Shell Scripting

S.Lochani Vilehya
Emp id : 43317

Shell Script

- It is a list of commands in computer program that is run by unix shell which is a command line interpreter
- The different operations performed by shell scripts are program execution , file manipulation and text printing
- The shell is a program that takes commands from the keyboard and gives them to the operating system to perform
- How to check types of shells available in our systems : **cat /etc/shells**

vlab@HYVLAB7:~/lochu/bash_script$ cat /etc/shells
/etc/shells: valid login shells
/bin/sh
/bin/bash
/usr/bin/bash
/bin/rbash
/usr/bin/rbash
/bin/dash
/usr/bin/dash
/usr/bin/xonsh

Shebang : #!/bin/bash

- This #! Is called shebang or shebang.
- The shebang is combo of the #(pound key) and !(exclamation mark)
- It is used to specify the interpreter with which the given script will be run by default

Access name of the shell and available shells

To get the shell name we are using : **echo $SHELL**

vlab@HYVLAB7:~/lochu$ echo $SHELL

/bin/bash

If you are not using bash shell then find its path where bash interpreter is located and use it

vlab@HYVLAB7:~/lochu$ which bash

/usr/bin/bash

vlab@HYVLAB7:~/lochu$ /usr/bin/bash

How to run first script in linux?

- Create script
  - **$ gvim demo.sh**
- Add shebang line and code to demo.sh :
  - **#!/bin/bash echo “hello”**
- Add execute permissions to file
  - **chmod +x demo.sh**
- Run the shell script
  - **./demo.sh**

Variables

- It is a character string in shell that stores some value
- It could be integer , filename, string or shell command itself
- Assigning value to variable uses the assignment operator **(=)**
- Syntax : **name = “value”**
- To print the variable , prefix the variable name with $
- Shell scripting supports special variables such as **$0**, which refers to name of the script
- **$1,$2,$3,..** etc are first,second,third… command line arguments
- **$#** gives the number of command line arguments passed to script

if-else command

- It also possible to use elif(else if) statement which allows you to chain several together .
- **Syntax:**

```
if [cond1]; then
    Commands
elif [cond2]; then
    Commands
else
    Commands
fi
```

How to install software in multiple types of OS?

```
#!/bin/bash

echo "script to install git"
echo "installation started"

# vlab ALL=(ALL) NOPASSWD: /usr/bin/apt,
/usr/bin/apt-get
#add the above line for passwordless
installation through sudo


if [ "$(uname)" = "Linux" ]; then
     echo "linux installing git.."
     sudo apt install git -y
elif [ "$(uname)" = "Fedora" ]; then
     echo "fedora installing git..."
     sudo dnf install git
else
     echo "not installing"
fi
```

Use **“uname”** command to the type of OS you are using for easy installation of packages with different command sin different linux distributions

vlab@HYVLAB7:~/lochu/bash_script$ uname

Linux

Read a file content

- Read a file content by opening it
  - vi/vim/nano editors
- Read a file content without opening it
  - cat , less , more
- Read a file content with conditions
  - more ,tail ,grep ,awk ,sed
- Read a file content using more
  - **more -n filename.txt**
    - Displays text upto specified line
  - **more +n filename.txt**
    - The text is displayed from specified line

Contd…

- Read content of file using head
  - **head filename**
    - By default it displays top 10 lines of file
  - **head -n filename**
    - Displays the top n number of lines of the file
- Read content of file using tail
  - **tail filename**
    - By default displays last 10 lines of file
  - **tail -n filename**
    - Displays the last n number of lines of file

awk command

- It is used for processing or analyzing test or data files which are organized by lines and columns
- Simple awk commands syntax:
  - **awk [options] '[selection_criteria] {action}' input-file**
  - cat input-file | awk [options] '[selction_criteria] {action}' input-file
  - Awk can take the following options
    - -F fs : to specify a field separator
    - -f file : to specify a file that contains awk script
    - -v var = value : to declare a variable
  - Selection criteria : pattern/condition
  - Action: it is a logic to perform action on each row/record

