

10.29.10.15

|| |

Getting started with Linux basic commands for directory operator display directory structure in tree format

'ls'

ls -a : list all files / dir including hidden ones

ls -A : similar to ls -a but skips current and previous directory.

ls -r : display files / directory in reverse order

-t : sorts files based on modified time

-R : lists sub directories recursively

-i : display inode number

-l : prints detailed information

-hide : exclude user defin file in output

'cd'

cd ~ : quick switch to home dir

.. : navigate to parent directory

cd <pathnames> : navigates to path name

'pwd'

print working directory

'ps'

Display info of running process.

21

Linux commands for operations such as redirection pipe filter, job controls, changing ownerships/permissions of files/links/directory

REDIRECTION

Redirection: features in linux, when executing a command you can change standard input/output devices.

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

Eg: `ls -al > listings`

Here the output of command `ls -al` is redirected to file `listing` instead of your screen

`cat listing` provide the output

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

Eg: Mail program in Linux can help you send emails from the terminal
`mail -s "Subject" to -address < filename`
This would attach the file with the email and it would be sent to the recipient

File Descriptor (FD): Every file has an associated ~~no~~ number called FD.

STDIN FD= 0

STDOUT FD= 1

STDERR FD= 2

Error Redirection: Error message can be redirected to a file

Eg: `myprogram 2> crashfile`

`2>` we redirect the error output to a file named `errorfile`

②

Pipes

Pipe command in Linux that lets you use two or more commands such that output of one command serves as input to the next. In short output of each process directly as input to the next one like a pipeline. Symbol is '|'

Eg:

cat filename | less

Allows you to show you only one scroll length of content at a time

cat filename | pg or more

Scroll down by simply hitting enter key.

Grep

It will scan the document for the desired information and present the result in format you want

grep search_string

cat filename | grep search_string

Options

- v Shows all lines that do not match the search string
- c Display only the count of matching lines
- n Shows the matching line and its number
- i Match both cases
- l Just shows the file with the string

Sort

Sort Filename

0 -07
1 -07

- r Reverse sorting
- n Sort numerically
- f Case insensitive sort

③

Filters

Lot of filter commands
awk, grep, sed, spell and awk

AWK

Is a remarkable pattern scanning and processing language, it can be used to build useful filters in linux.

Sed

- SED is powerful text stream editor. Can do insertion, deletion, search and replace (substitution).
- SED command in unix supports regular expression which allows it perform complex pattern matching

REPLACE

Sed command mostly used to replace text in a file

replace "unix" with "linux"

sed 's/unix/linux' <file name>

↳ substitution operation

'/' delimiter.

DELETE line

sed 'Nd' filename. N is number. \$ for last line

Range n to y 'Nc, yd'

SPELL

Spell is very minimalist spell-checking programs. Reads the file word by word and reads content against the dictionary.

~~spell~~ spell [option] FILE

JOB CONTROL

Sur kill using kill -9

To start job sh &

④

Change ownership

Chown command lets you change the file ownership and group through the CLI

chown [OPTION] . . . [OWNER]:[GROUP]

Eg:

chown root <filename>

Ownership has now been changed to 'root'.

For changing group use ':root' instead of 'root', for changing both use 'root:root'

- Change the owner~~ship~~ and group (or both) after checking ~~is~~ existing owner/group

chown --from=[curr-owner]:[curr-group]

[new-owner]:[new-group] [filename]

Eg:

chown --from root:knd knd:root filename

- Pick ownership information from a reference file

chown --reference=[file-name1] [filename2]

This will copy the owner and group info from file-name1 to filename2

- R → Recursively apply on all files and folder in

- We can use UID or GID instead of user and group name without any change in syntax.

- For verbose output use -v

⑤

13]

ADVANCED LINUX COMMANDS

curl, wget, ftp, ssh, and grep.

curl

Curl command allows you to download as well as upload data through CLI

curl [option] [curl]

Eg:- curl http://releases.ubuntu.com/18.04-desktop
-and 64.iso

◦ -O or --output <file> writes output to <file> instead of stdout

◦ Make curl use same download file name

curl -O [curl]

◦ Download multiple files

curl -O [url1] -O [url2] ...

◦ If we want curl to follow redirect use -L

◦ Resume Download from a point of interruption -C

wget

Wget is a CLI tool that allows for non-interactive download files from the internet. By non-interactive, it means that utility can work in background while user is not logged on. Supported protocols: http, https, ftp

wget [curl]

◦ Resume a download -c

◦ Download with different file name

wget -O [file-name] [curl]

◦ Redirect your wget output to log file

wget -o [log-file] [curl]

◦ Limit download speed

→ 50k

- ⑥
 - Read time out
- read-timeout=[time in seconds]
 - Number of bytes (interruption allowed)
- t, [no of bytes you want]
 - Display debug info
wget -debug [URL]
 - Change the download progress meter.
wget --progress=dot [URL]
~~dot, dotdotdot~~ options available on progress:
dot or bar
 - Displaying size in binary/mega/giga.
wget --progress=dot:binary [URL]

FTP

- ① Establish connection

ftp domain-name

ftp ip-addr

- ② Login with username and password

- ③ Working directory → ls

Change cd

- Download file

lcd /home/user/directory

If you don't specify the download location the file will be downloaded to the current directory where you were at the time you started FTP session
get file

- Upload file

put file

Ⓐ When the file that you want to upload is not in local disk you can use absolute path with /
put /path /file

④ Close ftp
bye or exit or quit

• SSH

SSH command provide a secure encrypted connection (RSA) between two host over insecure network.

