Operating System LAB 2

Linux Shell Commands

Instructor Info.

Name: Waqas Ali

Email: waqas.ali2@uet.edu.pk

BASIC COMMANDS

How to run commands

- Search—Type Terminal
- Press Ctrl+Alt+t to open Terminal
- When you will open Terminal, you will see,

[someone]$

- One command consists of three parts, i.e. command name, options, arguments.

Example)

[someone~]$ command-name optionA optionB argument1 argument2

BASIC COMMANDS

How to run commands

- `Between` command name, options and arguments, space is necessary.

- Options always start with "-"

- Example:

```
cd ..
ls  -l
 mv fileA fileB
```

BASIC COMMANDS

File Handling Commands

- cat: After concatenation Displays a Text File content
  - $cat >filename
  - $cat>file.txt
- cp: Copy one or more files to another location
  - $cp< Source> <destination>
  - $cp file1.txt file2.txt
- mkdir: create a directory/folder ---- touch file.txt
  - $mkdir <directory name or path>
  - $ mkdir OSLab1
- rm: Deletes a File
  - $rm <file name or path>
  - $rm file.txt
- grep: Finds a String
  - Grep –r “lab”

Example Usage

Let's say want to quickly locate the phrase "**our products**" in HTML files on your machine. Let's start by searching a single file. Here, our *PATTERN* is **"our products"** and our *FILE* is **product-listing.html**.

```
user@computer: ~/documents/html
$ grep "our products" product-listing.html
<p>You will find that all of our products are impeccably designed and
meet the highest manufacturing standards available <em>anywhere.</em><
/p>
$
```

BASIC COMMANDS

File Handling Commands

- mv: move the file/files
  - $mv <source> <destination>
- head: Displays the Beginning of a File
  - $ head -1 months
- tail: Displays the End of a File
- sort: Displays a File in Order

BASIC COMMANDS

Commands

- ls show files in current position
- cd change directory
- cp copy file or directory
- mv move file or directory
- rm remove file or directory
- pwd show current directory
- mkdir create directory
- rmdir remove directory
- less, more display file contents
- man command read the online manual page for a command
- whatis give brief description of a command

BASIC COMMANDS

Commands

- Who Display login name ,date , time and terminal
- Whoami Display only the user name
- Pwd Displays the path of the current working directory
- Date Displays current time and date
- Clear Clears the terminal screen
- Echo Displays the message on screen
- Exit Exit the Shell
- Touch creates new file

BASIC COMMANDS

Commands

- `su` switch user
- `passwd` change password
- `adduser` create new user account
  - *sudo adduser username*
  - *sudo su username (to check the created user)*
- `userdel` delete user account
- `df` show disk space usage
- `shutdown` reboot or turn off machine
  - *sudo shutdown now*

Relative and Absolute path
Absolute path

- Address from the root

  /home/linux/

  ~/linux

- Similar to:

  UET/ New Campus/ CS Department/ 2021 Batch/ Section A

Relative path

- Relative to your current location

  “ `.` ” your current location

  “..” one directory above your current location

  “`pwd`” *(present working directory)*gives you your current location of working directory

- Example

  `ls ./linux` : lists the content of the dir linux

  `ls ../../` : lists everything that is two dir higher

- Similar to:

  Go Left/turn right/take the TSOL/go

Redirect, append and pipe

Redirect and append

- Output of command is displayed on screen.
- Using “>”, you can redirect the output from screen to a file.
- Using “>>” you can append the output at the end of the file.

Pipe

- Some commands require input from a file or other commands *(a mechanism for sending data from one program to another)*.
- Using “|”, you can use output from other command as input to the command.

```
fast@ubuntu:~$ ls
client        Downloads          ns-allinone-2.35          server.cpp   Videos
client.cpp    examples.desktop   ns-allinone-2.35.tar.gz   server.cpp~
client.cpp~   f                  Pictures                  $SHELL
Desktop       f.txt              Public                    #SHELL
Documents     Music              server                    Templates
fast@ubuntu:~$ ls | head -3
client
client.cpp
client.cpp~
fast@ubuntu:~$
```

