

DEVELOPMENT AUTOMATION LAB

SUBMITTED BY

Kshitiz Saini

R171218058

500067035

CSE-DevOps B2

SUBMITTED TO

Mr. PREMKUMAR CHITHALURU

Assistant Professor (SS)

Department of Systematics



UNIVERSITY WITH A PURPOSE

School of Computer Science

UNIVERSITY OF PETROLEUM AND ENERGY STUDIES

Bidholi Campus, Energy Acres

Dehradun – 248007

S. No.	Experiment	Page No.	Signature
BASICS			
1)	Linux Basics	1 - 2	
2)	Shebang, Setting the permissions of Shell	3	
CASE STUDY MODULE 2			
1)	Archiving Logs	5-6	
2)	Auto discard old archives	7-8	
3)	MYSQL backup	9-10	
4)	Email Web server summary	11-12	
5)	Ensure the web server is running	13-14	
6)	User command validation	15	
7)	Disk Usage Alarm	16-17	
8)	Sending file to trash	18	
9)	Restoring file from trash	19-20	
10)	Logging delete action	21-22	
11)	File formatter	23	
12)	Encrypting and Decrypting files	24-25	
13)	Decrypting files	26-27	
14)	System Information	28-29	
15)	Bulk file downloader	30-31	
16)	Install LAMP Stack	32-33	
17)	Get NIC's IP	34	
SHELL PROGRAMS			
1)	Convert Hexadecimal to Binary	35-36	
2)	Convert Octal to Binary	37-38	
3)	Checking for Palindrome	39	
4)	Checking for Perfect number	40	
5)	Checking for Armstrong number	41	
6)	Printing DO using *	42	
7)	Printing DA using +	43	
8)	Convert Hexadecimal to Octal	44	
9)	Convert Octal to Hexadecimal	45	
10)	Convert Decimal to Hexadecimal	46	
MAKE AND MAKEFILE			
1)	Basic Setup	47-48	
2)	Building binary from Source Code	49	
3)	Transforming makefile to adhere standard rules	50-51	

4)	Archiving logs with make	52-53	
5)	Conditionals in make	54-55	

Linux

Linux is a free open-source operating system based on Unix. Linus Torvalds originally created Linux with the assistance of developers from around the world.

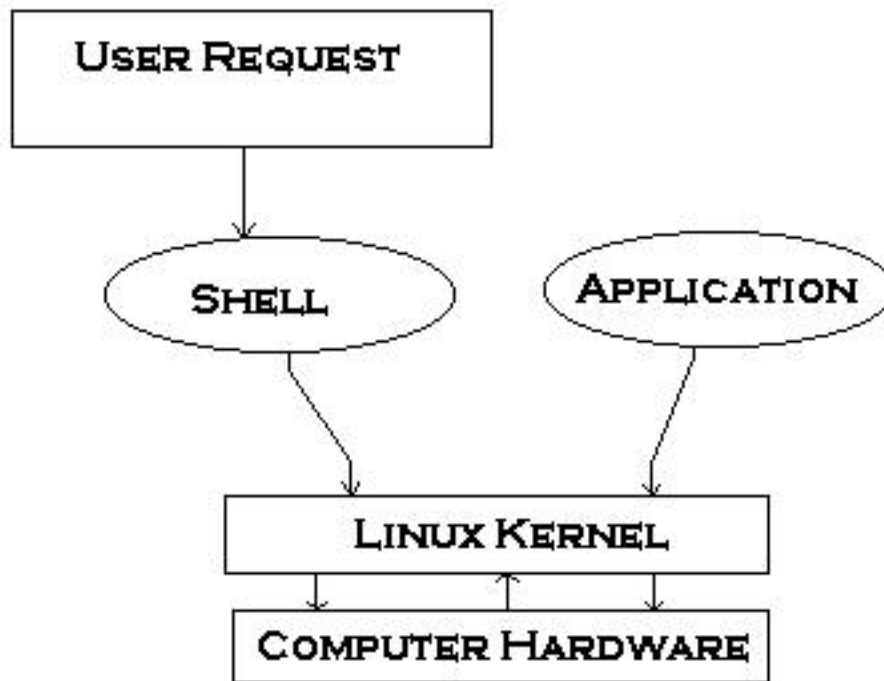
Key features of Linux are:

- Free
- Unix Like
- Open Source
- Network operating system

Linux is a kernel. A kernel provides access to the computer hardware and control access to resources such as:

- Files and data.
- Running programs.
- Loading programs into memory.
- Networks.
- Security and firewall.
- Other resources etc.

Kernel: It is the core of computer's program system. It has complete control over everything in the system.



OPERATING SYSTEM = KERNEL + UTILITIES + LIBRARY + MANAGEMENT SCRIPTS
+ INSTALLATION SCRIPTS

Shell: Shell is a special user program which provide interface to user to use Operating System services in a Command Line Interface.

How do I find out my current shell name?

To find shells available on our device we can use the command :

`cat /etc/shells`

```
kshitz@kshitz-zbook14:~$ cat /etc/shells
# /etc/shells: valid login shells
/bin/sh
/bin/bash
/bin/rbash
/bin/dash
```

Basic Command Line Editing

We can use the following key combinations to edit and recall commands:

- CTRL + L : Clear the screen.
- CTRL + W : Delete the word starting at cursor.
- CTRL + U : Clear the line i.e. Delete all words from command line.
- Up and Down arrow keys : Recall commands.
- Tab : Auto-complete files, directory, command names and much more.
- CTRL + R : Search through previously used commands.
- CTRL + C : Cancel currently running commands.
- CTRL + T : Swap the last two characters before the cursor.
- ESC + T : Swap the last two words before the cursor.
- CTRL + H : Delete the letter starting at cursor.

Executing A Command

Type your command, and press enter key. Try this the date command which will display current date and time:

`date`

```
kshitz@kshitz-zbook14:~$ date
Wed Oct  2 12:46:02 IST 2019
```

Getting Help In Linux

Most commands under Linux will come with documentation.

We can view documentation with the man command or info command. Many commands also accepts --help or -h command line option.

```
kshitz@kshitz-zbook14:~$ man date
kshitz@kshitz-zbook14:~$ info date
kshitz@kshitz-zbook14:~$ date --h
```

SHEBANG

The `#!` syntax used in scripts to indicate an interpreter for execution under UNIX / Linux operating systems.

Starting a Script with `#!`

1. It is called a shebang or a "bang" line.
2. It is nothing but the absolute path to the [Bash interpreter](#).
3. It consists of a number sign and an exclamation point character (`#!`), followed by the full path to the interpreter such as `/bin/bash`.
4. All scripts under Linux execute using the interpreter specified on a first line.
5. Almost all bash scripts often begin with `#!/bin/bash` (assuming that Bash has been installed in `/bin`)
6. This ensures that Bash will be used to interpret the script, even if it is executed under another shell.
7. The shebang was introduced by Dennis Ritchie between Version 7 Unix and 8 at Bell Laboratories. It was then also added to the BSD line at Berkeley.

Setting up permissions on a script

The `chmod command` (change mode) is a shell command in Linux. It can change file system modes of files and directories. The modes include permissions and special modes. Each shell script must have the execute permission. Mode can be either a symbolic representation of changes to make, or an octal number representing the bit pattern for the new mode bits.

Allowing everyone to execute the script, enter:

```
chmod +x script.sh
```

OR

```
chmod 0755 script.sh
```

Only allow owner to execute the script, enter:

```
chmod 0700 script.sh
```

To view the permissions, use:

```
ls -l script.sh
```

Grant all permissions to all users:

```
chmod 777 script.sh
```

Module 2

Automation Scenarios

Advantage of Automation

- Higher production rates.
- Better product quality.
- Improved safety.
- Greater output and increased productivity.
- Enhanced quality of work.

The Automation Scenarios are classified into two broad categories:

Scenarios that save time and effort:

In some situations, the same set of work should be done on multiple servers or multiple times on the same server. This category will walk through scenarios of this kind.

Scenarios that prevent errors:

Think of searching for a list of words in hundreds of files having thousands of lines. One cannot completely rely on a human to do this without any errors whereas a program is absolutely capable of doing this, within seconds with hundred percent reliability.

Automation Scenarios: Scenario 1 – Archiving Logs

How do I archive all the matching logs and move the archived files to the backup directory?

