Mawlana Bhashani Science and Technology University

Lab-Report

Report No: 02

Course code: ICT-3110

Course title: Operating Systems Lab

Date of Performance :

Date of Submission : 14/09/2020

Submitted by

Name: Mahfuza Talukdar

ID:IT-18009

3rd year 1st semester

Session: 2017-2018

Dept. of ICT

MBSTU.

Submitted To

Nazrul Islam

Assistant Professor

Dept. of ICT

MBSTU.

Experiment No : 02

Experiment Name : Basic Command of Linux operating System.

Theory :

The Linux command line is a text interface to your computer. Often referred to as the shell, terminal, console, prompt or various other names, it can give the appearance of being complex and confusing to use. Yet the ability to copy and paste commands from a website, combined with the power and flexibility the command line offers, means that using it may be essential when trying to follow instructions online, including many on this very website.

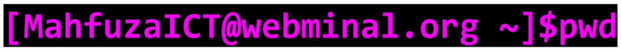
Linux Operating System's 15 commands –

(i) pwd
(ii) ls
(iii) cd
(iv) mkdir & rmdir
(v) rm
(vi) touch
(vii) man & --help
(viii) cp
(ix) mv
(x) df
(xi) echo
(xii) cat
(xiii) nano, vi, jed
(xiv) uname
(xv) hostname

Commanding Process :

The operation of Linux Basic Command –

1. **pwd** — When we first open the terminal, we are in the home directory of our user. To know which directory we are in, we can use the **"pwd"** command. It gives us the absolute path, which means the path that starts from the root. The root is the base of the Linux file system. It is denoted by a forward slash(/). The user directory is usually something like "/home/username".



```
[MahfuzaICT@webminal.org ~]$pwd
```

```
/home/MahfuzaICT
```

2. **ls** — we use the **"ls"** command to know what files are in the directory we are in. We can see all the hidden files by using the command **“ls -a”**.

```
[MahfuzaICT@webminal.org ~]$ls
mahfuza
```

3.cd — We use the **"cd"** command to go to a directory. For example, if we are in the home folder, and we want to go to the downloads folder, then we can type in **“cd Downloads”**. Remember, this command is case sensitive, and we have to type in the name of the folder exactly as it is. But there is a problem with these commands. Imagine we have a folder named “Raspberry Pi”. In this case, when we type in **“cd Raspberry Pi”**, the shell will take the second argument of the command as a different one, so we will get an error saying that the directory does not exist. Here, we can use a backward slash. That is, we can use **“cd Raspberry\ Pi”** in this case. Spaces are denoted like this: If we just type **“cd”** and press enter, it takes you to the home directory. To go back from a folder to the folder before that, you can type “**cd** ..” . The two dots represent back.

```
[MahfuzaICT@webminal.org ~]$cd mahfuza
[MahfuzaICT@webminal.org mahfuza]$pwd
/home/MahfuzaICT/mahfuza
```

4.mkdir & rmdir — We use the **mkdir** command when we need to create a folder or a directory. For example, if we want to make a directory called “DIY”, then we can type **“mkdir DIY”**. Remember, as told before, if we want to create a directory named “DIY Hacking”, then we can type “mkdir **DIY\ Hacking**”. Use **rmdir** to delete a directory.
But **rmdir** can only be used to delete an empty directory. To delete a directory containing files, use **rm**.

```
[MahfuzaICT@webminal.org mahfuza]$mkdir -v koli
mkdir: created directory âkoliâ
[MahfuzaICT@webminal.org mahfuza]$ls
  koli  mahfuza  mahfuzatalukdar  pori  talukdar
```

```
[MahfuzaICT@webminal.org mahfuza]$mkdir -v koli
mkdir: created directory âkoliâ
```

```
[MahfuzaICT@webminal.org mahfuza]$ls
koli  mahfuza  mahfuzatalukdar  pori  talukdar
```

