

## UNIX BASIC FILE AND DIRECTORY COMMANDS

### 1. Command Name: pwd

**Stands for:** Present Working Directory – shows the current directory (folder) we are in.

**Syntax:**

**\$ pwd**

**Sample Output:** /home/MSK/SS/18PW/18PW19

### 2. Command Name: mkdir

**Stands for:** Make Directory – **mkdir** command in Linux allows the user to create directories (also referred to as folders in some operating systems). This command can create multiple directories at once as well as set the permissions for the directories. It is important to note that the user executing this command must have write permissions to create a directory in the parent directory, or you may receive a 'permission denied' error.

**Syntax:**

**\$ mkdir dirname**

**Example:**

**\$ mkdir unix**

**-creates a folder called unix in the current directory**

Execute this **ls** command to list all the files and directories in the current directory and check whether unix directory has been created.

**\$ ls**

You can also create the directory **myfiles** in your home directory as shown below, specified here with a tilde ("~"), if you not currently in your home directory.

**\$ mkdir ~/myfiles**

**Syntax using -v option:**

**\$ mkdir -v directory names**

-It displays a message for every directory created.

### Example:

```
Terminal
File Edit View Search Terminal Help
rossoskull@RossoSkull ~/GFG $ mkdir -v one two three
mkdir: created directory 'one'
mkdir: created directory 'two'
mkdir: created directory 'three'
rossoskull@RossoSkull ~/GFG $ ls
one three two
rossoskull@RossoSkull ~/GFG $
```

### Syntax using -p option:

**\$ mkdir -p pathname of directory** - A flag which enables the command to create parent directories as necessary. If the directories exist, no error is specified.

### Example:

```
Terminal
File Edit View Search Terminal Help
rossoskull@RossoSkull ~/GFG $ mkdir -p -v first/second/third
mkdir: created directory 'first'
mkdir: created directory 'first/second'
mkdir: created directory 'first/second/third'
rossoskull@RossoSkull ~/GFG $
```

If -p option not used with mkdir then, if any of the parent directories doesn't exist, this command shows error.

```
Terminal
File Edit View Search Terminal Help
rossoskull@RossoSkull ~/GFG $ mkdir first/second/third
mkdir: cannot create directory 'first/second/third': No such file or directory
rossoskull@RossoSkull ~/GFG $
```

Using 'ls -R' command you can see the recursive listing of all folders in the directory as shown below.

```
test@test-VirtualBox ~/Desktop/Linux
File Edit View Search Terminal Help
test@test-VirtualBox:~/Desktop/Linux$ mkdir -p Linux/dirtest1/dirtest2
test@test-VirtualBox:~/Desktop/Linux$ ls -R
.:
Linux

./Linux:
dirtest1

./Linux/dirtest1:
dirtest2

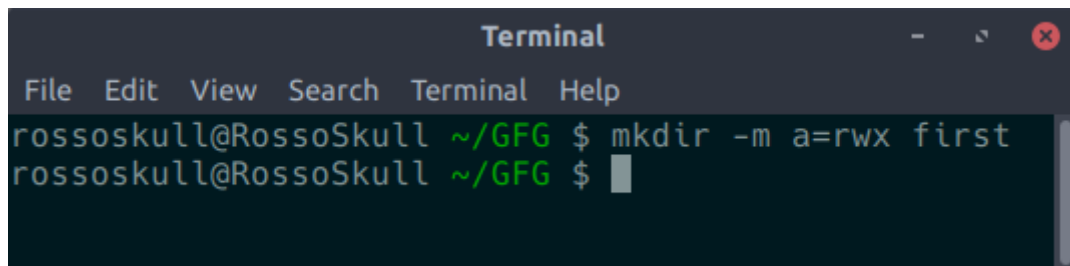
./Linux/dirtest1/dirtest2:
test@test-VirtualBox:~/Desktop/Linux$
```

**Syntax using -m option** - This option is used to set the file modes, i.e. permissions, etc. for the created directories.