Algorithm :

Read Input: Directory under which all log files are located. Example: /var/log

Read Input: Extension of log files. Example: log

Read Input: Location of the backup directory

Run: Use archiving executables such as 7z or tar to create archives of the logs matching the given input.

Move the created archive files to the backup directory.

Delete the old log files.

Code :

```
#bin/bash
read -e -p "Enter name of process whose logs to fetch : " log_application
read -e -p "Enter location to save log archive : " archive_location
read -e -p "Enter name of archive : " archive_name
more /var/log/syslog | grep $log_application > $archive_location/$archive_name.txt
cd $archive_location
tar -cf $archive_name.tar $archive_name.txt
rm $archive_name.txt
echo ""
echo ""
echo "File archived is : $archive_name and is located at $archive_location"
```

Description :

*) “tar” or tape archive is a tool that is used for compression.

Tar flags:

C: create archive

V: verbose: Tell summary

F: File status

Du: Tells the disk usage

S: Summary

H: human readable form

Output :

```
kshitiz@ubuntu:~/MODULE 2$ ./lab1.sh
Enter name of process whose logs to fetch : date
Enter location to save log archive : /home/kshitiz
Enter name of archive : scenario1

File archived is : scenario1 and is located at /home/kshitiz
```

Automation Scenarios: Scenario 2 – Auto-Discard Old Archives

How to Delete archive files that are older than two days automatically?

Algorithm :

Read Argument: Location of the archived files.

Get the timestamp of all the archived files in the given location.

Check: The timestamp of the file is older than two days

True: Delete the file

Check: The script is scheduled in the cron

False: Add the script to cron such that it gets executed everyday at 12 AM.

Code :

```
#bin/bash
```

```
read -e -p "Enter location of archived file : " location
```

```
cd $location
```

```
rm $(find -mtime +2 -name "*.tar")
```

```
rm $(find -mtime +2 -name "*.tar.gz")
```

```
rm $(find -mtime +2 -name "*.tar.bz2")
```

Description :

*) find command is used to check for patterns and work on them simultaneously.

*) -type f is used to check whether locked file system is a file or not.

*) -mtime is used to check the timestamp in date order.

*) -exec: is used to execute a command after matched file is found.

Output :

```
kshitiz@ubuntu:~/MODULE 2$ ./2.sh
Enter location of archived file : /home/kshitiz/test1

Archives removed.
```

Automation Scenarios: Scenario 3 – MySQL (RDBMS) Backups

How do I take MySQL Backups every 12 hours and keep it safely on the backup directory?

Algorithm :

Read Arguments: Location of the backup directory and the MySQL Credentials file (Credentials file should have a hostname, port, username and password of the database server).

Check: The Location of the MySQL Credentials file is valid. If true, read the credentials as local variables, Else throw an error saying credentials not found and write to the log.

Run: MySQL Dump command to retrieve all the data from MySQL. If it returns a non-zero exit code, then exit the program and write the error to the log.

Redirect the stdout returned above to a new file in the backup directory. This file can be used to restore the database.

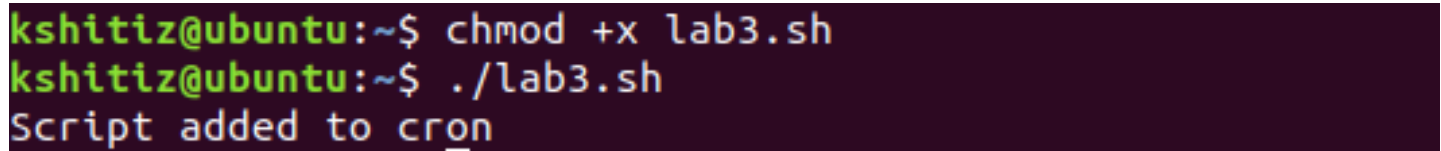
Check: The script is scheduled in the cron. If not, Add the script to cron such that it gets executed every 12 hours.

Code :

```
#!/bin/bash
if [ -z "$1" ]; then
    echo "ERROR: Credentials file not specified" >&2; exit 1;
elif [ -z "$2" ]; then
    echo "ERROR: Backup directory not specified" >&2; exit 1;
fi
credentials_file=$(realpath $1)
backup_directory=$(realpath $2)
if [ ! -f "$credentials_file" ]; then
    echo "ERROR: Credentials file does not exist" >&2; exit 1;
elif [ ! -d "$backup_directory" ]; then
    echo "ERROR: Backup directory does not exist" >&2; exit 1;
fi
source $credentials_file
if [ -z "${hostname:+word}" ]; then
    echo "ERROR: hostname is not set" >&2; exit 1;
elif [ -z "${username:+word}" ]; then
    echo "ERROR: username is not set" >&2; exit 1;
elif [ -z "${password:+word}" ]; then
    echo "ERROR: password is not set" >&2; exit 1;
fi
```

```
mysqldump -h$hostname -u$username -p$password --all-databases > backup.sql
if [[ $? != 0 ]]; then
    echo "ERROR: Error in taking mysql backup" >&2; exit 1;
fi
mv backup.sql $backup_directory/$(date +%F_%R).sql
path_to_script=$(realpath "$0")
if ! (crontab -l | grep -Fxq "0 */12 * * * $path_to_script $credentials_file $backup_directory"); then
    crontab -l | { cat; echo "0 */12 * * * $path_to_script $credentials_file $backup_directory"; } | crontab -
    echo "Script added to Cron"
fi
exit 0
```

Output :



```
kshitz@ubuntu:~$ chmod +x lab3.sh
kshitz@ubuntu:~$ ./lab3.sh
Script added to cron
```

Automation Scenarios: Scenario 4 – Email Web Server Summary

How do I email the summary of the web server requests every day?

Algorithm :

Read Argument: Location of Web Server's log files.

Filter the logs that are generated within the last 24 hours.

Iterate through the lines in the log and have a counter to track the count of each response code.

Finally, print the above summary and pipe it to the mail function so that summary can be emailed. If mail function returns a non-zero exit code, then exit the script and write the error message to the logs.

Check: The script is scheduled in the cron. If not, Add the script to cron such that it gets executed every 24 hours.

Code :

```
#!/bin/bash
if [ -f "$access_log.txt" ]; then
    rm $access_log.txt
fi
if [ -f "$error_log.txt" ]; then
    rm $error_log.txt
fi
cat /var/log/apache2/access_log > /Users/kshitiz/Desktop/case_study4/access_log.txt
cat /var/log/apache2/error_log > /Users/kshitiz/Desktop/case_study4/error_log.txt
read -e -p "Enter the name of the Access logs of HTTP web server: " access_logfile
read -e -p "Enter the name of the Error logs of the HTTP web server: " error_logfile
read -e -p "Enter the Email ID: " Email
access_log1=$(realpath $access_logfile)
error_log1=$(realpath $error_logfile)
if [ ! -f "$access_log1" ]; then
    echo "ERROR: The access log file does not exist" >&2; exit 1;
fi
if [ ! -f "$error_log1" ]; then
    echo "ERROR: The error log file does not exist" >&2; exit 1;
fi
cat $access_log1 | mail -s "Apache web server access logs" $Email
if [[ $? != 0 ]]; then
    echo "ERROR: Error in sending access log to email to $Email" >&2; exit 1;
fi
cat $error_log1 | mail -s "Apache web server error logs" $Email
```

```
if [[ $? != 0 ]]; then
    echo "ERROR: Error in sending error log to mail to $Email" >&2; exit 1;
fi
echo "The access and error logs both are successfully sent to $Email"
exit 0
```

Output :

```
kshitiz@ubuntu:~$ ./lab4.sh
The access and error logs both are successfully sent to 500067035@stu.upes.ac.in
kshitiz@ubuntu:~$
```

Automation Scenarios: Scenario 5 – Ensure Web Server is Running

Should I continuously monitor and Restart the web server if it is not running?

Algorithm :

Read Argument: Web server's listening port.

Run: 'netstat' command to find all the listening ports of the server.

Filter the above list with the web server's list port.

If the filtered list is empty, then write the timestamp to log and restart the web server.

Check: The script is scheduled in the cron.

False: Add the script to cron such that it gets executed every minute.