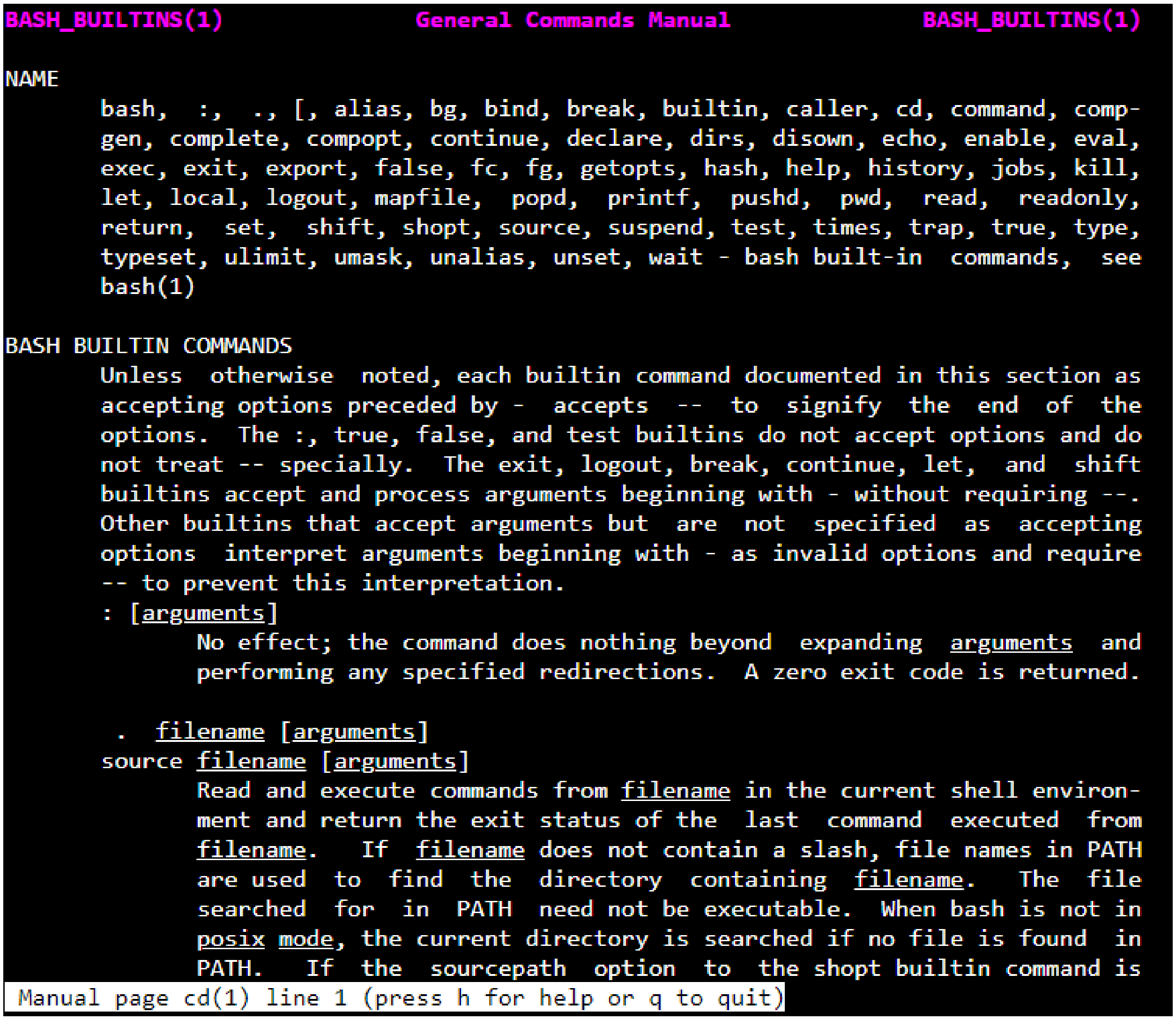
5.rm – We use the **rm** command to delete files and directories. Use "**rm -r**" to delete just the directory. It deletes both the folder and the files it contains when using only the **rm** command.

```
[MahfuzaICT@webminal.org mahfuza]$rm -r pori
[MahfuzaICT@webminal.org mahfuza]$ls
mahfuza  mahfuzatalukdar  talukdar
```

6.touch — The **touch** command is used to create a file. It can be anything, from an empty txt file to an empty zip file. For example, “**touch new.txt**”.

```
[MahfuzaICT@webminal.org ~]$ls
mahfuza  Mahfuzatalukdar  pori  talukdar
[MahfuzaICT@webminal.org ~]$touch koli.txt
[MahfuzaICT@webminal.org ~]$ls
koli.txt  mahfuza  Mahfuzatalukdar  pori  talukdar
```

7.man & --help — To know more about a command and how to use it, we use the **man** command. It shows the manual pages of the command. For example, “**man cd**” shows the manual pages of the **cd** command. Typing in the command name and the argument helps it show which ways the command can be used (e.g., **cd –help**).