**Syntax:**

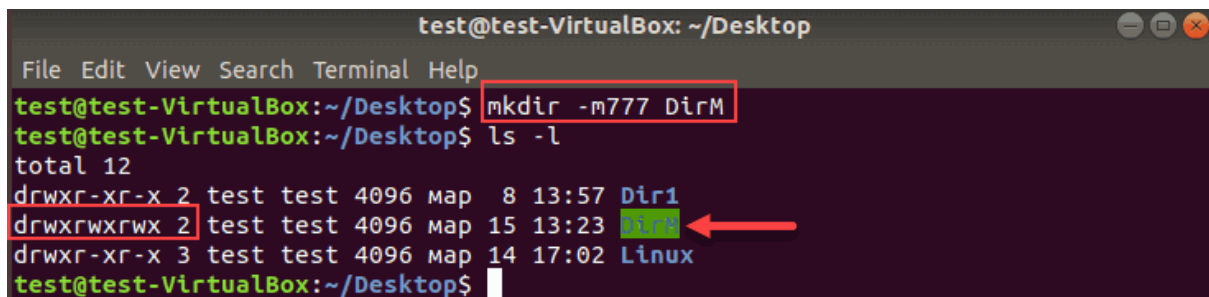
**mkdir -m a=rwx [directories]**

The above syntax specifies that the directories created give access to all the users to read from, write to and execute the contents of the created directories. You can use 'a=r' to only allow all the users to read from the directories and so on.

A terminal window titled "Terminal" with a menu bar (File, Edit, View, Search, Terminal, Help). The prompt is "rossoskull@RossoSkull ~/GFG \$". The command "mkdir -m a=rwx first" has been entered and executed. The prompt is now "rossoskull@RossoSkull ~/GFG \$".

```
Terminal
File Edit View Search Terminal Help
rossoskull@RossoSkull ~/GFG $ mkdir -m a=rwx first
rossoskull@RossoSkull ~/GFG $
```

Or

A terminal window titled "test@test-VirtualBox: ~/Desktop" with a menu bar (File, Edit, View, Search, Terminal, Help). The prompt is "test@test-VirtualBox:~/Desktop\$". The command "mkdir -m777 DirM" has been entered and executed. The prompt is now "test@test-VirtualBox:~/Desktop\$". The command "ls -l" has been entered and executed, showing the output. The permissions "drwxrwxrwx" for "DirM" are highlighted with a red box, and a red arrow points to it.

```
test@test-VirtualBox: ~/Desktop
File Edit View Search Terminal Help
test@test-VirtualBox:~/Desktop$ mkdir -m777 DirM
test@test-VirtualBox:~/Desktop$ ls -l
total 12
drwxr-xr-x 2 test test 4096 map 8 13:57 Dir1
drwxrwxrwx 2 test test 4096 map 15 13:23 DirM
drwxr-xr-x 3 test test 4096 map 14 17:02 Linux
test@test-VirtualBox:~/Desktop$
```

Read(r)=4

Write(w)=2

Execute(x)=1

So r+w+x=7. 777 indicates all the three permissions for owner, group and others (rwxrwxrwx)

### 3. Command Name: cd

**Stands for:** Change Directory - It takes us to the directory we specify after cd, if it exists or else shows error or permission denied (if we are not allowed to enter into the directory)

[To switch to a directory, you must have executable permissions for that directory it.](#)

**Syntax:**

**\$ cd [directory]**

**Example:**

**\$ cd unix**

unix is a directory in the current directory. The above command is same as

**\$ cd ./unix**

This is the same as the command:

**cd unix/**

**Other Examples:**

**cd /home/MS/SS/18PW/18PW19/unix/mydir**

- This cd command uses **absolute pathname** of the mydir directory starting from /(root)folder. This command can be used to move to mydir directory immaterial of the directory currently we are in.

**cd unix/mydir**

- This cd command uses **relative pathname** of mydir directory. If we are already in /home/MS/SS/18PW/18PW19 we can use this command.