Code :

```
# bin/bash
read -p "Enter listening port : " listening_port
netstat -au | grep -q ":listening_port"
if [ [ $? != 0 ] ];
then
echo "ERROR, Web server is not running" > &2
/etc/init.d/apache2 restart
fi
backupfolder="/home/kshitiz/backups"
cp -r /var/log/apache2/ $backupfolder
netstat -au
netstat -lu
```

Description :

*) netstat command is used to find the net (web server status)

*) In computing, netstat (network statistics) is a command-line network utility tool that displays network connections for the Transmission Control Protocol (both incoming and outgoing), routing tables, and a number of network interface (network interface controller or software-defined network interface) and network

*) -au flag is used to check only for internal server

*) -lu is used to check for servers and public connectivity as well.

Output :

```
kshitiz@ubuntu:~/MODULE 2$ ./lab5.sh
Enter listening port : 12401
./lab5.sh: line 6: [: too many arguments
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
udp        0      0 0.0.0.0:ipp             0.0.0.0:*
udp        0      0 0.0.0.0:39595           0.0.0.0:*
udp        0      0 0.0.0.0:mdns            0.0.0.0:*
udp        0      0 localhost:domain        0.0.0.0:*
udp        0      0 0.0.0.0:bootpc          0.0.0.0:*
udp6       0      0 [::]:mdns               [::]:*
udp6       0      0 [::]:52892              [::]:*
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
udp        0      0 0.0.0.0:ipp             0.0.0.0:*
udp        0      0 0.0.0.0:39595           0.0.0.0:*
udp        0      0 0.0.0.0:mdns            0.0.0.0:*
udp        0      0 localhost:domain        0.0.0.0:*
udp        0      0 0.0.0.0:bootpc          0.0.0.0:*
udp6       0      0 [::]:mdns               [::]:*
udp6       0      0 [::]:52892              [::]:*
```

Automation Scenarios: Scenario 6 – User Command Validation

How should I allow the users to execute other commands normally whereas the forbidden commands should throw an error?

Algorithm :

Check: Script has sudo permissions so that it can read files belonging to other users.

False: Throw an error saying "Run as root or sudo user"

Check: The forbidden commands and their error messages exist in a file /root directory.

False: Show editor to create and edit the file.

Read the list of actual users from the /etc/passwd file.

Add a function to the '.bashrc' file of all users that checks whether the given argument matches any pattern in the forbidden commands file. If the pattern is matched, throw an error showing the provided error message.

Point this to the trap directive of the '.bashrc' file so that this gets called before the users' command executes.

Code :

```
#bin/bash
b=$?
read -e -p "Enter the command to check : " command
test=`$command`
a=$?
if [ $a -eq $b ];
then
echo "Command Found"
echo "Type of Command is $(type -p $command)"
else
echo "Command not Found"
fi
```

Output :

```
kshitiz@ubuntu:~/MODULE 2$ ./lab6.sh
Enter the command to check : tty
Command Found
Type of Command is /usr/bin/tty
kshitiz@ubuntu:~/MODULE 2$ ./lab6.sh
Enter the command to check : ppy
Command not Found
```

Automation Scenarios: Scenario 7 – Disk Usage Alarm

How to monitor the Disk usage and alert if it is beyond the given threshold?

Algorithm :

Read Argument: The device name for which the disk usage should be monitored.

Check: No device exists with the given device name.

True: Throw an error and write to the logs.

Check: Threshold limit is greater than the maximum size of the Disk.

True: Throw an error and write to the logs.

Run the disk utility command to get the usage of the disk.

Check: Disk usage is greater than the Threshold limit.

True: Write to the /etc/motd file so that it will be shown whenever a user logs in.

Check: The script is scheduled in the cron.

False: Add the script to cron such that it gets executed every minute.

Code :

```
#!/bin/bash
LIMIT='80'
dir_name='/var'
subject="Disk Usage of $dir_name in Ubuntu"
cd $dir_name
used=`df . | awk '{print $5}' | sed -ne 2p | cut -d"%" -f1`
if [ $used -gt $LIMIT ]
then
    echo "Greater than limit."
fi
```

Description :

*) Limit keyword is used to check that whether the usage is in that percentage or above in 100 % value.

*) Mailx tool is used to send the mails via web server.

*) awk options 'selection_criteria { action }' input-file > output-file

*) The cut command in UNIX is a command for cutting out the sections from each line of files and writing the result to standard output. It can be used to cut parts of a line by byte position, character and field. Basically the cut command slices a line and extracts the text. It is necessary to specify option with command otherwise it gives error. If more than one file name is provided then data from each file is not precedes by its file name

Output :

```
kshitiz@ubuntu:~/MODULE 2$ ./lab7.sh  
Used % is 29
```

Automation Scenarios: Scenario 8 – Sending Files to Trash

How to move the deleted files/folders to the recycle bin?

Algorithm :

Set the location of the recycle bin directory and the locations of the remove and copy tool.

If there are no arguments given, then throw an error asking to provide the file path as arguments.

Get all the arguments using the getopt function over the while loop.

Map the arguments with their respective actions using the switch case statement. If 'f' is passed in the argument, delete the files/folders permanently by direct remove command. All the other arguments can be passed on to the remove command.

Create a recycle bin directory if doesn't exist.

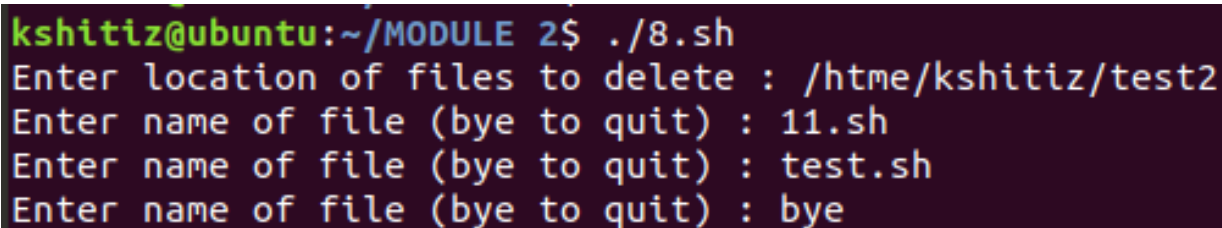
Copy the files/folders to the recycle bin and prefix the current date to those files.

Provide all other arguments and file paths to the remove utility.

Code :

```
#bin/bash
read -e -p "Enter location of files to delete : " LOCATION
INPUT_STRING=hello
while [ "$INPUT_STRING" != "bye" ]
do
read -e -p "Enter name of file (bye to quit) : " INPUT_STRING
mv $LOCATION/$INPUT_STRING .local/share/Trash/files
echo ""
echo ""
done
```

Output :

A terminal window with a dark background and light-colored text. The prompt is 'kshitiz@ubuntu:~/MODULE 2\$'. The user enters './8.sh'. The script prompts for the location of files to delete, and the user enters '/htme/kshitiz/test2'. The script then prompts for the name of the file to move, and the user enters '11.sh'. The script then prompts for the name of the file to move, and the user enters 'test.sh'. Finally, the script prompts for the name of the file to move, and the user enters 'bye'.

```
kshitiz@ubuntu:~/MODULE 2$ ./8.sh
Enter location of files to delete : /htme/kshitiz/test2
Enter name of file (bye to quit) : 11.sh
Enter name of file (bye to quit) : test.sh
Enter name of file (bye to quit) : bye
```

Automation Scenarios: Scenario 9 – Restoring Files from Recycle Bin

How to restore the deleted files/folders from the recycle bin?

Algorithm :

Set the location of the recycle bin directory, current directory and the path of the remove and copy tool.

Check whether the recycle bin directory exists in the current user's home directory. If not, throw an error.

If there are no arguments given, list the files in the recycle bin and exit.

If the pattern is provide in the arguments, search files matching that pattern and if nothing is found, throw an error say no match found.

If many matches are found, show the date, time, size and name of the file or folder and ask the user to choose a version to export.

If zero is entered, exit the application doing nothing. Else check if there is a file/folder with the same name in our current working directory and throw an error if true.

If everything above is fine, then move the chosen file from the recycle bin to the current directory and ask the user whether to delete the other versions of the file in the recycle bin. If yes, then delete other versions of the file. Else Retain them.