Check Disk utilization

```
#!/bin/bash

echo "check disk usage in Linux system"
disk_size=`df -h|grep '/dev/sda1'| awk
'{print $5}'|cut -d '%' -f1`
echo "$disk_size% is disk in root is
filled"
if [ $disk_size -gt 80 ]; then
     echo "disk is almost full!!..delete
files soon"
else
     echo "enough disk is available"
fi
```

vlab@HYVLAB7:~/lochu/bash_script$ sh disk_use.sh
check disk usage in Linux system
37% is disk in root is filled
enough disk is available

du command

- This command is a standard Linux/Unix command that allows user to get disk usage information quickly
- Flags:
  - -h : prints size outputs
  - -a : list the sizes of all files and directories in given file path
- Usage:
  - du -ah /tmp : it list the size of all files directory in path /tmp in human readable format

sort command

- Used to sort a file, arranging the records in a particular order
- Flags
  - -n : sort the file in numerical order (small to big)
  - -r : print the output in reverse order
  - -h : human numeric sort
  - -hr : combo of -h and -r flags

Command line arguments

- The arguments are parameters that are passed to a script while executing the, in the bash shell .
- They are also called positional parameters in linux

How to access arguments?	Description
$0	Script name
$1	First parameter of the script
$@	Complete list of arguments
$#	Total number of parameters
$$	Process id of the script
$*	Get all the arguments as double quoted
$?	Exit code for the script

Find the first 10 biggest file in file system and write the output to a file

```
#!/bin/bash

# Check if the path argument is provided
if [ -z "$1" ]; then
    echo "Please provide a directory path as the
first argument."
    exit 1
fi

echo "To get the first 10 biggest files in the
filesystem passed via positional arguments"

# Correct the assignment without spaces around
the '='
path="$1"

echo "The list of big files in filesystem $path"
du -ah "$path" | sort -hr | head -n 10
```

vlab@HYVLAB7:~/lochu/bash_script$ sh filesys_size.sh ~/lochu
To get the first 10 biggest files in the filesystem passed via positional arguments
The list of big files in filesystem /home/vlab/lochu
389M /home/vlab/lochu
373M /home/vlab/lochu/Github_Actions2.0/.git/objects/pack/pack-5aa49bdf32ec458a4e0d57ba07b558013fc39295.pack
373M /home/vlab/lochu/Github_Actions2.0/.git/objects/pack
373M /home/vlab/lochu/Github_Actions2.0/.git/objects
373M /home/vlab/lochu/Github_Actions2.0/.git
373M /home/vlab/lochu/Github_Actions2.0
8.8M /home/vlab/lochu/mynewenv
6.5M /home/vlab/lochu/mynewenv/lib/python3.9/site-packages
6.5M /home/vlab/lochu/mynewenv/lib/python3.9
6.5M /home/vlab/lochu/mynewenv/lib

find and mtime command

- It is used to find files and directories and perform subsequent operations on them
- It supports searching by file , folder, name, creation, date, modification date, owner and permissions
- **Usage:**
  - **find . filename :** find file with name filename in current working directory
  - We can specify any other location also in place of “.”

- Modified timestamp (mtime) indicates the last time the contents of a file were modified. For example , if new contents were added , deleted , or replaces in a file , the modified timestamp is changed
- +n for greater than n , -n for less than n , n for exactly n
- **-mtime +30** : get the files greater than 30 days

Contd..(demo)

- As we may not have 30 days old file to create use touch command
  - **touch -amt 201512180130.09 file1.txt**
  - -a = accessed -m = modified -t = timestamp
  - Use [[CC]YY]MMDDhhmm[.ss] time format
- To delete files that are 30 days old
  - **find . -mtime +30 -delete**

Delete older logs/files

```
#!/bin/bash
echo "to delete files which are older than
30 days"
path="$1"
echo $path
find $path -mtime +30 -delete

if [ $? -eq 0 ]; then
     echo "files are successfully deleted"
else
     echo "deletion have some issues"
fi
```

vlab@HYVLAB7:~/lochu/bash_script$ touch -amt 201512180130.09 file1.txt
vlab@HYVLAB7:~/lochu/bash_script$ ls -ltr
total 20
-rw-rw-r-- 1 vlab vlab 0 Dec 18 2015 file1.txt
-rwxrwxr-x 1 vlab vlab 20 Oct 15 14:02 myscript.sh
-rwxrwxr-x 1 vlab vlab 36 Oct 15 14:07 date.sh
-rw-rw-r-- 1 vlab vlab 409 Oct 15 14:47 install.sh
-rw-rw-r-- 1 vlab vlab 289 Oct 15 15:41 disk_use.sh
-rw-rw-r-- 1 vlab vlab 403 Oct 15 16:10 filesys_size.sh
vlab@HYVLAB7:~/lochu/bash_script$ gvim delete_old_files.sh
vlab@HYVLAB7:~/lochu/bash_script$ cd ..
vlab@HYVLAB7:~/lochu$ sh bash_script/delete_old_files.sh bash_script/
to delete files which are older than 30 days
bash_script/
files are successfully deleted
vlab@HYVLAB7:~/lochu$ ls bash_script/
date.sh disk_use.sh install.sh
delete_old_files.sh filesys_size.sh myscript.sh