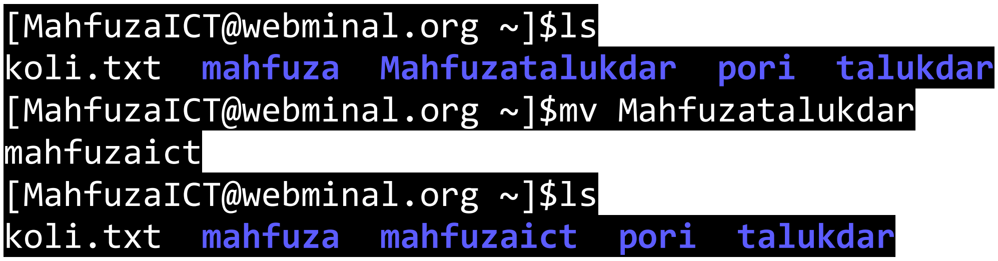


```
BASH_BUILTINS(1)              General Commands Manual              BASH_BUILTINS(1)

NAME
       bash,  :,  ., [, alias, bg, bind, break, builtin, caller, cd, command, comp-
       gen, complete, compopt, continue, declare, dirs, disown, echo, enable, eval,
       exec, exit, export, false, fc, fg, getopts, hash, help, history, jobs, kill,
       let, local, logout, mapfile,  popd,  printf,  pushd,  pwd,  read,  readonly,
       return,  set,  shift, shopt, source, suspend, test, times, trap, true, type,
       typeset, ulimit, umask, unalias, unset, wait - bash built-in  commands,  see
       bash(1)

BASH BUILTIN COMMANDS
       Unless  otherwise  noted, each builtin command documented in this section as
       accepting options preceded by -  accepts  --  to  signify  the  end  of  the
       options.  The :, true, false, and test builtins do not accept options and do
       not treat -- specially.  The exit, logout, break, continue, let,  and  shift
       builtins accept and process arguments beginning with - without requiring --.
       Other builtins that accept arguments but  are  not  specified  as  accepting
       options  interpret arguments beginning with - as invalid options and require
       -- to prevent this interpretation.
       : [arguments]
              No effect; the command does nothing beyond  expanding  arguments  and
              performing any specified redirections.  A zero exit code is returned.

        .  filename [arguments]
       source filename [arguments]
              Read and execute commands from filename in the current shell environ-
              ment and return the exit status of the  last  command  executed  from
              filename.   If  filename does not contain a slash, file names in PATH
              are used  to  find  the  directory  containing  filename.   The  file
              searched  for  in  PATH  need not be executable.  When bash is not in
              posix mode, the current directory is searched if no file is found  in
              PATH.   If  the  sourcepath  option  to  the shopt builtin command is
 Manual page cd(1) line 1 (press h for help or q to quit)
```

8.cp — We use the **cp** command to copy files through the command line. It takes two arguments: The first is the location of the file to be copied, the second is where to copy.

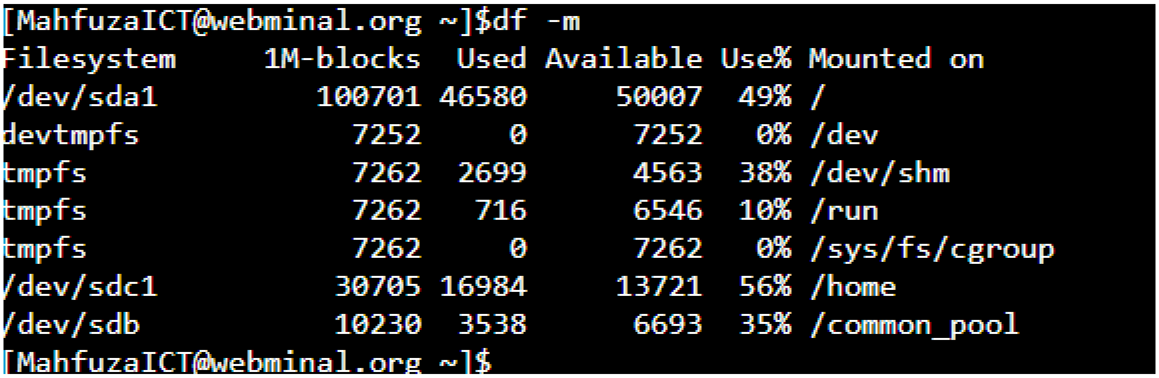
```
[MahfuzaICT@webminal.org pori]$cp new.txt
/home/MahfuzaICT/talukdar
[MahfuzaICT@webminal.org pori]$ls
/home/MahfuzaICT/talukdar
new.txt
```

9.mv — We use the **mv** command to move files through the command line. We can also use the **mv** command to rename a file. For example, if we want to rename the file “**text**” to “**new**”, we can use “**mv text new**”. It takes the two arguments, just like the **cp** command.