Code :

```
#bin/bash
```

```
read -e -p "Enter location where you want to put files : " LOCATION
```

```
INPUT_STRING=hello
```

```
while [ "$INPUT_STRING" != "bye" ]
```

```
do
```

```
read -e -p "Enter name of file (bye to quit) : " INPUT_STRING
```

```
mv .local/share/Trash/files/$INPUT_STRING $LOCATION
```

```
echo ""
```

```
echo ""
```

```
done
```

Output :

```
kshitiz@ubuntu:~/MODULE 2$ ./9.sh
Enter location where you want to put files : /htme/kshitiz/test1
Enter name of file (bye to quit) : 11.sh
Enter name of file (bye to quit) : test.sh
Enter name of file (bye to quit) : bye
```


Automation Scenarios: Scenario 10 – Logging Delete Actions

How to Log all the deleted actions?

Algorithm :

Set the path of the file to store the delete logs.

Get the list of files/directories to remove from the argument.

If no arguments are given, then show the usage instruction.

If -s is passed in the argument, do not log the delete action.

Just remove the -s argument and pass the remaining to the remove tool without logging.

If only the path of the files/directories is given in the arguments, add the command entered with arguments prefixed by the date, time and user name to the log file and then delete the file.

To replace the default remove tool, the path to the script can be kept as an alias in the .bashrc file of the users.

Code :

```
#!/bin/bash
if [ $# -eq 0 ]
then
echo "No inputs send via command line!! Exiting" >&2; exit 1;
fi
path=$1
path1="/home/kshitiz/MODULE2/"
if [ -d $path ]
then
rm -r $path
echo `date +%F` >> $path1/del.log
echo "$whoami" >> $path1/del.log
echo "$path" >> $path1/del.log
fi
if [ -f $path ]
then
rm $path
echo `date +%F` >> $path1/del.log
echo "$USERs" >> $path1/del.log
echo "$path" >> $path1/del.log
fi
```

Description :

*) “%F” flag along with date includes the full format of dates as yyyy-mm-dd format

*) “USER” command is the alternative to the “whoami” command.

*) `rm -r` does recursive deletion in the subfolders as well.

Output :

```
kshitiz@ubuntu:~/MODULE_2$ ./lab10.sh /home/kshitiz/test2
kshitiz@ubuntu:~/MODULE_2$ ./lab10.sh /home/kshitiz/test1/11.sh
kshitiz@ubuntu:~/MODULE_2$
```

delete.log created in the folder.

Automation Scenarios: Scenario 11 – File Formatter

How to Read and Display all the file inputs with line numbers?

Algorithm :

Read the list of files acquired through arguments in a for loop and iterate over each file.

In each iteration, Get the count of lines and the characters with the help of the default word count (wc) utility available in Linux.

List (ls) utility can be used to get the owner of the file.

Print the header bar that displays filename, its lines, its characters and the owner of the file.

Set a counter to indicate the line numbers of the file. Loop through each line of the file and prefix the line content with the respective line number for each iteration.

Do the same for all files and finally show the output through the less utility for better accessibility.

Code :

```
#!/bin/bash
FILES=/home/kshitiz/test2/file2.txt
for f in $FILES
do
echo "Processing $f"
echo -en '\n'
wc $f
echo -en '\n'
#ls -l $f
stat -c "%U" $($f)
cat -n $f
echo -en '\n'
echo -en '\n'
echo -en '\n'
done
```

Output :

```
kshitiz@ubuntu:~/MODULE_2$ ./lab11.sh
Processing /home/kshitiz/test2/file2.txt

0 0 0 /home/kshitiz/test2/file2.txt
```

Automation Scenarios: Scenario 12 – Encrypting Files

How to Encrypt the given files or folders recursively?

Algorithm :

If no arguments are given, throw an error and exit the program.

Read the password to use for encrypting the files.

Find the path of the files from the given argument. If a directory is given, get the files recursively under that directory. Throw an error and exit the program if the resolved file is not a suitable file type.

Sort and remove the duplicate entries in the files list.

Loop through each file and encrypt the file with the builtin 'gpg' utility with the given password.

If the 'gpg' utility exits with a non-zero exit code, throw the error message and end the program. Else delete the original file and show the encrypted file's name to indicate that it is successfully encrypted.

Finally, print a message to indicate that encryption is completed and exit the program.

Code :

```
#!/bin/bash
read -p "Enter the path of the folder to Encrypt: " f
gpg -c $f
echo "Folder encrypted!!"
read -p "Do you want to decrypt now? " res
case $res in
y|Y|yes|YES) gpg -d $f.gpg ;;
*) echo "Folder is still encrypted!!!" ;;
esac
```

Description:

*) -c command is used to create encryption with a symmetric password.

Output :

```
kshitiz@ubuntu:~/MODULE_2$ ./lab12_13.sh
Enter the path of the folder to Encrypt: /home/kshitiz/test1/file2.txt
Folder encrypted!!
Do you want to decrypt now? no
Folder is still encrypted!!!
kshitiz@ubuntu:~/MODULE_2$ ./lab12_13.sh
Enter the path of the folder to Encrypt: /home/kshitiz/file2.txt
Folder encrypted!!
Do you want to decrypt now? yes
gpg: AES256 encrypted data
gpg: encrypted with 1 passphrase
```

Automation Scenarios: Scenario 13 – Decrypting Files

How to Decrypt the given files or folders recursively?

Algorithm :

If no arguments are given, throw an error and exit the program.

Read the password to use for decrypting the files.

Find the path of the files from the given argument. If a directory is given, get the files recursively under that directory. Throw an error and exit the program if the resolved file is not a suitable file type.

Sort and remove the duplicate entries in the files list. Filter only the files that end with '.gpg' extension.

Loop through each file and decrypt the file with the builtin 'gpg' utility with the given password. This utility prints the decrypted content to the stdout. Hence redirect the stdout to a file with the same name but without '.gpg' extension at the end.

If the 'gpg' utility exits with a non-zero exit code, throw the error message and end the program. Else delete the encrypted file and show the decrypted file's name to indicate that it is successfully decrypted.

Finally, print a message to indicate that decryption is completed and exit the program.

Code :

```
#!/bin/bash
read -p "Enter the path of the folder to Encrypt: " f
gpg -c $f
echo "Folder encrypted!!"
read -p "Do you want to decrypt now? " res
case $res in
y|Y|yes|YES) gpg -d $f.gpg ;;
*) echo "Folder is still encrypted!!!" ;;
esac
```

Description:

*) -c command is used to create encryption with a symmetric password.

Output :

```
kshitiz@ubuntu:~/MODULE_2$ ./lab12_13.sh
Enter the path of the folder to Encrypt: /home/kshitiz/test1/file2.txt
Folder encrypted!!
Do you want to decrypt now? no
Folder is still encrypted!!!
kshitiz@ubuntu:~/MODULE_2$ ./lab12_13.sh
Enter the path of the folder to Encrypt: /home/kshitiz/file2.txt
Folder encrypted!!
Do you want to decrypt now? yes
gpg: AES256 encrypted data
gpg: encrypted with 1 passphrase
```

Automation Scenarios: Scenario 14 – System Information

How to show the most often required status and information of the system?

Algorithm :

The most often checked status of a computer while troubleshooting is given below.

date - Prints the current date and time based on the local time zone of the system

w - A built-in utility to show currently logged in users, their session duration, their IP and their mode of login.

last - This utility shows the history of logins made. Since we don't need the entire history, only the last 5 logins can be shown.

df - It prints the list of available disks with their used space, free space, total space, no. of blocks and their mount points.

free - This tool provides the details about the installed Memory and the allocated Swap Memory.

top - This helps in displaying the running processes with their memory and CPU consumptions details.

netstat - This network utility can provide the information of established and listening connections from the current system.

ss - A very extensive tool to get the statistics of all the sockets in the system.