wget command

- It is a free utility for non-interactive download of files from the web
- It supports HTTPS,HTTP and FTP protocols
- It is non-interactive means it can work in the background while the user is not logged on , which allows you to start a retrieval and disconnect from the system , letting wget finish the work
- **Syntax:**
  - **wget http://google.com**

Use for loop to go over directories and delete specified file

```
#!/bin/bash
FILE_NAME="file.txt"

# Loop through all files found in the
current directory
for file in $(find . -type f -name
"$FILE_NAME"); do
    # Check if the current file exists
    if [ -f "$file" ]; then
        echo "$file exists and deleting
it"
        rm -rf "$file"  # Delete the
current file
    fi
done
```

vlab@HYVLAB7:~/lochu/bash_script$ touch file.txt

vlab@HYVLAB7:~/lochu/bash_script$ sh loop.sh

./file.txt exists and deleting it

vlab@HYVLAB7:~/lochu/bash_script$ ls

delete_old_files.sh loop.sh

Docker service

- To view the running status of docker use the command
  - **$ systemctl status docker**

```
● docker.service - Docker Application Container Engine
     Loaded: loaded (/lib/systemd/system/docker.service; enabled; ven>
     Active: active (running) since Wed 2024-10-16 10:54:15 IST; 2s a>
TriggeredBy: ● docker.socket
       Docs: https://docs.docker.com
   Main PID: 351934 (dockerd)
      Tasks: 10
     Memory: 115.7M
```

- To stop the docker services if it disabled use the command
  - **$ systemctl stop docker**
- To restart the stopped docker services
  - **$ systemctl start docker**

Checking the status of the docker service

```
#!/bin/bash

echo "====status check docker
service===="
status="`systemctl status docker | awk
'NR==3 {print}' | cut -d ':' -f 2 | cut
-d '(' -f 1`"
echo $status

if [ $status = "active" ]; then
     echo "docker is running fine..."
else
     echo "docker service is not
running..."
fi
```

- Here NR in the awk command is “Number of records”

vlab@HYVLAB7:~/lochu/bash_script$ sh docker_service.sh

====status check docker service====

active

docker is running fine...

What is Crontab ?

- It is a list of commands that you want to run on a regular schedule , and also the name of the command used to manage that list
- Crontab stands for “cron table”, because it uses the job scheduler cron to execute tasks
- Cron itself is named after “chronos,the greek word”
- Cron is system process which will automatically perform tasks for you according to set schedule
- Ex : MIN HOUR DOM MON DOW CMD
- **Commands:**
  - **crontab -e :** enter this command to add the script and fields of time when you want to schedule the run
  - **crontab -l :** to list the content of the crontab created

Crontab Fields and Allowed Ranges (Linux Crontab Syntax)

Field	Description	Allowed Value
MIN	Minute field	0 to 59
HOUR	Hour field	0 to 23
DOM	Day of Month	1-31
MON	Month field	1-12
DOW	Day Of Week	0-6
CMD	Command	Any command to be executed.

Schedule a job automatically to run at regular intervals and check if docker service is down , if it's down start the service

```
#!/bin/bash
# Log file path
LOGFILE="/home/vlab/lochu/bash_script/log.txt"

# Log the current date and time
{
    echo "$(date): Starting docker_service.sh"
    echo "==== Status Check Docker Service ===="

    # Check Docker service status
    status=$(systemctl is-active docker)
    echo "Docker service status: $status"

    if [ "$status" = "active" ]; then
        echo "Docker is running fine..."
    else
        echo "Docker service is not running..."
        echo "Starting Docker service..."

        # Start Docker using sudo with the -S option
        echo "Welcome@1234" | sudo -S /bin/systemctl start docker

        # Check if the service started successfully
        if [ $? -eq 0 ]; then
            echo "Docker service started successfully."
        else
            echo "Failed to start Docker service."
        fi
    fi

    echo "$(date): Finished docker_service.sh"
} >> "$LOGFILE" 2>&1
```

