstall 0 Feb  7 12:37 abc
-bash-3.2$ chmod 0 abc
-bash-3.2$ ls -l abc
---------- 1 735873 oinstall 0 Feb  7 12:37 abc
-bash-3.2$ chmod 743 abc
-bash-3.2$ ls -l abc
-rwxr---wx 1 735873 oinstall 0 Feb  7 12:37 abc
-bash-3.2$ chmod 755 abc
-bash-3.2$ ls -l abc
-rwxr-xr-x 1 735873 oinstall 0 Feb  7 12:37 abc
-bash-3.2$ chmod 777 abc
-bash-3.2$ ls -l abc
-rwxrwxrwx 1 735873 oinstall 0 Feb  7 12:37 abc
```

3.3 Video 3 : Users, File Permission

http://www.youtube.com/watch?v=zRw0SKaXSfI

3.4 Quiz Time

Q1. Which is the superuser for Unix

A. system
B. root

C. all users
D. there's no such user

Answer :B

Q2. What is stored in the /etc/passwd file?

A. user account information
B. user password
C. Both A and B
D. password configuration details

Answer: C

Q3. Which command is used to change the file permissions in Unix?

A. perm
B. sysconfig
C. systemconfig
D. chmod

Answer: D

Q4. Unix logically groups the users.

A. True
B. False
C. Not completely
D. Sometimes

Answer: A

CHAPTER 4 - UNIX DIRECTORY COMMANDS

4.1 Objective

- To provide overview of the most common commands to work with directories. These commands are available on any Linux (or Unix) system.

4.2 Course Content

The commands discussed include:

- mkdir
- cd

- cp
- mv
- rmdir
- rm

4.2.1 Introduction

What is a directory?

When a user interacts with the UNIX shell, the shell considers the user to be located somewhere within a filesystem. Every item in the UNIX filesystem tree is either a file, or a directory. A directory is like a file folder. A directory can contain files, and other directories. A directory contained within another is called the *child* of the other. A directory in the filesystem tree may have many children, but it can only have one parent. A file can hold information, but cannot contain other files, or directories. A directory is actually implemented as a file that has one line for each item contained within the directory. Each line in a directory file contains only the name of the item, and a numerical reference to the location of the item. The reference is called an *i-number(inode)*, and is an index to a table known as the*i-list*. The i-list is a complete list of all the storage space available to the file system. The place in the file system tree where an user is located is called the *current working directory*.

- A file in unix file system which contains information about other files .
- It contains information about files and directories through which UNIX looks up to obtain information about a particular file.
- As we login, we are put in our home directory.
- At a time where the user is located is called the *current working directory*.

4.2.2 mkdir(make directory) command:

The mkdir command creates a directory with specified name in the present working directory or specified path.

Syntax: $ mkdir <dir_name> [<dir_name2>]

```
$ mkdir  Games
$ ls
Games
$
```

```
$ mkdir  Games/Indoor Games/Outdoor
$ ls Games
Indoor Outdoor
$
```

The above command creates two directories, Indoor and Outdoor in specified path under Games

4.2.3 cd (Change Directory) command:

cd command provides navigation between directories.
$ cd / : Takes you to root directory

```
$ pwd
/home/amit
$ cd /
$ pwd
/
$
```

$ cd ~ : Takes you to home Directory (/home/userid)

```
$ pwd
$ /
$ cd ~
$ pwd
/home/amit
$
```

$ cd .. : Takes you to current parent Directory

```
$ pwd
/home/amit
$ cd ..
$ pwd
/home
$
```

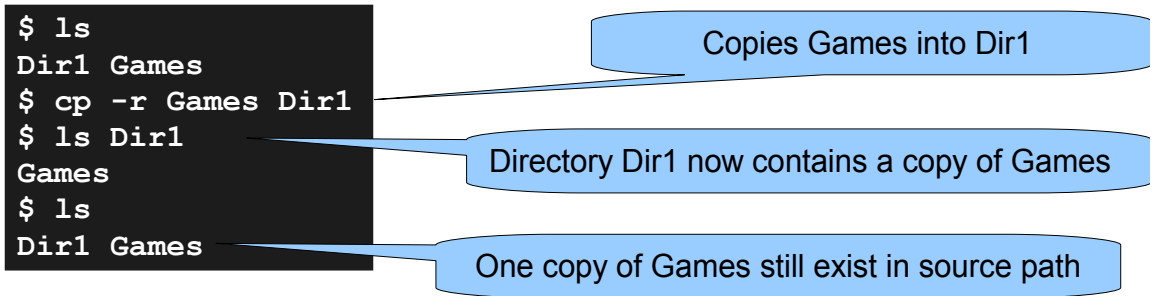
$ cd <dir_name> : Takes you to the desired directory

