

Move files from 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/ ...

Expected outcome -

```
abc/abc.txt
def/def.txt
ghi/ghi.txt
jkl/jkl.txt
```

- a) Create files in current directory or any temporary directory - abc.txt, def.txt, ghi.txt, jkl.txt
- b) Print list of files to move.
- c) Segregate basename and extension of a file.
- d) Create folder using basename.
- e) Move file to newly created folder.
- f) Iterate above steps for all files.

A script to move all the files whose extension is .txt from a folder to a folder whose name is the same as the file's basename.

```
#!/bin/bash -x
```

```
for file in `ls *.txt`
```

```
do
```

```
    folderName=`echo $file | awk -F. '{print $1}'`
```

```
    if [ -d $folderName ]
```

```
    then
```

```
        rm -r $folderName;
```

```
    fi
```

```
    mkdir $folderName;
```

```
    mv $file $folderName;
```

```
done
```

- c) Segregate basename and extension of a file.
- d) Create folder using basename.
- e) Move file to newly created folder.
- f) Iterate above steps for all files.

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

- a) Create files with name abc.log.1, def.log.1, ghi.log.1, jkl.log.1, mno.log.1
- b) Print list of files to rename.
- c) Segregate basename and extension of a file
- d) Print Date Command to show in ddmmyy
- e) Append Date to the log file name
- f) Iterate above steps for all files which has extension .log.1

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

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

```
for file in `ls *.1`
```

```
do
```

```
    folderName=`echo $file | awk -F. '{print $1}'`
```

```
    currentDate=$(date +%d%m%Y);
```

```
    MiddleName=`echo $file | awk -F. '{print $2}'`
```

```
    if [ -d $folderName ]
```

```
    then
```

```
        rm -r $folderName;
```

```
    fi
```

```
    mkdir $folderName;
```

```
    mv $file $folderName-$currentDate.$MiddleName
```

```
    mv *.log $folderName
```

```
done
```

7) Iterate above steps for all files which has extension .log.

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

- a) Identify files which have modified time greater than 7 days
- b) Move these files to the backup folder

L I N U X C

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

```
for file in `ls *.log`  
do
```

```
    folderName=`echo $file | awk -F. '{print $1}'`;  
    currentdateAgo=$(date --date="7 days ago" +%d%m%y);  
    creationDate=`echo $file | awk -F. '{print $2}'`;  
    If [ $creationDate <= $currentdateAgo ]  
    then  
        mv $file backupDir;  
    fi
```

```
done
```

L I N U X

Check if a folder exists or not. If it's not present, create it

- a) Test if particular folder exists in current directory or not
- b) If its doesn't exists then create it else print "folder already exists.."

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

```
print "Enter the Folder Name: ";  
read folderName;  
if [ -d $folderName ]  
then  
    echo "$folderName";  
Else
```

```
mkdir $folderName  
fi
```

Execute command "hello" and "ls" and check its execution status and print whether command executed successful or not.

- a) Execute "hello" command at command prompt
- b) Check execution status of "hello" command
- c) Execute "ls" command at command prompt
- d) Check execution status of "ls" command

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

```
abc=`hello`  
echo $?  
def=`ls`  
echo $?
```

Find a word "systemd" from all log files in the folder /var/log and print number of occurrence more than 0 against each file.

- a) Use linux command to search word and print occurrence

```
#!/bin/bash -x
```

```
for file in `ls *.log`  
do  
count=`find . -name $file | xargs grep -i "systemd" | wc -l`  
  
echo $count " " $file  
  
done
```

C O M M A N D S

Data analysis / manipulation (Awk)

| Id | Employee Name | Job Title | Base Pay | Overtime Pay | Other Pay | Total Pay | TotalPayBenefits |
|----|---------------|-----------------|----------|--------------|-----------|-----------|------------------|
| 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 | 77916 | 56120 | 198306 | 332343 | 32343 |
| 5 | PATRICK | DEPUTY CHIEF | 134401 | 9737 | 182234 | 326373 | 326373 |
| 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 |
| 12 | 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
- b) Print only EmployeeName and TotalPay

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'
- b) Extract TotalPay and calculate sum. Print the result on terminal.

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
- b) Print the result on terminal.

iv) Print average BasePay

- a) Read data file 'data.csv' from command line and extract BasePay values and calculate its average
- b) Print the result on terminal.