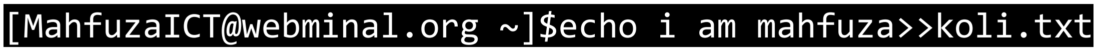
```
[MahfuzaICT@webminal.org ~]$ls
koli.txt  mahfuza  Mahfuzatalukdar  pori  talukdar
[MahfuzaICT@webminal.org ~]$mv Mahfuzatalukdar
mahfuzaict
[MahfuzaICT@webminal.org ~]$ls
koli.txt  mahfuza  mahfuzaict  pori  talukdar
```

10.df — We use the **df** command to see the available disk space in each of the partitions in our system. We can just type in **df** in the command line and you can see each mounted partition and their used/available space in % and in KBs. If you want it shown in megabytes, you can use the command "**df -m**".



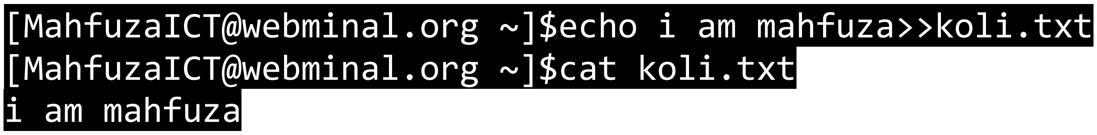
```
[MahfuzaICT@webminal.org ~]$df -m
Filesystem     1M-blocks  Used Available Use% Mounted on
/dev/sda1         100701 46580     50007  49% /
devtmpfs            7252     0      7252   0% /dev
tmpfs               7262  2699      4563  38% /dev/shm
tmpfs               7262   716      6546  10% /run
tmpfs               7262     0      7262   0% /sys/fs/cgroup
/dev/sdc1          30705 16984     13721  56% /home
/dev/sdb           10230  3538      6693  35% /common_pool
[MahfuzaICT@webminal.org ~]$
```

11.echo — The "**echo**" command helps us move some data, usually text into a file. For example, if we want to create a new text file or add to an already made text file, we just need to type in, "**echo hello, my name is AfrinZaman >> orin**". We do not need to separate the spaces by using the backward slash here, because we put in two triangular brackets when we finish what we need to write.