Code :

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

```
date
```

```
w
```

```
last
```

```
df
```

```
free
```

```
top
```

```
ss
```


Output :

```
kshitz@ubuntu:~/MODULE_2$ ./lab14.sh
Mon Nov 25 08:53:57 IST 2019
08:53:57 up 16 min, 1 user, load average: 0.55, 0.58, 0.66
USER      TTY      FROM            LOGIN@   IDLE   JCPU   PCPU WHAT
kshitz    :1        :1              08:38    ?xdm?  1:16   0.01s /usr/lib/gdm3/g
kshitz    :1        :1              Mon Nov 25 08:38    still logged in
reboot    system boot  5.0.0-35-generic Mon Nov 25 08:37    still running
kshitz    :1        :1              Thu Nov 21 13:12 - crash (3+19:25)
reboot    system boot  5.0.0-35-generic Thu Nov 21 13:12    still running
kshitz    :1        :1              Tue Nov 12 23:42 - crash (8+13:29)
reboot    system boot  5.0.0-32-generic Tue Nov 12 23:38    still running
kshitz    :0        :0              Mon Nov 11 08:22 - crash (1+15:16)
reboot    system boot  5.0.0-32-generic Mon Nov 11 08:21    still running

wtmp begins Mon Nov  4 16:35:06 2019
Filesystem            1K-blocks      Used Available Use% Mounted on
udev                   1498660         0   1498660   0% /dev
tmpfs                   304556         0    304556   1% /run
/dev/sda1              30830500 8967012 20274344 31% /
tmpfs                  1522776         0   1522776   0% /dev/shm
tmpfs                   5120           4     5116   1% /run/lock
tmpfs                  1522776         0   1522776   0% /sys/fs/cgroup
/dev/loop2             4352          4352         0 100% /snap/gnome-calculator/544
/dev/loop1            144128       144128         0 100% /snap/gnome-3-26-1604/97
/dev/loop3             3840          3840         0 100% /snap/gnome-system-monitor/111
/dev/loop0            91264         91264         0 100% /snap/core/8039
/dev/loop6            55936         55936         0 100% /snap/core18/1265
/dev/loop7            55808         55808         0 100% /snap/core18/1192
/dev/loop5             1024          1024         0 100% /snap/gnome-logs/81
/dev/loop8            159872       159872         0 100% /snap/gnome-3-28-1804/91
/dev/loop9            91264         91264         0 100% /snap/core/7917
/dev/loop11           144128       144128         0 100% /snap/gnome-3-26-1604/98
/dev/loop10           45312         45312         0 100% /snap/gtk-common-themes/1353
/dev/loop13           15104         15104         0 100% /snap/gnome-characters/359
/dev/loop15            3840          3840         0 100% /snap/gnome-system-monitor/107
/dev/loop16            1024          1024         0 100% /snap/gnome-logs/73
/dev/loop17            4352          4352         0 100% /snap/gnome-calculator/501
/dev/loop18           15104         15104         0 100% /snap/gnome-characters/367
/dev/loop19           43904         43904         0 100% /snap/gtk-common-themes/1313
tmpfs                  304552         16    304536   1% /run/user/121
tmpfs                  304552         52    304500   1% /run/user/1000
/dev/loop12           160512       160512         0 100% /snap/gnome-3-28-1804/110
total                  304556       154514    184760   shared buff/cache available
Mem:                  3045556       154514    184760   43684   1315656   1264792
```

Automation Scenarios: Scenario 15 – Bulk File Downloader

How to download all the files provided by the list of given URLs?

Algorithm :

Set the default directory to store downloaded files.

Read the path of the downloads list from the arguments. If it is not provided, throw an error and exit the program.

If a second argument is given, set the given argument as the download directory.

Check if the download directory exists. If not, create the directory.

wget - A built-in utility to download the files from a HTTP or HTTPS endpoint.

Loop through each lines of the download list file and use wget in each iteration to download the file by providing the URL and path to download as inputs.

Code :

```
#!/bin/bash
a=20
while [ $a -lt 40 ]
do
wget "https://nptel.ac.in/courses/111104075/pdf/Module1/Lecture$a-Module1-Anova-1.pdf"
a=$((a+1))
done
```

Description:

*) wget is the non-interactive network downloader which is used to download files from the server even when the user has not logged on to the system and it can work in background without hindering the current process.

Output :

[illegible]

Automation Scenarios: Scenario 16 – Install LAMP Stack

How to install Apache, MySQL, PHP on Linux machine (LAMP stack)?

Algorithm :

Update the Apt packages so that apt tool knows where to get the required softwares.

Pull the latest update and security patches available for your Linux distribution by the 'apt-get upgrade' command.

Running 'apt-get install Apache2' will install the latest stable version of apache web server (2.4.x).

Let us install PHP 7 as the programming language of our choice. Most popular open source software such as Wordpress (Blog), Drupal (CMS) and Magento (E-commerce framework) are written in PHP.

As the data backend, MySQL can be installed. mysql-server listens on port 3306 for incoming connections and mysql-client is capable of initiating a connection to the mysql-server.

To provide appropriate permissions for the executable code, we change the ownership of the /var/www directory so that apache server will have sufficient permissions to read and execute the code located under this directory. For modern web applications, we need the Apache's rewrite module to be turned on.

Finally restart the Apache web server for all the changes to take effect.

Code :

```
#!/bin/bash
echo -e "\n\nUpdating Apt Packages and upgrading latest patches\n"
sudo apt-get update -y && sudo apt-get upgrade -y
echo -e "\n\nInstalling Apache2 Web server\n"
sudo apt-get install apache2 apache2-doc apache2-mpm-prefork apache2-utils libexpat1 ssl-cert -y
echo -e "\n\nInstalling PHP & Requirements\n"
sudo apt-get install libapache2-mod-php7.0 php7.0 php7.0-common php7.0-curl php7.0-dev php7.0-gd php-pear php7.0-mcrypt php7.0-mysql -y
echo -e "\n\nInstalling MySQL\n"
sudo apt-get install mysql-server mysql-client -y
echo -e "\n\nPermissions for /var/www\n"
sudo chown -R www-data:www-data /var/www
echo -e "\n\nPermissions have been set\n"
echo -e "\n\nEnabling Modules\n"
sudo a2enmod rewrite
sudo phpenmod mcrypt
echo -e "\n\nRestarting Apache\n"
sudo service apache2 restart
echo -e "\n\nLAMP Installation Completed"
exit 0
```

Output :

```
Installing MySQL
Reading package lists... Done
Building dependency tree
Reading state information... Done
mysql-client is already the newest version (5.7.27-0ubuntu0.18.04.1).
mysql-server is already the newest version (5.7.27-0ubuntu0.18.04.1).
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.

Permissions for /var/www

Permissions have been set

Enabling Modules
Module rewrite already enabled
sudo: phpenmod: command not found

Restarting Apache

LAMP Installation Completed
```

Automation Scenarios: Scenario 17 – Get NIC's IPs

How to Find Public IP and private IP of the given network interface card?

Algorithm :

Check if NIC name is given in the argument. If not, throw an error and exit the program.

ip - A built-in tool to get the network related information and modify the network settings of the Linux OS.

If the given NIC name doesn't exist, throw an error saying that NIC doesn't exist and exit the program.

Using the ip tool, get the information of the given NIC and pipe the output to the 'sed' utility to fetch the Private IP address of the interface.

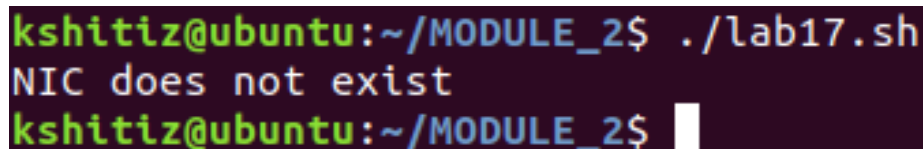
To find the Public IP of the interface, we need some server outside the LAN. Hence the public site <http://ifconfig.co> can be used to retrieve the Public IP of our system.

By doing a 'curl --interface eth0 http://ifconfig.co/ip', the Public IP of the eth0 interface will be returned in the stdout and can be used further.

Code :

```
#!/bin/bash
if[ $( ifconfig | grep -o $1 ) == $1 ]
then
ifconfig $1 | sed 3,8d | sed 1d
echo " Public IP: "
curl --interface $1 http://ifconfig.co
else
echo "NIC does not exist"
exit 125
fi
```

Output :



```
kshitiz@ubuntu:~/MODULE_2$ ./lab17.sh
NIC does not exist
kshitiz@ubuntu:~/MODULE_2$
```

Shell Scripts

1) Shell script to convert Hexadecimal to Binary

Algorithm :

To convert HexaDecimal number to Binary, the binary equivalent of each digit of the HexaDecimal number is evaluated and combined at the end to get the equivalent binary number.