### Different functionalities of cd command:

- **cd /** - this command is used to change directory to the root directory.  
The root directory is the first directory in your filesystem hierarchy.
  - **\$ cd /**
  - **Now, check in which directory you are in using the command**
  - **\$ pwd**
- **cd dir\_1/dir\_2/dir\_3** -This command is used to move inside a directory from a directory
  - **Syntax:**
  - **\$ cd dir\_1/dir\_2/dir\_3**
  - **Example:**
  - **\$ cd unix/mydir/pbsheet**
  - **But all these directories must already be available, else the command will show error. Now, check in which directory you are in using the command**
  - **\$ pwd**
- **cd ~** or **cd** -this command is used to change directory to the home directory.  
  
**\$ cd ~**  
  
**Or just type**  
  
**\$ cd**  
  
**Now, check in which directory you are in using the command**  
  
**\$ pwd**

- For example, if you want to navigate to the Downloads directory, which is inside your home directory, you would type:

- **cd ~/Downloads**

- You can also navigate to another user's home directory using the following syntax:

- **cd ~username**

- **cd ..** - this command is used to move to the parent directory of current directory, or the directory one level up from the current directory. **“..” represents parent directory.**

- **\$ cd ..**

- **cd “dir name”** - This command is used to navigate to a directory with white spaces. Instead of using double quotes we can use single quotes then also this command will work.

- **Syntax: \$ cd "dir name"**

- **Example:**

- **\$ cd “My Songs”**

**Or**

- The following command work same as cd “My Songs” command.

- **Example:**

- **\$ cd My\ Songs**

- **cd -** -Switch back to previous directory where you working earlier.
- **cd --** - Show last working directory from where we moved
- **cd ../../** - Move two directory up from where you are now.

#### 4. Multiple ways to create files:

**Example:** Create empty files called 'file1' and 'file2'

**\$ touch file1 file2**

**Example:** Create file1 with contents got as input from user

**\$ cat > file1**

Hello

^d

**Example:** Create file1 with contents of file2.

**\$ cat > file2 < file1**

The above command takes as input the contents of file1 and creates file2 with file1's content. Similar to copying a file.

**Example:** Create file with editors like vim, vi, nano.....

**\$ nano new-filename**

**Example:**

Storing output of commands in files

**\$ ls > file**

**To display the contents of a file the command used is**

**\$ cat filename**

**Or**

**Can open the file in an editor like nano to see its contents**

**\$ nano existing-filename**

## 5. Redirection

Redirection is a feature in Linux such that when executing a command, you can change the standard input/output/error devices. The basic workflow of any Linux command is that it takes an input and give an output or error.

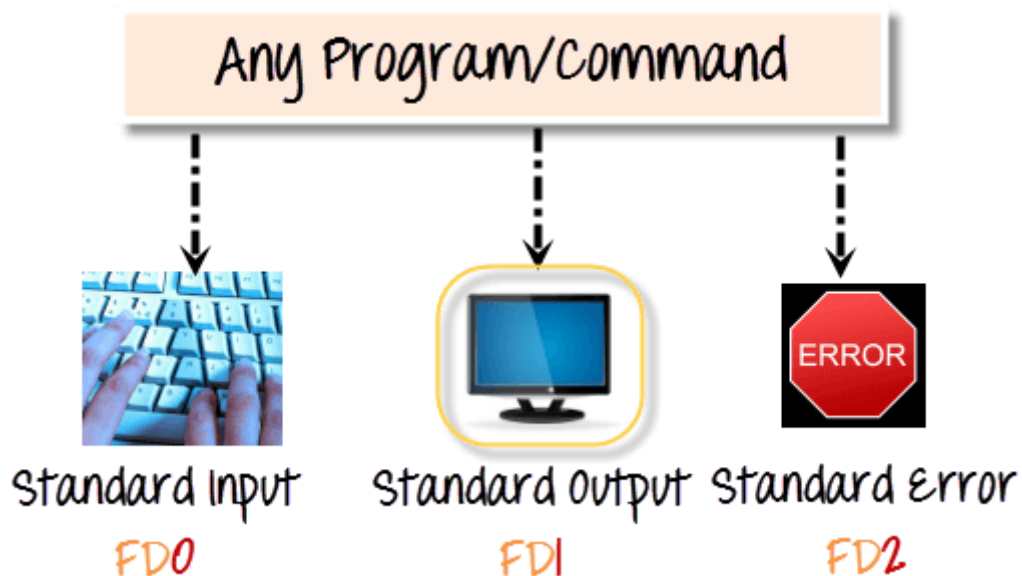
- The standard input (stdin) device is the keyboard.
- The standard output (stdout) device is the screen.
- The standard error (stderr) device is the screen.