PERMISSION

- All of files and directories have owner and permission.
- There are three types of permission, readable, writeable and executable.
- Permissions are given to three kinds of groups. owner, group member and others.
- r□readable, w□writable, x□executable

Example:

```
shahidost@ubuntu: ~/Desktop
shahidost@ubuntu:~/Desktop$ ls -l
total 58156
-rw-r--r-- 1 shahidost shahidost        0 Aug 17 01:02 dfasdfasd
-rw-rw-r-- 1 shahidost shahidost        4 Aug 17 05:44 file
-rw-rw-r-- 1 shahidost shahidost        4 Aug 17 05:45 file.txt
-rw-rw-r-- 1 shahidost shahidost        0 Aug 17 01:01 file.txt~
-rw-rw-r-- 1 shahidost shahidost       14 Aug 17 06:07 myoutput
-rw-r--r-- 1 shahidost shahidost 59529999 Nov 10  2014 ns-allinone-2.35.tar.gz
-rw-rw-r-- 1 shahidost shahidost     2000 May 17 21:42 TPL.c
shahidost@ubuntu:~/Desktop$
```

Example:

```
shahidost@ubuntu: ~/Desktop
shahidost@ubuntu:~/Desktop$ ls -l
total 58156
-rw-r--r-- 1 shahidost shahidost        0 Aug 17 01:02 dfasdfasd
-rw-rw-r-- 1 shahidost shahidost        4 Aug 17 05:44 file
-rw-rw-r-- 1 shahidost shahidost        4 Aug 17 05:45 file.txt
-rw-rw-r-- 1 shahidost shahidost        0 Aug 17 01:01 file.txt~
-rw-rw-r-- 1 shahidost shahidost       14 Aug 17 06:07 myoutput
-rw-r--r-- 1 shahidost shahidost 59529999 Nov 10  2014 ns-allinone-2.35.tar.gz
-rw-rw-r-- 1 shahidost shahidost     2000 May 17 21:42 TPL.c
shahidost@ubuntu:~/Desktop$
```

- In the above example the first 10 characters of the output are what we look at to identify permissions.
- The first character identifies the file type. If it is a dash (-) then it is a normal file. If it is a d then it is a directory.
- The following 3 characters represent the permissions for the owner.
- A letter represents the presence of a permission and a dash (-) represents the absence of a permission. In this example the first file owner has all permissions (read, write but not execute).
- The following 3 characters represent the permissions for the group. In this example the group has the ability to read but not write or execute.
- Note that the order of permissions is always read, then write then execute.
- Finally the last 3 characters represent the permissions for others (or everyone else). In this example they have the read permission and nothing else.

PERMISSION

Command

- `chmod` change file mode, add or remove permission
- `chown` change owner of the file

```
chown owner_name file_name
```

Example)

`chmod a+w` filename

add writable permission to all users

`chmod o-x` filename

remove executable permission from others

`chmod a+x`

Gives permission to the user to execute a file

- u□user (owner), g□group, o□others a□all

PERMISSION

1. Grant the execute permission to the group.
2. Then remove the write permission for the owner.

```
user@bash: ls -l frog.png
-rwxr----x 1 harry users 2.7K Jan 4 07:32 frog.png
user@bash:
user@bash: chmod g+x frog.png
user@bash: ls -l frog.png
-rwxr-x--x 1 harry users 2.7K Jan 4 07:32 frog.png
user@bash:
user@bash: chmod u-w frog.png
user@bash: ls -l frog.png
-r-xr-x--x 1 harry users 2.7K Jan 4 07:32 frog.png
user@bash:
```

- Read is equivalent to ‘4’.
- Write is equivalent to ‘2’.
- Execute is equivalent to ‘1’

0 – no permission

1 – execute

2 – write

3 – write and execute

4 – read

5 – read and execute

6 – read and write

7 – read, write, and execute

```
chmod 775 /path/to/file
```

1. Write the permissions you want the file to have. To make your life easier, write the permissions grouped into sets of three letters.
2. Under each letter, write a digit 1; under each dash write a digit zero. Ignore the dash at the very beginning that tells you whether it's a file or directory. This gives you three **binary** numbers.

3.