vlab@HYVLAB7:~/lochu/bash_script$ crontab -e
crontab: installing new crontab
vlab@HYVLAB7:~/lochu/bash_script$ crontab -l
* * * * * /bin/bash /home/vlab/lochu/bash_script/docker_service.sh >> /home/vlab/lochu/bash_script/log.txt 2>&1
vlab@HYVLAB7:~/lochu/bash_script$ systemctl stop docker
vlab@HYVLAB7:~/lochu/bash_script$ systemctl status docker
● docker.service - Docker Application Container Engine
Loaded: loaded (/lib/systemd/system/docker.service; enabled; ven>
Active: **inactive (dead)** since Wed 2024-10-16 14:15:14 IST; 3s ago

vlab@HYVLAB7:~/lochu/bash_script$ cat log.txt
==== Status Check Docker Service ====
Docker service status: inactive
Docker service is not running...
Starting Docker service...
[sudo] password for vlab: Docker service started successfully.
Wednesday 16 October 2024 02:17:14 PM IST: Finished docker_service.sh

vlab@HYVLAB7:~/lochu/bash_script$ systemctl status docker
● docker.service - Docker Application Container Engine
Loaded: loaded (/lib/systemd/system/docker.service; enabled; ven>
Active: **active (running)** since Wed 2024-10-16 14:17:14 IST; 3min>

Read command

- Bash read command with many options to control the user input.
- Some options do not require additional parameters ,while others have mandatory parameters
- **Syntax :** read <options> <arguments>

The table below shows all the possible command options and their description.

Option	Description
`-a <array>`	Assigns the provided word sequence to a variable named `<array>`.
`-d <delimiter>`	Reads a line until the provided `<delimiter>` instead of a new line.
`-e`	Starts an interactive shell session to obtain the line to read.
`-i <prefix>`	Adds initial text before reading a line as a prefix.
`-n <number>`	Returns after reading the specified number of characters while honoring the delimiter to terminate early.
`-N <number>`	Returns after reading the specified number of chars, ignoring the delimiter.
`-p <prompt>`	Outputs the prompt string before reading user input.
`-r`	Disable backslashes to escape characters.
`-s`	Does not echo the user's input.
`-t <time>`	The command times out after the specified time in seconds.
`-u <file descriptor>`	Read from file descriptor instead of standard input.

Read the user input and print the status of service in system

```
#!/bin/bash

echo "----------welcome to service
status check-----------"

read -p "enter the service name to
check its status: " service_name

if [ -z $service_name]; then

    echo "please enter a valid service
name"

else

    systemctl status $service_name

fi
```

vlab@HYVLAB7:~/lochu/bash_script$ sh while_read.sh
----------welcome to service status check-----------
enter the service name to check its status:
please enter a valid service name
vlab@HYVLAB7:~/lochu/bash_script$ sh while_read.sh
----------welcome to service status check-----------
enter the service name to check its status: docker
while_read.sh: 4: [: missing]
● docker.service - Docker Application Container Engine
Loaded: loaded (/lib/systemd/system/docker.service; enabled; ven>
Active: **active (running)** since Wed 2024-10-16 14:17:14 IST; 2h 9>
TriggeredBy: ● docker.socket
Docs: https://docs.docker.com
Main PID: 360111 (dockerd)
Tasks: 10
Memory: 56.1M

grep command

- It is used to display only the matched pattern : by default , grep displays the entire line which has the matched string
- We can used different options to print the matched lines in different manner

```
Options Description
-c : This prints only a count of the lines that match a pattern
-h : Display the matched lines, but do not display the filenames.
-i : Ignores, case for matching
-l : Displays list of a filenames only.
-n : Display the matched lines and their line numbers.
-v : This prints out all the lines that do not matches the pattern
-e exp : Specifies expression with this option. Can use multiple times.
-f file : Takes patterns from file, one per line.
-E : Treats pattern as an extended regular expression (ERE)
-w : Match whole word
-o : Print only the matched parts of a matching line,
 with each such part on a separate output line.

-A n : Prints searched line and nlines after the result.
-B n : Prints searched line and n line before the result.
-C n : Prints searched line and n lines after before the result.
```

Finding errors in logs using grep

```
error_file=`cat ./log.txt`

matched_error=`grep -i Sorry ./log.txt`

echo $matched_error

if [ $? -eq 0 ]; then

     echo "something went wrong:
$matched_error"

else

     echo "no error in log"

fi
```

vlab@HYVLAB7:~/lochu/bash_script$ sh error_grep.sh

[sudo] password for vlab: Sorry, try again.