Code :

```
#bin/bash
#Converting hexadecimal to binary
echo "Enter a Hexadecimal Number with spaces between digits : "
read -a string
for element in ${string[@]}
do
case $element in
0)
echo "$element -- 0000"
;;
1)
echo "$element -- 0001"
;;
2)
echo "$element -- 0010"
;;
3)
echo "$element -- 0011"
;;
4)
echo "$element -- 0100"
;;
5)
echo "$element -- 0101"
;;
6)
echo "$element -- 0110"
;;
7)
echo "$element -- 0111"
;;
```

```

8)
    echo "$element -- 1000"
    ;;
9)
    echo "$element -- 1001"
    ;;
a)
    echo "$element -- 1010"
    ;;
b)
    echo "$element -- 1011"
    ;;
c)
    echo "$element -- 1100"
    ;;
d)
    echo "$element -- 1101"
    ;;
e)
    echo "$element -- 1110"
    ;;
f)
    echo "$element -- 1111"
    ;;
*)
    echo "Invalid hexadecimal digit. -- $element">&2
esac
done

```

Output :

```

kshitiz@kshitiz-zbook14:~/repo/Development-Automation-Tasks/WHATSAPP_QUESTIONS$
./problem1.sh
Enter a Hexadecimal Number with spaces between digits :
1 2 4 0 1
1 -- 0001
2 -- 0010
4 -- 0100
0 -- 0000
1 -- 0001

```


Shell Scripts

2) Shell script to convert Octal to Binary

Algorithm :

To convert an Octal number to Binary, the binary equivalent of each digit of the octal number is evaluated and combined at the end to get the equivalent binary number.

Code :

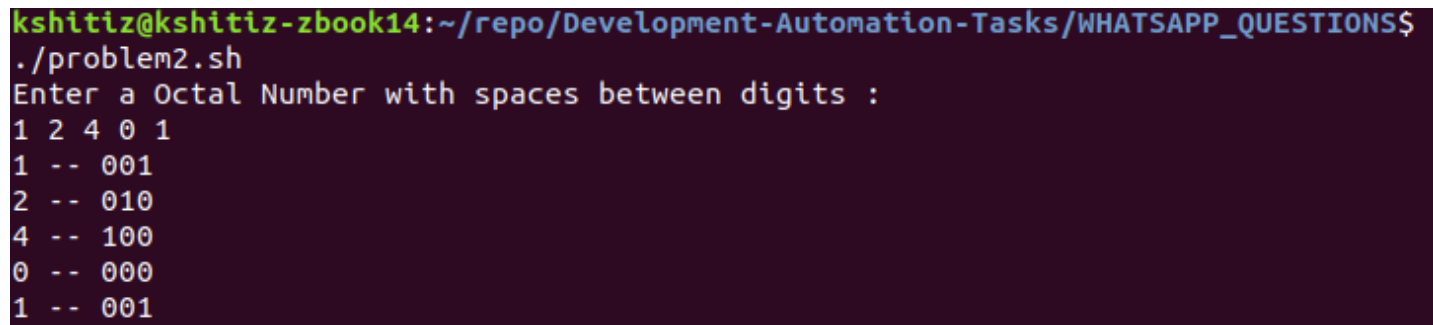
```
#bin/bash
#Converting octal to binary

echo "Enter a Octal Number with spaces between digits : "
read -a string

for element in ${string[@]}
do
case $element in
0)
echo "$element -- 000"
;;
1)
echo "$element -- 001"
;;
2)
echo "$element -- 010"
;;
3)
echo "$element -- 011"
;;
4)
echo "$element -- 100"
;;
5)
echo "$element -- 101"
;;
6)
echo "$element -- 110"
;;
7)
```

```
        echo "$element -- 111"
    ;;
*)
    echo "Invalid octal digit. -- $element">&2
esac
done
```

Output :



```
kshitiz@kshitiz-zbook14:~/repo/Development-Automation-Tasks/WHATSAPP_QUESTIONS$
./problem2.sh
Enter a Octal Number with spaces between digits :
1 2 4 0 1
1 -- 001
2 -- 010
4 -- 100
0 -- 000
1 -- 001
```

Shell Scripts

3) Shell script to check for Palindrome

Algorithm :

- Input the number.
- Find the reverse of the number.
- If the reverse of the number is equal to the number, then return true. Else, return false.

Code :

```
#bin/bash
#Checking for palindrome
read -p "Enter number : " n
number=$n
reverse=0
while [ $n -gt 0 ]
do
a=`expr $n % 10 `
n=`expr $n / 10 `
reverse=`expr $reverse \* 10 + $a`
done
echo "Reverse number is : $reverse"
if [ $number -eq $reverse ]
then
    echo "Number is palindrome"
else
    echo "Number is not palindrome"
fi
```

Output :

```
kshitiz@kshitiz-zbook14:~/repo/Development-Automation-Tasks/WHATSAPP_QUESTION$
./problem3.sh
Enter number : 12401
Reverse number is : 10421
Number is not palindrome
kshitiz@kshitiz-zbook14:~/repo/Development-Automation-Tasks/WHATSAPP_QUESTION$
./problem3.sh
Enter number : 121
Reverse number is : 121
Number is palindrome
```

Shell Scripts

4) Shell script to check for Perfect Number

Algorithm :

- Input the number.
- Find all divisors of the number except the number itself.
- If the sum of all divisors of the number is equal to the number, then return true. Else, return false.

Code :

```
#bin/bash
echo Enter a number
read no
i=1
ans=0
while [ $i -le 'expr $no / 2' ]
do
    if [ 'expr $no % $i' -eq 0 ]
    then
        ans='expr $ans + $i'
    fi
    i='expr $i + 1'
done
if [ $no -eq $ans ]
then
    echo $no is perfect
else
    echo $no is NOT perfect
fi
```

Output :

```
kshitiz@kshitiz-zbook14:~/repo/Development-Automation-Tasks/WHATSAPP_QUESTION$
./problem4.sh
Enter a number
12
./problem4.sh: line 5: [: expr $no / 2: integer expression expected
12 is NOT perfect
```

Shell Scripts

5) Shell script to check for Armstrong number

Algorithm :

- read number
- set sum=0 and duplicate=number
- reminder=number%10
- sum=sum+(reminder*reminder*reminder)
- number=number/10
- repeat steps 4 to 6 until number > 0
- if sum = duplicate
display number is armstrong
- else
display number is not armstrong
- stop

Code :

```
#bin/bash
#Check if a number is armstrong
read -p "Enter a number : " number
temp=$number
sum=0
digits=0
n=0
while [ $temp -gt 0 ]
do
digits=`expr $temp % 10`
n=`expr $digits \* $digits \* $digits`
sum=`expr $sum + $n`
temp=`expr $temp / 10`
done
if [ $sum -eq $number ]
then
echo "It is an Armstrong Number."
else
echo "It is not an Armstrong Number."
fi
```

Output :

```
kshitiz@kshitiz-zbook14:~/repo/Development-Automation-Tasks/WHATSAPP_QUESTION$
./problem5.sh
Enter a number : 153
It is an Armstrong Number.
```

Shell Scripts

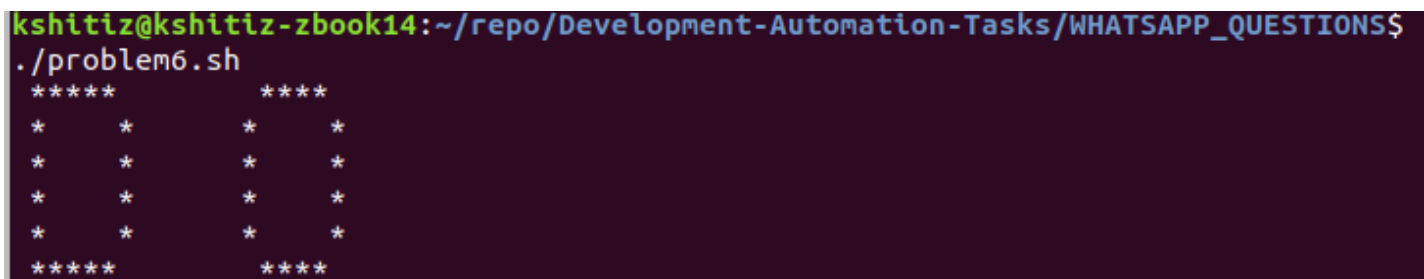
6) Shell script to print "DO" using stars (*)