Now convert each set of three digits to a single digit using this table:

Binary	Becomes	Binary	Becomes
000	0	100	4
001	1	101	5
010	2	110	6
011	3	111	7

4. Now use that number in a chmod command to set your desired permissions on the file

PROCESS MANAGEMENT

- Process is a unit of running program.
- Each process has some information, like process ID, owner, priority, etc.

Example) Output of "`top`" command

PID	USER	PRI	NI	SIZE	RSS	SHARE	STAT	%CPU	%MEM	TIME	COMMAND
12035	nomura	15	0	1080	1080	840	R	0.3	0.2	0:00	top
1	root	15	0	472	436	420	S	0.0	0.0	0:04	init
2	root	15	0	0	0	0	SW	0.0	0.0	0:00	keventd
3	root	15	0	0	0	0	SW	0.0	0.0	0:00	kapmd
4	root	34	19	0	0	0	SWN	0.0	0.0	0:00	ksoftirqd_CPU0
5	root	15	0	0	0	0	SW	0.0	0.0	0:59	kswapd
6	root	15	0	0	0	0	SW	0.0	0.0	0:00	bdflush

PROCESS MANAGEMENT

```
top - 06:47:36 up  1:04,  2 users,  load average: 0.14, 0.05, 0.06
Tasks: 309 total,   2 running, 307 sleeping,   0 stopped,   0 zombie
%Cpu(s):  1.3 us,  0.7 sy,  0.0 ni, 97.7 id,  0.0 wa,  0.3 hi,  0.0 si,  0.0 st
KiB Mem:   1905480 total,   839216 used,  1066264 free,    58688 buffers
KiB Swap:  1955836 total,        0 used,  1955836 free.   343916 cached Mem

  PID USER      PR  NI    VIRT    RES    SHR S %CPU %MEM     TIME+ COMMAND
 1168 root      20   0  270016  34812  11596 S  0.7  1.8   0:18.22 Xorg
 1565 root      20   0  165504   4660   3744 S  0.7  0.2   0:03.53 vmtoolsd
 3638 shahido+  20   0   29264   1800   1176 R  0.7  0.1   0:00.67 top
 3139 root      20   0       0      0      0 S  0.3  0.0   0:00.24 kworker/u128+
 3566 shahido+  20   0  583864  19540  13060 S  0.3  1.0   0:00.93 gnome-termin+
    1 root      20   0   33772   3136   1456 S  0.0  0.2   0:01.44 init
    2 root      20   0       0      0      0 S  0.0  0.0   0:00.01 kthreadd
    3 root      20   0       0      0      0 S  0.0  0.0   0:00.01 ksoftirqd/0
    4 root      20   0       0      0      0 S  0.0  0.0   0:00.01 kworker/0:0
    5 root       0 -20       0      0      0 S  0.0  0.0   0:00.00 kworker/0:0H
    7 root      20   0       0      0      0 S  0.0  0.0   0:00.85 rcu_sched
```

- **Line 1-2** Tasks is just another name for processes. It's typical to have quite a few processes running on your system at any given time. Most of them will be system processes. Many of them will typically be sleeping. This is ok. It just means they are waiting until a particular event occurs, which they will then act upon.

PROCESS MANAGEMENT

- **Line 3** CPU information.
- **Line 4** This is a breakdown of working memory (RAM). Don't worry if a large amount of your memory is used. Linux keeps recently used programs in memory to speed up performance if they are run again. If another process needs that memory, they can easily be cleared to accommodate this.
- **Line 5** This is a breakdown of Virtual memory on your system. If a large amount of this is in use, you may want to consider increasing it's size. For most people with most modern systems having gigabytes of RAM you shouldn't experience any issues here.
- **Lines 6 - ---** Finally is a listing of the most resource intensive processes

PROCESS MANAGEMENT

Commands

- `kill` Stops a program. The program is specified by process ID.
- `killall` Stops a program. The program is specified by name.
- `ps` Shows process status
- `top` Shows system usage statistics

https://linoxide.com/linux-command/essential-linux-basic-commands/

https://linoxide.com/linux-how-to/linux-commands-brief-outline-examples/

https://www.maketecheasier.com/file-permissions-what-does-chmod-777-means/