something went wrong: Sorry, try again.

sed command

- Sed stands for stream editor
- Sed command performs lot of functions like:
  - Viewing file contents
  - Searching
  - Find and replace
  - Insertion and deletion
- Sed also supports regular expressions which allows it performs complex pattern matching
- Sed can perform any operations on file without opening the file
- **Syntax :**
  - **sed [options] commands [file-to-work-with-sed]**

Viewing file contents using sed command

- **sed ‘’ file :** prints contents in file
- **sed ‘-p’ file :** prints the content in file twice (once by sed command other by the use of -p option)
- **sed -n ‘p’ file :** this will ignore 2 times print
- **sed -n ‘$p’ file :** this will print the last line
- **sed -n ‘3,10p’ file :** it will print 3rd to 10th lines
- **sed ‘10d’ file :** print file contents except 10th line
- **sed ‘3,8d’ file :** print file contents except from 3rd to 8th lines
- **sed -i ‘10d’ line :** deletes tenth line from original file
- **sed -i.back ‘3,5d’ file :** keeps backup of file with .back extension before deleting the 3 to 5th lines

Find and replace with sed

- **sed ‘s/root/devops’ file :** substitute root with devops first word occurrence with new word
- **sed ‘s/root/devs/g’ file :** substitute root with devs globally in whole file
- **sed -i ‘s/root/devs/g’ file :** replaces root with devs globally in original file
- **sed -i.back ‘s/root/devs/g’ file :** keeps backup of the file before globally replacing the words in the file
- **sed ‘/search/s/old/new/g’ file :** replaces the old word with new word if that new line consists of “search” word

Insertion and deletion with sed command

- **sed -i ‘line_numberi (content to be inserted)’ file_name**
  - It inserts the content mentioned in the file before the specified line number
  - Here i beside line_number means insert
  - Example : sed -i ‘1i —----------’ test.txt
- **sed -i ‘line_numbera (content to be added)’ file_name**
  - It inserts the content mentioned in the file after the specified line number
  - Here a beside the line number means after
  - Example : sed -i ‘10a —----------’ test.txt

Curl request

- Curl makes a get request to the target URL and check if whether the server is able access the URL following HTTP request
- **Curl flags:**
  - **-s :** silent request, it will show progress bar
  - **-w :** it will display the information on terminal
  - **“%{http_code}” :** to get the http code and for success its 200

Checking status of the URL using curl

```
#!/bin/bash

URL="https://github.com/srijyothsna18/test
ie_champs"
response=$(curl -sw "%{http_code}" $URL)

content=$(echo "$response" | sed -n '$p')
echo "$content"

if [ "$content" = 200 ]; then
     echo "request is working fine"
else
     echo "request access is denied"
fi
```

vlab@HlYVLAB8:~/lochu/bash_script$ sh curl.sh

200

request is working fine

Top command

The top command provides a dynamic, real-time view of a system's processes, CPU usage, memory consumption, and more

Its Header Section consists of system overview like:

1. **Current time and uptime:** Shows the current time and how long the system has been running.
2. **Number of users:** Displays how many users are logged in.
3. **Load average:** Shows the system load average over the last 1, 5, and 15 minutes. The load average represents the average number of processes waiting to be run (higher numbers indicate higher load).
4. **Tasks:** Displays the total number of processes and their statuses (running, sleeping, stopped, and zombie).
5. **CPU usage: S**hows the percentage of CPU being used by user processes (us), system processes (sy), idle time (id), and more.
6. **Memory usage:** Displays total memory, free memory, used memory, and the amount of memory used for buffers and cache.
7. **Swap usage:** Shows total swap memory, used swap, free swap, and available memory.

Contd…

The lower section is a list of processes running on the system, with details about each process:

1. **PID:** Process ID.
2. **USER:** The user who owns the process.
3. **PR:** Process priority.
4. **NI:** Nice value (priority adjustment).
5. **VIRT:** Virtual memory used by the process.
6. **RES:** Resident memory (physical memory in use).
7. **SHR:** Shared memory used by the process.
8. **S:** Process state (e.g., R for running, S for sleeping).
9. **%CPU:** Percentage of CPU usage.
10. **%MEM:** Percentage of memory usage.
11. **TIME+:** Total CPU time the process has used.
12. **COMMAND:** The name of the command or process.

