**1.MOVE FILE FROME ONE FOLDER TO THE RESPECTIVE FOLDERS.**

E.g current folder have files abc.txt def.txt ghi.txt jkl.txt

You have to move these files to the folder like abc.txt=>abc/, def.txt=>def/…

Expexted outcome –

abc/abc.txt

def/def.txt

ghi/ghi.txt

jkl/jkl.txt

1. Create files in current directory or any temporary directory – abc.txt, def.txt, ghi.txt,jkl.txt
2. Print list of the files to move
3. Segregate basename and extension of a file.
4. Create folder using basename
5. Move file too newly created folder
6. Iterate above steps for all flies.

**O/P**

#! /bin/bash

for filename in `ls \*.txt`

do

foldername=`echo $filename | awk -F.1 '{print $1}'`

mkdir $foldername

mv $filename $foldername

echo $filename "is moved to" $foldername

done

**2.Append current date to all log files name which has extension .log.1 from a folder**

E.g original file – access.log.1

New updated file name – access-20102019.log

1. Create files with name abc.log.1,def.log.1, ghi.log.1 ,jkl.log.1, mno.log.1
2. Print list of files to rename
3. Segregate basename and extension of a file
4. Print date command to show in ddmmyy
5. Append date to the log file name
6. Iterate above steps for all files which has extension .log.1

**O/P**

#! /bin/bash

for filename in `ls \*log.1`

do

foldername=`echo $filename |awk -F. '{print $1}'`

fil=`echo $filename |awk -F. '{print $2}'`

today=`date +'%d-%m-%y'|awk -F- '{print $1$2$3}'`

echo $foldername-$today-$fil

done

**3.Archive the files from /var/log folder which have modified 7days ago and move it to your backup folder**

1. Identify files which have modified time greater than 7 days
2. Move these files to the backup folder

**O/P**

#! /bin/bash

mkdir backupfolder

for file in `ls | find . -mtime -4`

do

mv $file backupfolder

echo "backup taken"

done

**4.Check if a folder exists or not. if it’s not present, create it**

1. Test if particular folder exists in current directory or not
2. If its doesn’t exists then create it else print “folder already exists..”

**O/P**

#! /bin/bash

for filename in `ls`

do

if [ -d copyfolder ]

then

echo "folder already exist"

else

mkdir copyfolder

fi

done

**5.Execute command “hello” and “ls” and check its execution status and print whether command executed successful or not.**

1. Execute “hello” command at command prompt
2. Check execution status of “hello” command
3. Execute ”ls” command at command prompt
4. Check execution status of “ls” command

**O/P**

#! /bin/bash/

echo 'hello' $?

if [ $? -eq 0 ]

then

echo "Command executed successful"

else

echo "Command not executed"

fi

echo 'ls' $?

if [ $? -eq 0 ]

then

echo "Command executed successful"

else

echo "Command not executed"

fi

**6.Create process list table display process id, parent process id, command name,% of memory consumption, % of cpu utilization**

PID PPID CMD %MEM %CPU

760 1 /usr/bin/dockerd-h unix:// 3.5 0.0

776 1 /usr/bin/containerd 0.7 0.1

7266 757 sshd:root@pts/0 0.6 0.0

759 1 /usr/sbin/rsysload-n 0.5 0.0

347 1 /usr/lib/system/system-jo 0.3 0.0

484 1 /usr/sbin/NetworkManager 0.3 0.0

1 0 /usr/lib/system/system 0.2 0.0

7268 7266 -bash 0.2 0.0

758 1 /usr/bin/python – Es /usr/sb 0.1 0.0

**O/P**

#! /bin/bash/

ps -elf |awk '{print $2,$3,$6$7}'

**7.Print last 4 frequently access urls count in sorted order from /var/log/httpd/access.log**

1. View /var/log/httpd/access.log
2. Print field which has urls data.
3. Sort extracted urls and count it
4. Print 4 unique urls