Code :

```
#bin/bash
#Printng do with *
```

```
echo " *****"
echo " *   *   *   *"
echo " *   *   *   *"
echo " *   *   *   *"
echo " *   *   *   *"
echo " *****"
```

Output :



```
kshitz@kshitz-zbook14:~/repo/Development-Automation-Tasks/WHATSAPP_QUESTIONS$ ./problem6.sh
*****
*   *   *   *
*   *   *   *
*   *   *   *
*   *   *   *
*****
```

Shell Scripts

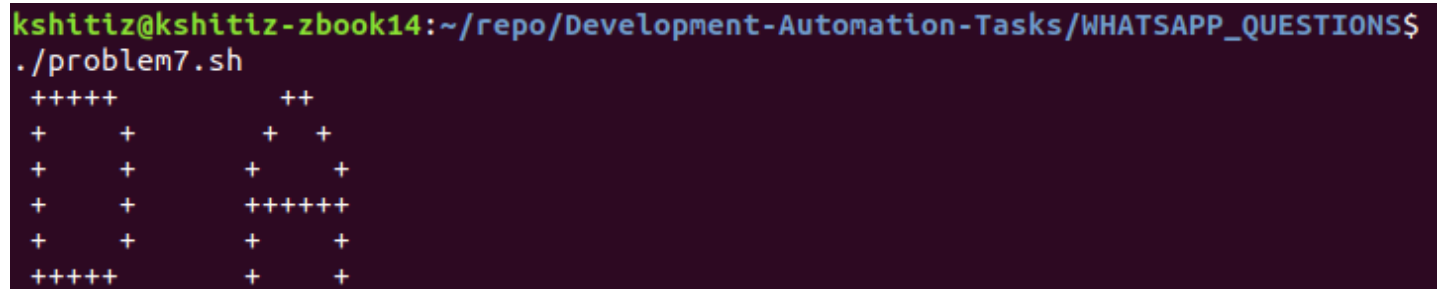
7) Shell script to print "DA" using plus (+)

Code :

```
#bin/bash
#Printing da with +
```

```
echo "+++++      ++"
echo " +      +      +  +"
echo " +      +      +    +"
echo " +      +      ++++++"
echo " +      +      +      +"
echo "+++++      +      +"
```

Output :



```
kshitiz@kshitiz-zbook14:~/repo/Development-Automation-Tasks/WHATSAPP_QUESTIONS$
./problem7.sh
+++++      ++
+      +      +  +
+      +      +    +
+      +      ++++++
+      +      +      +
+++++      +      +
```

Shell Scripts

8) Shell script to convert Hexadecimal to Octal

Algorithm :

- Find the equivalent binary number for each digit of the given hexadecimal number. Add 0's to the left if any of the binary equivalent is shorter than 4 bits.
- Separate the binary digits into groups, each containing 3 bits or digits from right to left. Add 0s to the left, if the last group contains less than 3 bits.
- Find the octal equivalent for each binary group.

Code :

```
#!/bin/bash
read -p "enter hexadecimal number: " hex
echo "octal conversion is: "
echo "ibase=16;obase=8;$hex" | bc
```

Output :

```
kshitiz@kshitiz-zbook14:~/repo/Development-Automation-Tasks/WHATSAPP_QUESTION$  
./problem8.sh  
enter hexadecimal number: 00010010010000000001  
octal conversion is:  
2000200020000000000001
```


Shell Scripts

9) Shell script to convert Octal to Hexadecimal

Algorithm :

- Find the equivalent binary number for each digit of the given hexadecimal number. Add 0's to the left if any of the binary equivalent is shorter than 4 bits.
- Separate the binary digits into groups, each containing 3 bits or digits from right to left. Add 0s to the left, if the last group contains less than 3 bits.
- Find the octal equivalent for each binary group.

Code :

```
#!/bin/bash
read -p "enter octal number: " hex
echo "hexadecimal conversion is: "
echo "obase=16;ibase=8;$hex" | bc
```

Output :

```
kshitiz@kshitiz-zbook14:~/repo/Development-Automation-Tasks/WHATSAPP_QUESTIONS$
./problem9.sh
enter octal number: 2000200020000000000001
hexadecimal conversion is:
100100100000000001
```

Shell Scripts

10) Shell script to convert Decimal to Hexadecimal

Algorithm :

- Store the remainder when the number is divided by 16 in a temporary variable temp. If temp is less than 10, insert (48 + temp) in a character array otherwise if temp is greater than or equals to 10, insert (55 + temp) in the character array.
- Divide the number by 16 now
- Repeat the above two steps until the number is not equal to 0.
- Print the array in reverse order now.

Code :

```
#!/bin/bash
read -p "enter decimal number: " hex
echo "hexadecimal conversion is: "
echo "obase=16;ibase=10;$hex" | bc
```

Output :

```
kshitiz@kshitiz-zbook14:~/repo/Development-Automation-Tasks/WHATSAPP_QUESTION$
./problem10.sh
enter decimal number: 12401
hexadecimal conversion is:
3071
```

Make and Makefile

Basic Setup

To create the needed C codes of addition, subtraction, multiplication and division and also a user defined header file mymath.h

Code :

Add:

```
#include "mymath.h"

int add(int a, int b) {
    return a + b;
}
```

Subtract:

```
#include "mymath.h"

int add(int a, int b) {
    return a - b;
}
```

Multiply:

```
#include "mymath.h"

int add(int a, int b) {
    return a * b;
}
```

Devide:

```
#include "mymath.h"

int add(int a, int b) {
    return a / b;
}
```

Main:

```
#include<stdio.h>
#include "mymath.h"

int main() {
```

```
int a, b;
printf("Hello World\n\n");
printf("Enter Two Numbers(A B): ");
scanf("%d %d", &a, &b);
printf("Addition: %d\n", add(a, b));
printf("Subtraction: %d\n", subtract(a, b));
printf("Multiplication: %d\n", multiply(a, b));
printf("Division: %d\n\n", divide(a, b));
return 0;
}
```

Mymath.h

```
int add(int a, int b);
int subtract(int a, int b);
int multiply(int a, int b);
int divide(int a, int b);
```

Description:

We have to create a calculator in modular fashion, say all independent micro services like add, subtract, divide, multiply etc. are to be created by different teams.

Thus, different functions are created along with the main() function and the header file that includes all the functions.

Make and Makefile

Building binary from the source code

To create a makefile that specifies all the files whose object is to be made and then their merged build is to be done using make command that creates the required target file.

Code :

Makefile:

```
mycalculator: main.c mymath.h add.c subtract.c multiply.c divide.c
    gcc -o mycalculator main.c mymath.h add.c subtract.c multiply.c divide.c
```

Description:

Makefile includes all the files that are to be compiled independently and then to be merged together to make the required object file mycalculator.

Output:

```
Hello Kshitiz (
kshitiz@kshitiz-zbook14:~/repo/AUTOMATION LAB/make and makefile$ ls
add.c divide.c main.c Makefile multiply.c mymath.h subtract.c
kshitiz@kshitiz-zbook14:~/repo/AUTOMATION LAB/make and makefile$ make
gcc -o mycalculator main.c mymath.h add.c subtract.c multiply.c divide.c
kshitiz@kshitiz-zbook14:~/repo/AUTOMATION LAB/make and makefile$ ls
add.c      main.c      multiply.c   mymath.h
divide.c   Makefile   mycalculator subtract.c
kshitiz@kshitiz-zbook14:~/repo/AUTOMATION LAB/make and makefile$ ./mycalculator
Hello World

Enter Two Numbers(A B): 13 14
Addition: 27
Subtraction: -1
Multiplication: 182
Division: 0

kshitiz@kshitiz-zbook14:~/repo/AUTOMATION LAB/make and makefile$
```

Make and Makefile

Transforming Makefile to adhere standard rules

Creating a dynamic Makefile that uses triggers to automatically rebuild the target file when any of sub files are changed.