CPU load alert

```
#!/bin/bash

load=`top -bn1|grep load|awk '{printf
"%.2f\n", $(NF-2)}'`

echo $load

if [ "$(echo "$load > 4" | bc -l)" -eq 1
]; then

     echo "cpu load is very high :
$load"

else

     echo "cpu running fine"

fi
```

vlab@HYVLAB8:~/lochu/bash_script$ sh cpuload.sh

1.11

cpu running fine

Take backup of directories in system and transfer it to a specified location

```
#!/bin/bash

backup_dir=("$HOME/lochu/bash_script"
"$HOME/lochu/tmt")
dest_dir="$HOME/lochu/backup"
mkdir -p $dest_dir

backup_date=$(date +%b-%d-%y)

echo "starting backup of : ${backup_dir[@]}"

for dir in "${backup_dir[@]}"; do

      dir_name=$(basename "$dir")
      echo "$dir_name"

      if [ ! -d "$dir" ]; then
         echo "Directory $dir does not exist,
skipping..."
         continue
      fi
```

```
sudo tar -Pczf
"/tmp/${dir_name}-${backup_date}.tar.gz" "$dir"
      if [ $? -eq 0 ]; then
            echo "$i backup succeeded"
      else
            echo "$i backup failed"
      fi


      cp /tmp/${dir_name}-${backup_date}.tar.gz
$dest_dir
      if [ $? -eq 0 ];then
            echo "$i transferred succeeded"
      else
            echo "$i transferred failed"
      fi
done

sudo rm /tmp/*.gz
echo "backup is done"
```

contd…

vlab@HYVLAB8:~/lochu/bash_script$ ./backup_dir.sh
starting backup of : /home/vlab/lochu/bash_script /home/vlab/lochu/tmt
bash_script
 backup succeeded
 transferred succeeded
tmt
 backup succeeded
 transferred succeeded
backup is done
vlab@HYVLAB8:~/lochu/bash_script$ cd ..
vlab@HYVLAB8:~/lochu$ cd backup/
vlab@HYVLAB8:~/lochu/backup$ ls
bash_script-Oct-17-24.tar.gz tmt-Oct-17-24.tar.gz

systemd

- systemd is a system and service manager for Linux operating systems, designed to provide a fast and efficient way to manage services, processes, and the system's overall state.
- **Basic Components of systemd:**
    - **Units:** The basic objects in systemd, ach unit represents a resource that systemd manages, and they can be categorized into various types, each with specific purposes like service units (.service), socket units (.socket), mount units (.mount), and more.
    - **Targets:** Special types of units that group other units together, similar to runlevels in SysV init.
    - **Journal:** The logging system for systemd, where logs can be queried using journalctl.

systemctl command

- The systemctl command is a powerful tool in Linux for managing systemd services, sockets, devices, and other system resources. It is commonly used to control the state of the system and services on systems that use systemd as the init system
- **Basic Syntax**
  - **systemctl [OPTIONS] COMMAND [NAME]**

Common commands of systemctl

command	usage
sudo systemctl start service_name	Start a service
sudo systemctl stop service_name	Stop a service
sudo systemctl restart service_name	Restart a service
sudo systemctl reload service_name	Reload a service’s configuration without restarting
sudo systemctl enable service_name	Enable a service to start a boot
sudo systemctl disable service_name	Disable a service from staring on boot
systemctl list-units --type=service	List all loaded services
systemctl status service_name	Check the status of a service

journalctl command

- The Journal is the logging system used by systemd to collect and manage logs from various services, including system messages and application logs.
- **Common journalctl commands:**
  - **journalctl :** to view logs
  - **journalctl -u my_service.service :** view logs for a specific service
  - **journalctl -u my_service.service :** view logs since the last boot
  - **journalctl -f :** follows logs in real time

What is service in linux ?