Expect sample output-

3458 /index.html

300 /api/swagger-ui.html

100 /favi.ico

20 /robots.txt

**O/P**

cat access.log| awk '{print $7}'|sort |uniq -c|sort -nr|head -4

**8.Print list of last 4 frequently access unique urls at particular hours from /var/log/httpd/access/log**

1. View access.log witout opening it using editor.
2. Print urls which has given timestamp.
3. Sort extracted urls and count it
4. Print 4 unique urls

Expect sample output-

3458 /index.html

300 /api/swagger-ui.html

100 /favi.ico

20 /robots.txt

**O/P**

cat access.log | awk -F : '{if($2==06 || $2==07)print $0}' |sort |uniq -c|sort -nr |head -4

**9.Print list of web response code count in the unique sorted order at specific hours**

1. View access.log without opening it using editor
2. Print web response code field which has given timestamp
3. Sort extracted response code and count it
4. Print 4 unique response code count

Expected sample output-

1000 200

100 304

**O/P**

cat access.log| awk '{print $9}'|sort |uniq -c|sort -nr|head -4

**10.Print list of last 10 unique sorted client IP from /var/log/httpd/access.log**

1. View access.log opening it using editor
2. Print client IP field from access log
3. Sort extracted client IP and count it
4. Print 4 unique client IPs

Expected sample output –

3635.107.181.177.135

423 27.62.203.44

45 157.44.195.138

4157.39.158.225

**O/P**

cat access.log| awk '{print $1}'|sort |uniq -c|sort -nr|head -4

**11.Data analysis / Manipulation (Awk)**

ID Employee name Job title Base pay Overtime Other pay Total pay Totalpay

Benefits

1 NATHANIEL GM 167411 0 400184 567595 567595

2 GARY CAPTAIN 155966 245131 137811 538909 538909

3 ALBERT CAPTAIN 212739 106088 16452 335279 335279

4 CHRISTOPHER MECHANIC 7916 56120 198306 332343 332343

5 PATRICK DEPUTY CHIEF 134401 9737 182234 326372 326372

6 DAVID ASST DEPUTY 118602 8601 189082 316285 316285

7 ALSON BATTALION CHIEF 92492 89062 134426 315981 315981

8 DAVID DEPUTY DIRECTOR 256576 0 51322  307899 307899

10 JOANNE CHIEF 285262 0 17115 302377 302377

1. PATRICIA CAPTAIN 99722 87082 110804 297608 297608

13 EDWARD EXECUTIVE 294580 0 0 294580 294580

i) Print Employeename and Totalpay who has Basepay greater than 10000

a)Read data file ‘data.csv’ from command line and extract rows which have Basepay>10000

**=>** cat data.csv | awk ’{ if($4>10000) print $0**}’**

b)Print only Employeename and Totalpay

**=>** cat data.csv | awk ‘{print $2,$7}’

ii) What is the aggregate Totalpay of employees whose Jobtitle is ’CAPTAIN’

a)read data file ‘data.csv/from command line and extract rows which have ‘CAPTAIN’ in the column ‘Jobtitle’

**=>** cat data.csv | awk '{if ($3=="CAPTAIN") print $0}'

b)Extract Totalpay and calculate sum. Print the result on terminal.

**=>** cat data.csv | grep CAPTAIN |awk '{if (sum+=$7) print sum}' |sort -nr |head -1

III)Print JobTitle and Overtimepay who has Overtimepay is between 7000 and 10000

a)Read data file ‘data.csv’ from command line and extract Jobtitle and Overtimepay for column value range between 7000-10000

**=>** cat data.csv | awk '{if ($5>7000 && $5<10000) print $3,$5}'

b)Print the result on terminal.

**=>** cat data.csv | awk '{if ($5>7000 && $5<10000) print $3,$5}'

iv)Print average Basepay

a)Read data file ’data.csv’ from command line and extract basepay value and calculate its average

**=>** cat data.csv |awk '{if (sum+=$4) print sum/11}' |sort -nr |head -1

b)Print the result on terminal

**=>** cat data.csv |awk '{if (sum+=$4) print sum/11}' |sort -nr |head -1