Code :

Makefile:

```
INCL = mymath.h
SRC = main.c add.c subtract.c multiply.c divide.c
OBJ = $(SRC:.c=.o)
EXE = mycalculator
# Compiler, Linker Defines
CC = /usr/bin/gcc
CFLAGS = -ansi -pedantic -Wall -O2
LIBPATH = -L.
LDFLAGS = -o $(EXE) $(LIBPATH) $(LIBS)
CFDEBUG = -ansi -pedantic -Wall -g -DDEBUG $(LDFLAGS)
RM = /bin/rm -f
# Compile and Assemble C Source Files into Object Files
%.o: %.c
$(CC) -c $(CFLAGS) $*.c
# Link all Object Files with external Libraries into Binaries
$(EXE): $(OBJ)
$(CC) $(LDFLAGS) $(OBJ)
# Objects depend on these Libraries
$(OBJ): $(INCL)
# Create a gdb/dbx Capable Executable with DEBUG flags turned on
debug:
$(CC) $(CFDEBUG) $(SRC)
# Clean Up Objects, Executables, Dumps out of source directory
clean:
$(RM) $(OBJ) $(EXE) core a.out
```

Description:

Creating a standard Makefile that adheres all the rules of professional build. Also, creating dynamic Makefile that detects any changes made to the sub files and trigger automatic rebuild of the target file.

*) First, we initialize the variables with dependencies and then we command the creation of objects.

Output:

```
Hello Kshitiz (
kshitiz@kshitiz-zbook14:~/repo/AUTOMATION LAB/make and makefile$ ls
add.c      main.c      multiply.c   mymath.h    subtract.c
divide.c   Makefile   mycalculator Screenshots
kshitiz@kshitiz-zbook14:~/repo/AUTOMATION LAB/make and makefile$ make
/usr/bin/gcc -c -ansi -pedantic -Wall -O2 main.c
main.c: In function 'main':
main.c:9:2: warning: ignoring return value of 'scanf', declared with attribute warn_unused_result [-Wunused-result]
   scanf("%d %d", &a, &b);
   ^~~~~~
/usr/bin/gcc -c -ansi -pedantic -Wall -O2 add.c
/usr/bin/gcc -c -ansi -pedantic -Wall -O2 subtract.c
/usr/bin/gcc -c -ansi -pedantic -Wall -O2 multiply.c
/usr/bin/gcc -c -ansi -pedantic -Wall -O2 divide.c
/usr/bin/gcc -o mycalculator -L. main.o add.o subtract.o multiply.o divide.o
kshitiz@kshitiz-zbook14:~/repo/AUTOMATION LAB/make and makefile$ ls
add.c  divide.c  main.c  Makefile  multiply.o  mymath.h  subtract.c
add.o  divide.o  main.o  multiply.c mycalculator Screenshots subtract.o
kshitiz@kshitiz-zbook14:~/repo/AUTOMATION LAB/make and makefile$ ./mycalculator
Hello World

Enter Two Numbers(A B): 613 17
Addition: 630
Subtraction: 596
Multiplication: 10421
Division: 36
```

Make and Makefile

Archiving logs with Make

Creating logs using Makefile

Code :

Makefile:

```
SHELL = /bin/bash
LOGDIR = /var/log
LOGEXT = log
BACKUPDIR = /archives
start: createarchive movearchive deletelogs
echo "Logs files are archived and deleted"
createarchive:
tar czf archive.tar.gz $(shell find $(LOGDIR) -name ".*$(LOGEXT)")
movearchive:
mv archive.tar.gz $(BACKUPDIR)/$(shell date +%F).tar.gz
deletelogs:
rm $(shell find $(LOGDIR) -name ".*$(LOGEXT)")
```

Description:

Create a standard Makefile to create the archive file using tape archive.

Output:

```
kshitiz@kshitiz-zbook14:~/repo/AUTOMATION LAB/make and makefile (copy)$ sudo su
[sudo] password for kshitiz:
root@kshitiz-zbook14:/home/kshitiz/repo/AUTOMATION LAB/make and makefile (copy)#
make
tar czf archive.tar.gz /var/log/Xorg.1.log /var/log/vbox-setup.log /var/log/boot
strap.log /var/log/gpu-manager.log /var/log/boot.log /var/log/apt/history.log /v
ar/log/apt/term.log /var/log/dpkg.log /var/log/unattended-upgrades/unattended-up
grades-dpkg.log /var/log/unattended-upgrades/unattended-upgrades-shutdown.log /v
ar/log/unattended-upgrades/unattended-upgrades.log /var/log/teamviewer15/TeamVie
wer15_Logfile.log /var/log/teamviewer15/signaturekey.log /var/log/teamviewer15/i
ninstall_teamviewerd.log /var/log/apache2/error.log /var/log/apache2/access.log /v
ar/log/apache2/other_vhosts_access.log /var/log/Xorg.0.log /var/log/alternatives
.log /var/log/fontconfig.log /var/log/installer/casper.log /var/log/kern.log /va
r/log/mysql/error.log /var/log/apport.log /var/log/auth.log /var/log/teamviewer1
4/TeamViewer14_Logfile_OLD.log /var/log/teamviewer14/TeamViewer14_Logfile.log /v
ar/log/teamviewer14/signaturekey.log /var/log/teamviewer14/install_teamviewerd.l
og /var/log/vmware/vmware-usbarb-4967.log /var/log/vmware/vmware-usbarb-773.log
/var/log/vmware/vmware-usbarb-901.log /var/log/vmware/vmware-usbarb-791.log /var
/log/vmware/vmware-usbarb-941.log /var/log/vmware/vmware-usbarb-851.log /var/log
/vmware/vmware-usbarb-779.log /var/log/vmware/vmware-usbarb-807.log /var/log/vmw
are/hostd-653.log /var/log/vmware/hostd.log
tar: Removing leading '/' from member names
mv archive.tar.gz /archives/2019-11-25.tar.gz
mv: cannot move 'archive.tar.gz' to '/archives/2019-11-25.tar.gz': No such file
or directory
Makefile:10: recipe for target 'movearchive' failed
make: *** [movearchive] Error 1
root@kshitiz-zbook14:/home/kshitiz/repo/AUTOMATION LAB/make and makefile (copy)#
ls
add.c          divide.c  Makefile    mymath.h     subtract.c
archive.tar.gz main.c    multiply.c  Screenshots
root@kshitiz-zbook14:/home/kshitiz/repo/AUTOMATION LAB/make and makefile (copy)#
```

Make and Makefile

Conditionals in Make

Using if else condition in Makefile

Code :

Newmake.mk:

DEBUG = False

SHELL = /bin/bash

CC = /usr/bin/gcc

OBJECTS = main.o add.o subtract.o multiply.o divide.o

ifeq (\$(DEBUG),True)

CFLAG = -Wall -g

else

CFLAG =

endif

mycalculator: \${OBJECTS}

\${CC} \${CFLAG} -o \$@ \${OBJECTS}

%.o : %.c

\${CC} \${CFLAG} -o \$@ -c \$<

Output:

```
Hello Kshitiz (  
kshitiz@kshitiz-zbook14:~/repo/AUTOMATION LAB/make and makefile (3rd copy)$ ls  
add.c      main.c      mymath.h    Screenshots  
divide.c   multiply.c  newmake.mk  subtract.c  
kshitiz@kshitiz-zbook14:~/repo/AUTOMATION LAB/make and makefile (3rd copy)$ cat  
newmake.mk >> Makefile  
kshitiz@kshitiz-zbook14:~/repo/AUTOMATION LAB/make and makefile (3rd copy)$ ls  
add.c      main.c      multiply.c  newmake.mk  subtract.c  
divide.c   Makefile    mymath.h    Screenshots  
kshitiz@kshitiz-zbook14:~/repo/AUTOMATION LAB/make and makefile (3rd copy)$ make  
/usr/bin/gcc -o main.o -c main.c  
/usr/bin/gcc -o add.o -c add.c  
/usr/bin/gcc -o subtract.o -c subtract.c  
/usr/bin/gcc -o multiply.o -c multiply.c  
/usr/bin/gcc -o divide.o -c divide.c  
/usr/bin/gcc -o mycalculator main.o add.o subtract.o multiply.o divide.o  
kshitiz@kshitiz-zbook14:~/repo/AUTOMATION LAB/make and makefile (3rd copy)$ ls  
add.c      divide.o  Makefile    mycalculator  Screenshots  
add.o      main.c   multiply.c  mymath.h      subtract.c  
divide.c   main.o   multiply.o  newmake.mk    subtract.o  
kshitiz@kshitiz-zbook14:~/repo/AUTOMATION LAB/make and makefile (3rd copy)$ ./my  
calculator  
Hello World  
  
Enter Two Numbers(A B): 117 13  
Addition: 130  
Subtraction: 104  
Multiplication: 1521  
Division: 9
```