- In easy terms , a service is a program or application in linux that runs or expects to run in the background. That is , it is running without the need for the user to be aware of it all the time
- Generally, a linux service has the following characteristics :
  - no graphical interface
  - **UNIT :** name of the service
  - **LOAD :** to know if it is loaded in the memory
  - **ACTIVE :** state in which it is (high level) can be active , reloading , inactive , failed , activating , deactivating
  - **SUB :** state of activation (low level) can be in one of the following
    - States : dead , closed , failed , inactive or running
  - **Description :** brief description of the service

Contd…

- The **[Unit]** section consists of description , documentation details etc
- **[Service]** section defines the service type, username , group , what to do in failure , restart timeout. The main is **'ExecStart'** which says to start our script file.
- You can also define **'ExecStartPre'** to define anything before the actual script file
- **'SyslogIdentifier'** is the keyword to identify our service in syslog
- Similarly , **'ExecStop'** is the instruction to day what to do to stop the service
- **[Install]** section is used to define different levels of target in the system

Creating a service in linux to monitor disk

- Lets create a service for continuously monitoring the disk
- Create a “.service” file in the directory “/etc/systemd/system” where all the service files are present
- Define all the [Unit],[Service] and [Install] sections
- Create a .sh file for printing “date” and disk usage “df -h”
- Start the service with the command “systemctl start “service_filename”
- Verify the output in the log file you have given in the shell script to print th date and disk usage

fs_monitor.service and disk_usage.sh

```
[Unit]
Description=FS monitoring service
Documentation=https://devopstechstack.com

[Service]
Type=simple
User=root
Group=root
TimeoutStartSec=0
Restart=on-failure
RestartSec=30s
#ExecStartPre=
ExecStart=/home/vlab/lochu/bash_scripts/disk_monitor.sh
SyslogIdentifier=Diskutilization
#ExecStop=

[Install]
WantedBy=multi-user.target
```

```
#!/bin/bash


#script check file system utilization every
30s store in a file

while true

do

      date >> /home/vlab/lochu/fs-monitor.txt

      sudo df -h >>
/home/vlab/lochu/fs-monitor.txt

      sleep 30

done
```

Output of fs_monitor.service

vlab@HYVLAB8:/etc/systemd/system$ systemctl status fs_monitor.service
● fs_monitor.service - FS monitoring service
Loaded: loaded (/etc/systemd/system/fs_monitor.service; disabled; ve>
Active: **inactive (dead)**
Docs: https://devopstechstack.com
vlab@HYVLAB8:/etc/systemd/system$ systemctl start fs_monitor.service
vlab@HYVLAB8:/etc/systemd/system$ systemctl status fs_monitor.service
● fs_monitor.service - FS monitoring service
Loaded: loaded (/etc/systemd/system/fs_monitor.service; disabled; vendor preset: enabled)
Active: **active (running)** since Fri 2024-10-18 16:03:35 IST; 24min ago
Docs: https://devopstechstack.com
Main PID: 247291 (disk_monitor.sh)
Tasks: 2 (limit: 9340)
Memory: 776.0K
CGroup: /system.slice/fs_monitor.service
├─247291 /bin/bash /home/vlab/lochu/bash_script/disk_monitor.sh
└─248011 sleep 30

vlab@HYVLAB8:~/lochu$ cat fs-monitor.txt
Friday 18 October 2024 04:03:35 PM IST

Filesystem	Size	Used	Avail	Use%	Mounted on
udev	3.9G	0	3.9G	0%	/dev
tmpfs	787M	1.9M	785M	1%	/run
/dev/sda1	46G	29G	15G	68%	/
tmpfs	3.9G	292M	3.6G	8%	/dev/shm
tmpfs	5.0M	4.0K	5.0M	1%	/run/lock
tmpfs	3.9G	0	3.9G	0%	/sys/fs/cgroup
/dev/loop0	128K	128K	0	100%	/snap/bare/5
/dev/loop2	64M	64M	0	100%	/snap/core20/2379
/dev/loop3	64M	64M	0	100%	/snap/core20/2318
/dev/loop4	347M	347M	0	100%	/snap/gnome-3-38-2004/119
/dev/loop8	75M	75M	0	100%	/snap/core22/1621
/dev/loop10	13M	13M	0	100%	/snap/snap-store/1113
/dev/loop11	92M	92M	0	100%	/snap/gtk-common-themes/1535
/dev/loop9	506M	506M	0	100%	/snap/gnome-42-2204/176
/dev/loop13	50M	50M	0	100%	/snap/snapd/18357
/dev/loop12	13M	13M	0	100%	/snap/snap-store/1216
/dev/loop14	39M	39M	0	100%	/snap/snapd/21759
/dev/loop1	75M	75M	0	100%	/snap/core22/1612
/dev/loop6	272M	272M	0	100%	/snap/firefox/5014
/dev/sda7	297G	130G	153G	46%	/home
tmpfs	787M	84K	787M	1%	/run/user/1000
/dev/loop15	272M	272M	0	100%	/snap/firefox/5091