```
$ pwd
/home
$ cd amit
$ pwd
/home/amit
$cd Games/Indoor
$ pwd
/home/amit/Games/Indoor
```

4.2.4 cp command:

The command cp with -r option copies a directory and all its contents(sub-directory and files) recursively to another.

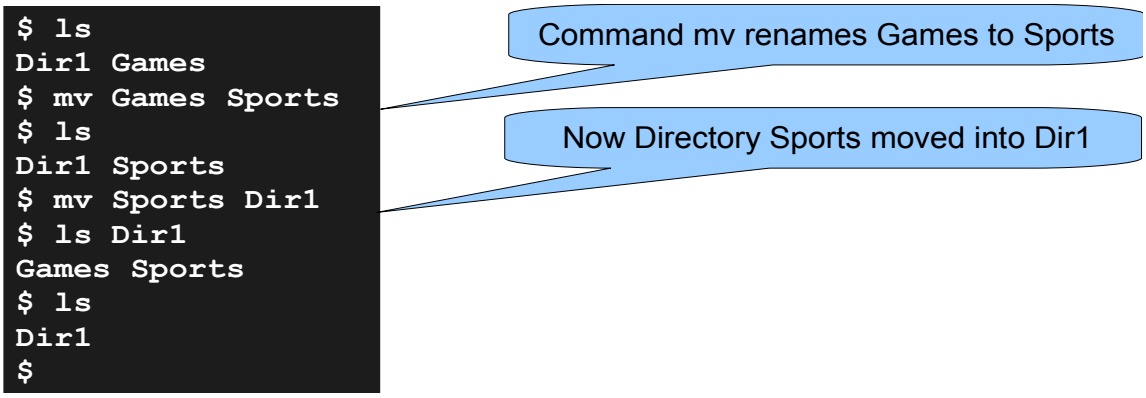
Syntax:
cp -r <source_directory> <destination_path>



4.2.5 mv Command:

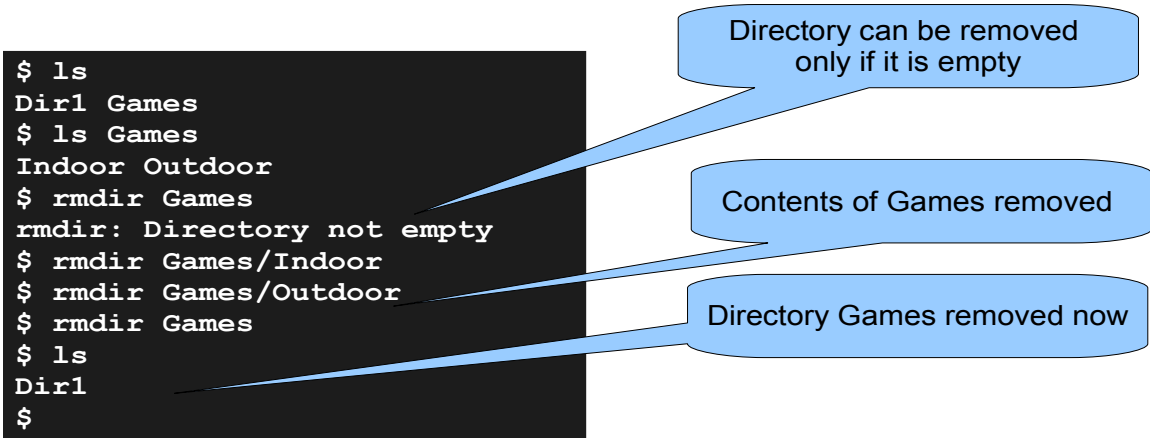
Moves the contents of a file or directory.
If the source and destination path is same it renames the file or directory.

Syntax: $ mv <source_dir> <destination_dir>



4.2.6 rmdir(remove directory) command:

To remove any existing empty directory.
Syntax: rmdir <dir_name>
Remark: To delete a Directory with content/which is not empty rm command can be used.
Syntax: rm -R /path

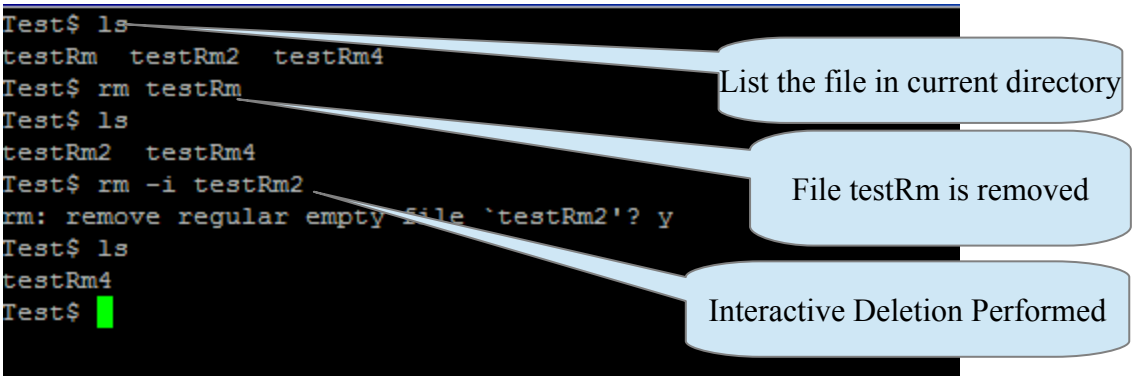


4.2.7 rm (remove files) command:

Removes existing files and directories.

Syntax: rm filename

Remark: The file being deleted may contain important information,so “-i” option is recommended to be used along with rm command.



4.2.8 Video 4: Directory commands

http://www.youtube.com/watch?v=VPgrtk0HQB0&list=PL8A83A276F0D85E70
http://www.youtube.com/watch?v=HsBEzs6Q7w4&list=PL8A83A276F0D85E70

4.3 Quiz Time

Q1. Which command in Unix will help in creating new directories?

A. mkdir
B. makedir
C. create
D. new

Answer: A

Q2. Which command will help in deleting an empty directory?

A. delete
B. deletedir
C. remove
D. rmdir

Answer: D

Q3. Which command will help in moving a directory?

A. move
B. dirmover
C. mv
D. mvdir

Answer: C

Q4. Which command is used to rename a directory?

A. rename
B. changename
C. remdir
D. mv

Answer: D

Q5. Which option when used with cp will copy an directory will all its contents?

A. -R
B. -a
C. -m
D. -f

Answer: A