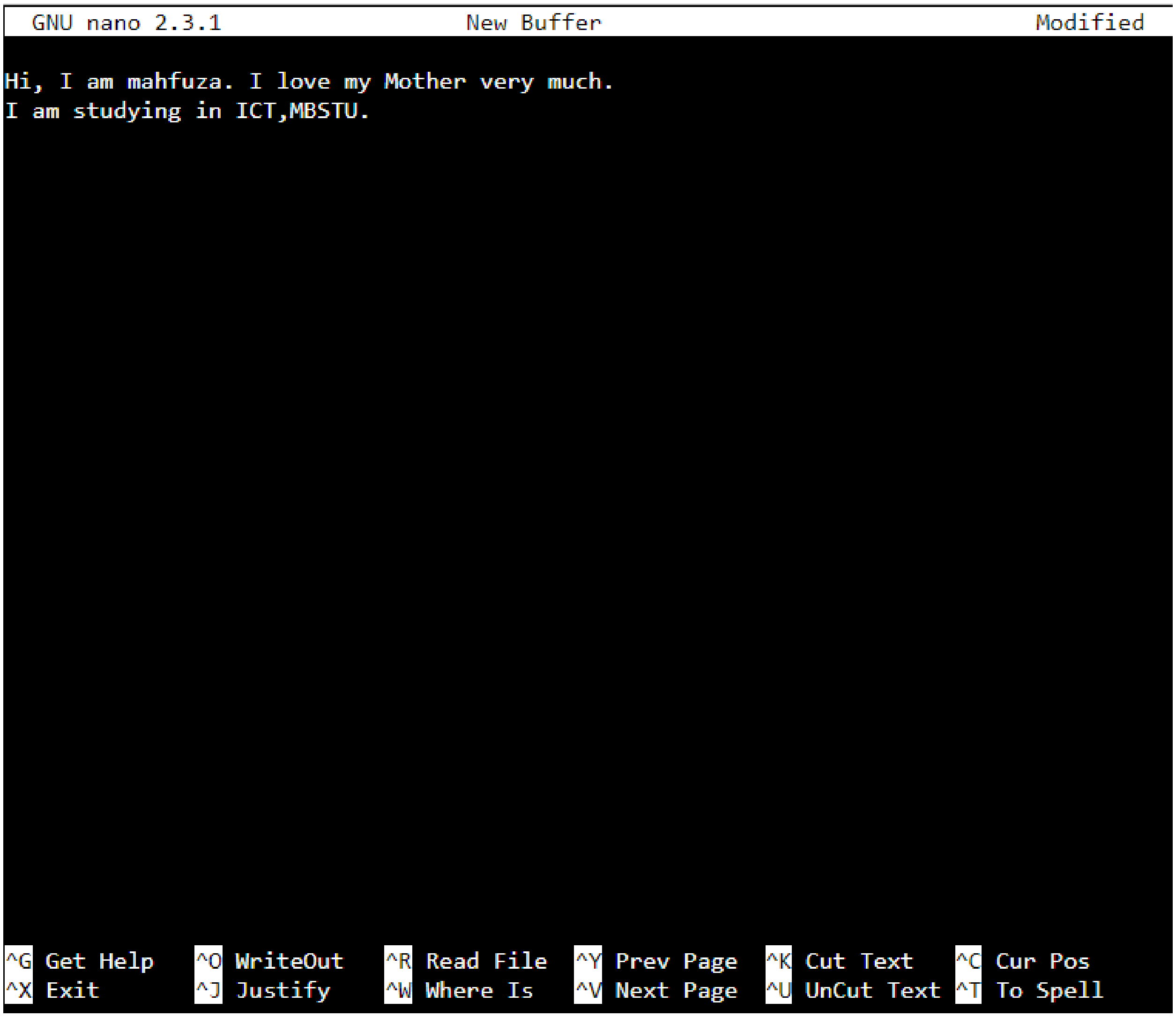
```
[MahfuzaICT@webminal.org ~]$echo i am mahfuza>>koli.txt
```

12.cat — We use the **cat** command to display the contents of a file. It is usually used to easily view programs.



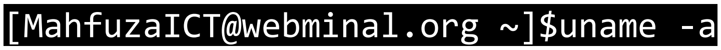
```
[MahfuzaICT@webminal.org ~]$echo i am mahfuza>>koli.txt
[MahfuzaICT@webminal.org ~]$cat koli.txt
i am mahfuza
```

13.nano, vi, jed — nano and **vi** are already installed text editors in the Linux command line. The **nano** command is a good text editor that denotes keywords with color and can recognize most languages. And **vi** is simpler than **nano**. We can create a new file or modify a file using this editor. For example, if we need to make a new file named **"check.txt**", we can create it by using the command "**nano check.txt**". We can save our files after editing by using the sequence Ctrl+X, then Y (or N for no). In my experience, using **nano** for HTML editing doesn't seem as good, because of its color, so I recommend **jed** text editor. We will come to installing packages soon.



```
  GNU nano 2.3.1                  New Buffer                          Modified

Hi, I am mahfuza. I love my Mother very much.
I am studying in ICT,MBSTU.
```

14.uname — We use **uname** to show the information about the system our Linux distro is running. Using the command "**uname -a**" prints most of the information about the system. This prints the kernel release date, version, processor type, etc.



```
[MahfuzaICT@webminal.org ~]$uname -a
```

```
Linux server-1.localdomain 3.10.0-514.16.1.el7.x86_64 #1
SMP Wed Apr 12 15:04:24 UTC 2
   6_64 x86_64 x86_64 GNU/Linux
```

15.hostname — We use **hostname** to know our name in our host or network. Basically, it displays our hostname and IP address. Just typing “**hostname**” gives the output. Typing in “**hostname -I**” gives us our IP address in our network.

```
[MahfuzaICT@webminal.org ~]$hostname
server-1.localdomain
[MahfuzaICT@webminal.org ~]$hostname -I
217.182.92.164
```

Discussion :

From this lab, we learn about linux command and know more about Linux Operating System. We use 15 linux commands in this lab and this lab have helped us a lot. We can use Linux Operating System in future.