Case statement

- The case command in Bash is a conditional statement that allows you to execute different blocks of code based on the value of a variable or expression. It's often used as an alternative to multiple if statements, especially when you have several possible values to check.
- Here’s the basic syntax of a case statement:

```
case expression in
    pattern1)
        # commands for pattern1
        ;;
    pattern2)
        # commands for pattern2
        ;;
    *)
        # commands if no patterns match
        ;;
esac
```

xargs command

- The xargs command in Unix is a utility that takes input from standard input (stdin) and converts it into arguments to a command.
- It is commonly used to build and execute command lines from standard input, allowing you to work with lists of data more effectively.
- **Basic syntax:**
  - **xargs [OPTION]... [COMMAND [ARGUMENTS]...]**
- **Usage:**
  - **xargs rm < files.txt**
  - This command will read each line of files.txt and pass it as an argument to rm

ps command

- The ps command in Unix and Linux is used to display information about the currently running processes. The -u option is used to show processes for a specific user.
- **Syntax:**
  - **ps -u lochu**
  - This command will display a list of all processes owned by the user lochu, including details like the process ID (PID), terminal (TTY), time, and command being executed.

chown command

- The chown command in Linux is used to change the ownership of files and directories.
- **Basic syntax:**
  - **chown [OPTIONS] NEW_OWNER:NEW_GROUP FILE**
  - **NEW_OWNER:** The username or user ID of the new owner.
  - **NEW_GROUP (optional):** The group name or group ID of the new group.
  - **FILE:** The file or directory for which you want to change the ownership.
- **Example:**
  - sudo chown john:staff example_dir

Script to automate account selection operation

- Below is the link to the code :
  - https://drive.google.com/file/d/1wp4nqeMnDo0QWalBqAzAxNFLTdSsrqiR/view?usp=sharing
- To add a user account in bash use the command
  - sudo useradd user_name
- To delete the user account use the command
  - sudo userdel user_name

Output if the user does not exist

vlab@HYVLAB8:~/lochu/bash_script$ ./delete_account.sh
step #1 - determine user account name to delete

1
please enter the user name of the user
account you wish to delete from system: lochu
1
is lochu the user account
you wish to delete from the system?[y/n]y

account , lochu , not found
 leaving script...

Output if we do not respond back to questions within time (60s)

vlab@HYVLAB8:~/lochu/bash_script$ ./delete_account.sh
step #1 - determine user account name to delete

1
please enter the user name of the user
account you wish to delete from system: lochu
1
is lochu the user account
you wish to delete from the system?[y/n]y

i found this record;
lochu:x:1001:1001::/home/lochu:/bin/sh

1
is this the correct user account? [y/n]y

step #2 - find process on system belonging to user account
there are no processes for this account currently running
step #3 : find files on system belonging to user account
creating a report of all files owned by lochu
it is recommended that you backup/archieve these file
and then do one of two things:
1)delete the files
2)change the files ownership to a current user account

please wait . this may take a while
report is complete
name of the report : lochu_Files_241021.rpt
location of report : /home/vlab/lochu/bash_script

step #4 : remove the account
1
do you wish to remove lochu's account from system? [y/n]2
please answer the questions

do you wish to remove lochu's account from system? [y/n]3
one last try.. please answer the question

do you wish to remove lochu's account from system? [y/n]4
since you refuse to answer the question
exiting program

Output to delete existing user by killing it’s running processes

vlab@HYVLAB8:~/lochu/bash_script$ ./delete_account.sh
step #1 - determine user account name to delete
1
please enter the user name of the user
account you wish to delete from system: lochu
1
is lochu the user account
you wish to delete from the system?[y/n]y

i found this record;
lochu:x:1001:1001::/home/lochu:/bin/sh
1
is this the correct user account? [y/n]y

step #2 - find process on system belonging to user account

there are no processes for this account currently running

step #3 : find files on system belonging to user account
creating a report of all files owned by lochu

it is recommended that you backup/archieve these file
and then do one of two things:
  1)delete the files
  2)change the files ownership to a current user account

please wait . this may take a while
[sudo] password for vlab:
report is complete
name of the report : lochu_Files_241021.rpt
location of report : /home/vlab/lochu/bash_script

step #4 : remove the account
1
do you wish to remove lochu's account from system? [y/n]y
user account lochu has been removed

THANK YOU