### File Descriptors (FD)

**In Linux/Unix, everything is a file. Regular file, Directories, and even Devices are files.** Every File has an associated number called File Descriptor (FD).

Your screen also has a File Descriptor. When a program is executed the output is sent to File Descriptor of the screen, and you see program output on your monitor. If the output is sent to File Descriptor of the printer, the program output would have been printed.

Whenever you execute a program/command at the terminal, 3 files are always open, viz., standard input, standard output, standard error.



These files are always present whenever a program is run. As explained before a file descriptor, is associated with each of these files.

File	File Descriptor
Standard Input STDIN	0
Standard Output STDOUT	1
Standard Error STDERR	2



By default, error stream is displayed on the screen. Error redirection is routing the errors to a file other than the screen.

## Output Redirection

The '>' symbol is used for output (STDOUT) redirection.

### Example:

Execute the command and see the output

```
$ ls -al
```

**(Note:** Just **'ls'** commands list the names of all files in the directory. **'ls -l'** gives detailed information about all files. **'ls -a'** will give you the whole list of a directory including the hidden files also. In Unix/Linux, hidden files start with a dot (.) and can't be seen in the regular directory. So **'ls -al'** will give detailed listing of all files including hidden files).

Now execute

```
$ ls -al > listings
```

**Here the output of command `ls -al` is re-directed to file "listings" instead of your screen.**

```
home@VirtualBox:~$ ls -al > listings
home@VirtualBox:~$ cat listings
total 324
drwxr-xr-x 26 home home 4096 2012-09-10 10:42 .
drwxr-xr-x  3 root root 4096 2012-09-01 19:43 ..
-rw-rw-r--  1 home home    0 2012-09-10 09:25 abc
```

**Note:** Use the correct file name while redirecting command output to a file. If there is an existing file with the same name, the redirected command will delete the contents of that file and then it may be overwritten."

If you do not want a file to be overwritten but want to add more content to an existing file, then you should use '>>' operator for appending contents to a file.

```
home@VirtualBox:~$ cat sample
Hang on for the best Linux Lessons.
home@VirtualBox:~$ echo Thanks for reading >> sample
home@VirtualBox:~$ cat sample
Hang on for the best Linux Lessons.
Thanks for reading
```

You can redirect standard output, to not just files, but also devices!

```
$ cat music.mp3 > /dev/audio
```

The cat command reads the file music.mp3 and sends the output to /dev/audio which is the audio device. If the sound configurations in your PC are correct, this command will play the file music.mp3

### **Input redirection**

The '<' symbol is used for input(STDIN) redirection

#### **Example:**

```
$ wc < file1
```

- Reads the contents from the file1 as input and display the number of lines, words and chars in file1. **Here the contents of file1 is redirected as input to the cat command.**

### **Why Error Redirection?**

Error re-direction is one of the very popular features of Unix/Linux. Frequent UNIX users will reckon that many commands give you massive amounts of errors.

- For instance, while searching for files, one typically gets permission denied errors. These errors usually do not help the person searching for a particular file.
- While executing shell scripts, you often do NOT want error messages cluttering up the normal program output.

The solution is to re-direct the error messages to a file.

#### **Example 1**

Assume you have unix directory in your current directory. Now execute the command

```
$ mkdir unix
```

-you will get an error. So instead redirect error to an error file if you don't want to see error as shown below

```
$ mkdir unix 2 > errorfile
```

The file descriptor for standard error is 2. Using "2>" we re-direct the error output to a file named "errorfile"