SSH commands

ssh-keygen - creates a key pair for public key auth

ssh-copy-id - config public key as auth on a server

ssh-agent - agent to hold private key for single Ssh user

ssh-add - tool to add key to the agent

SCP - file transfer client SCP like command line interface

SFTP - file transfer client with SFTP like CLI

sshd - open SSH server

Connection

SSH sample.ssh.com

SSH alternative-username@sample.ssh.com

SSH -l alternative-username@sample.ssh.com

Execute Remote commands

SSH run remote command

SSH sample.ssh.com ls /tmp/idea

Options

-1 use protocol version 1 only

-4 use IPv4 address only

-6 use IPv6 address only

- ③
 - a enable forwarding of the auth agent connection.
 - a disable forwarding auth agent connection.
 - C use data compression
 - c cipher spec select the cypher specification
 - D [bind_address:] port
 - E log_file Debug logs to log_file
 - F config_file specifies a per-user config file
 - g Allows remote host to connect to local forward port.
 - i identity_file file from which the identity key for public key auth
 - J [user@] host[:port]
 - l login name
 - p port to connect to on remote host
 - q quiet mode
 - V display version number
 - v verbose
 - X Enable X11 forwarding

grep

Page 2 of this note book

| EXP 4 |

Shell programming: write shell script to show various system configuration

- Current login user and his login name
- Your current shell
- Your home directory
- Your OS name
- Current path setting
- Your current working directory
- Number of users currently logged in

```
#!/bin/bash
$ LOGNAME
echo "User is $USER username is $CURRENTNAME
echo "Your current shell is $SHELL"
echo "Your home directory is $HOME" NAME
echo "Your OS Name is $hostnamet NAME"
echo "Current path setting $PATH" $OSTYPE
echo "Your current working dir $PWD
echo "Number of users logged in" who
```

OUTPUT

User name is student username is student

Your current shell is \$ /bin/bash

Your home directory is \$home/student

Your OS Name is : linux-gnu

Current path setting is PATH: /usr/local

Current working directory \$ /home/student

Number of logged on user: 1 user(s)

EXP 5

Write shell script to show various system config like

- your OS version, release number, kernel
- All available shells
- Computer CPU info like processor type, speed
- Memory info
- hard disk info like size, hard disk, cache memory, model
- file system (mounted)

```
#!/bin/bash
```

```
echo "OS `cat /etc/os-release`"
```

```
echo "Available shells `cat /etc/shells`"
```

```
echo "Memory info `cat /proc/meminfo`"
```

```
echo "file mounted `cat /proc/mounts`"
```

```
echo "hard disk info `lsblk`"
```

OUTPUT (Full output is 200+ lines)

NAME="Ubuntu"

VERSION="12.04.5 LTS, Precise Pangolin"

ID=ubuntu

*

*

*

cgroup /sys/fs/cgroup/memory cgroup rw,
relatime, memory 0 0

cgroup /sys/fs/cgroup/freezer cgroup rw, rdonly
freezer 0 0

EXP 6

Calculator

```
#!/bin/bash
y=0
while [ "$y" -eq 0 ]
do
    echo " Enter operand 1 "
    read a
    echo " Enter operand 2 "
    read b
    x=0
    while [ $x -eq 0 ]
    do
        echo " 1. +"
        echo " 2. -"
        echo " 3. *"
        echo " 4. /"
        echo " 5. %"
        read e
        case $e in
            1) echo $(expr $a + $b);;
            2) echo $(expr $a - $b);;
            3) echo $(expr $a * $b);;
            4) echo $(expr $a / $b);;
            5) echo $(expr $a % $b);;
        esac
    done
done
```

```
        esac  
        echo "press 0 to continue"  
        echo "press 1 to exit"  
        read x  
        done.  
        echo "press 0 to continue"  
        echo "press 1 to exit"  
        read y  
        done
```

OUTPUT

Enter operand 1
1

Enter operand 2

1. +

2. -

3. *

4. /

5. %

6.

7. 3

Press 0 to continue

Press 1 to exit

0

Enter operand 1
3

Enter operand 2

6

1. +

2. -

3. *

4. /

5. %

6.

7. 8

Press 0 to continue

Press 1 to exit

1

[EXP 7]

Write a script called addnames that is to be called as follows

• /addnamee *ulist* *username*

Here *ulist* is - the name of the file contains the list of usernames and *username* is particular student's name. The script should

- Check that correct number of arguments was received and print a message in case the ~~file~~ number of arguments is incorrect
- Check whether the *ulist* file exist and print an error message if it does not
- check whether the *username* already exist in the file. If the user name exist, print a message stating that the name already exist. Otherwise add the username to the end of the list.

SCRIPT

```
#!/bin/bash
if [ $1 #eq 2 ]
then
    if [ -f $1 ]
    then
        val=`cat $1 | grep $2`
        if [ -z $val ]
        then
            echo "No Such user in file. Adding new user..."
            echo $2 >> $1
            echo "Added"
        fi
    fi
fi
```

```
else
    echo "User already exists"
fi
else
    echo "No such file"
fi
else
    echo "Two arguments expected"
fi
```

OUTPUT

\$ cat temp.txt

Kowsik

Ajay

Naraneeth

\$ bash addname.sh

Two arguments expected

\$ bash addname.sh t.txt Ganesh

No such file

\$ bash addname.sh temp.txt Ganesh

No such user in file. Adding username ..

Added

\$ bash addname.sh temp.txt Ganesh

User already exists

\$ cat temp.txt

Kowsik

Ajay

Naraneeth Ganesh

Shell script to print ~~first~~ first 15 digit in fib series.

#!/bin/bash

ASSIGNMENT
QUESTION 1

a=0

b=1 echo \$a echo \$b

c=\$((\$a + \$b))

echo \$c

for i in {1..15}

do

\$a=\$b

b=\$c

~~c=\$a~~

c=\$((\$a + \$b))

echo \$c

done

OUTPUT

1	144
2	233
3	377
5	987
8	2867
13	
21	
34	
55	
89	

✓ Verified
@eg

Calculate the sum of two array.

ASSIGNMENT
QUESTION 2

$i = 0$

echo "List 1"

$\text{while } [\$i < \text{lt } \$1]$

do

 read a

$l[\$i] = \a

$i = \$[\$i + 1]$

done

$m = ()$

echo "List 2"

$\text{while } [\$i < \text{lt } \$1]$

do

 read a

$m[\$i] = \a

$i = \$[\$i + 1]$

$\text{done}.$

$s = ()$

$i = 0$

$\text{while } [\$i < \text{lt } \$1]$

do

$s[\$i] = \$[\$\{ l[\$i] \} + \$\{ m[\$i] \}]$

$i = \$[\$i + 1]$

done

$\text{echo "Sum = \$\{ s[\$i] \} "}$

OUTPUT

List 1

1
2
3

List 2

1
2
3

$$S_{nm} = 2 \cdot 4 \cdot 6$$

✓ O/P Verified
@Raj

Exp 8

8. Version Control System setup & using GIT. Try the following.

Creating repository

→ Local

To create a local repository we need to initialize by using

`git init [repository name]`
command.

→ Remote (GitHub)

github.com/new fill the fields and hit create button

Checking out Repository

`git -b <branch-name>`

If the branchname doesn't exist it will create new branch in that name.

Adding content to repository

`git add .` → used to add all files & folders

`git add filename` → just a file.

Committing the data to a repository

`git commit -m "Commit message"`

Update the local copy

① update your local repo from central repo

`git pull upstream <branch-name>`

② Making edits & save

`git add .` `git commit -m "`

③ Push changes from local to github
git push origin <branch-name>

Compare different versions

git diff [--option] <commit> <commit> <path>

Eg:-

git diff HEAD^n HEAD main.c

git diff HEAD^n..HEAD --main.c

git diff HEAD~2 HEAD--main.c

} Same meaning
check 2 vers.
behind.

Conflict & conflict resolution

If 2 users edit to same line and push the git recognize as conflict. Conflict resolution can be done in 2 ways.

- ① ~~github~~ Place a git merge request if there is a conflict. It shows resolve button and the conflict is shown like

<<< HEAD

===== } body

==== }

>>> ~~branch~~ branch -q

use resolve editor coming online and decide whether to keep the line above or below ==

Then delete << and >>

Then click the button resolved.

- ② If offline
make merge request

git fetch

git merge

if any error regarding the conflict open that file in
text editor and decide the lines to keep above or
below === then,

git add .

git commit

| EXP 9 |

Shell script which starts on system bootup and kills every process which uses more than specified amount of memory or CPU

Script file

```
#!/bin/bash
```

```
¶
```

```
LIMIT=1234
```

```
# Temporarily set the limit as 1234kbytes.
```

```
pss -eo rss=, pid=, user=, com= k -rss |  
while read size pid user com  
do
```

```
if [ $user = $USER ]
```

```
then
```

```
# Only kill process that are under this user
```

```
if [ $size >= LIMIT ]
```

```
then
```

```
kill -9 $pid
```

```
echo "Process $pid was killed"
```

```
fi
```

```
fi
```

```
done
```

To run on every reboot

Method 1

Using Cron

```
sudo nano /etc/crontab
```

Add the following line

@reboot /home/user/script.sh

Method 2

Put the file inside /etc/init.d/

chmod +x /etc/init.d/start-my-app

Sometime need to add symlink /etc/rc.d/

ln -s /etc/init.d/start-my-app /etc/rc.d/