1)

```
#!/bin/bash -x
```

```
data=`cat data.csv | awk '{if ($7 >="10000") print NR $2 " " $7}';`
echo "$data " ""
```

2)

```
#!/bin/bash -x
```

```
data=`cat data.csv |grep -i CAPT | awk '{total= total+$7}END{print total}';`
echo "$data " ""
```

3)

```
#!/bin/bash -x
```

```
data=`cat data.csv | awk '{ if($5 >= "7000" && $5 <= "10000") print NR " " $2 " $5 }'`  
echo "$data " ""
```

4)

```
#!/bin/bash -x
```

```
data=`cat data.csv |grep -i CAPT | awk '{total= total+$4}END{print total/NR}'`;  
echo "$data"
```

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

- a) View access.log without opening it using editor.
- b) Print client ip field from access log
- c) Sort extracted client IP and count it
- d) Print 4 unique client IPs

Expect sample output -

```
3635 107.181.177.135  
423 27.62.203.44  
45 157.44.195.138  
4 157.39.158.225
```

```
#!/bin/bash -x
```

```
data=`grep -o "[0-9]\{1,3\}\.[0-9]\{1,3\}\.[0-9]\{1,3\}\.[0-9]\{1,3\}" access.log | sort -n | uniq -c |  
sort -n | sort -rn | head -n 4`;
```

```
echo "$data " " ";
```

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

- a) View access.log without opening it using editor.
- b) Print web response code field which has given timestamp
- c) Sort extracted response code and count it
- d) Print 4 unique response code count

Expected sample output -

```
1000 200  
100 304
```

```
#!/bin/bash -x
```

```
data=`cat access.log | cut -d "" -f3 | cut -d ' ' -f2 | sort | uniq -c | sort -rn | head -n 2`;
```

```
echo "$data " " ";
```


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

- a) View access.log without opening it using editor.
- b) Print urls which has given timestamp.
- c) Sort extracted urls and count it
- d) Print 4 unique urls

Expect sample output -

```
3458 /index.html
300 /api/swagger-ui.html
100 /favi.ico
20 /robots.txt
```

```
#!/bin/bash -x
```

```
data=`awk -vDate=`date -d'now-2 hours' +%d/%b/%Y:%H:%M:%S` '$4 > Date {print $11}'
access.log | grep -i "http" | sort -n | uniq -c | sort -n | sort -rn | head -n 4`;
```

```
echo "$data" " " " ";
```

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

- a) View /var/log/httpd/access.log
- b) Print field which has urls data.
- c) Sort extracted urls and count it
- d) Print 4 unique urls

Expect sample output -

```
3458 /index.html
300 /api/swagger-ui.html
100 /favi.ico
20 /robots.txt
```

```
#!/bin/bash -x
```

```
data=`cat access.log | grep -i "http" | awk '{print $11}' |grep -i "http" | sort | uniq -c |sort -nr |
head -n 4`;
```

```
echo "$data" " " " ";
```

Create process list table displays 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/rsyslogd -n | 0.5 | 0.0 |
| 347 | 1 | /usr/lib/systemd/systemd-journal | 0.3 | 0.0 |
| 484 | 1 | /usr/sbin/NetworkManager | 0.3 | 0.0 |
| 1 | 0 | /usr/lib/systemd/systemd | 0.2 | 0.0 |
| 7268 | 7266 | -bash | 0.2 | 0.0 |
| 758 | 1 | /usr/bin/python -Es /usr/sbin | 0.1 | 0.0 |

Command for creating:

```
$ top | awk '{print $2 " " $4 " " $13 " " $11 " " $10}'
```

Get user info from /etc/passwd and change ownership of user's home directory (select userid higher than 1000)

- View /etc/passwd file
- Print the 1st field from /etc/passwd file
- Print all userids > 1000
- Print the 2nd field to get home directory
- Use command substitution to get user list and home directory
- Change ownership of above home directory with user which is retrieve above
- Iterate above steps for all userid > 1000

E.g /etc/passwd =>

```
ashishv:x:1002:1002::/home/ashishv:/bin/bash
```

extract values mark in bold - user and home directory

Expected output -

```
/home/ashishv:
```

```
drwx----- 6 ashishv ashishv 4096 Aug 6 12:48 ashishv
```

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

```
for user in `cat /etc/passwd`
do
```



```

UID= `cat /etc/passwd | awk -F: '{ print $3}`
if($UID -ge "1000")
then
    homeDir= `cat /etc/passwd | awk -F: '{ if($3 >= 1000) print $7}`

    Ownership= `ls -l /etc/passwd`
    chmod u+x /etc/passwd;
    changedOwInfo=`ls -l /etc/passwd | awk '{ print $6 " " $7 " " $8 " " }`
    echo " $homeDir : "
    echo " $changedOwInfo ";
fi
done

```

Find the difference between original file and the updated file.
Apply changes to the original file.

- Create two directories as "original" and "updated"
- Copy given file 'original-file.sh' to the folder "original" and "updated-file.sh" to the folder "updated"
- Find the difference between these directories using linux command
- Make copy of folder "original" to some other directory as "original-backup" and apply changes to 'original-file.sh' file
- Verify that both folders "updated" and "original-backup" have no difference.

- ```
$ Mkdir original updated
```
- ```
$ cp original-file.sh original
$ cp updated-files.sh updated
```
- ```
diff -q original/ updated/
```

Only in original/: original-file.sh  
Only in updated/: updated-files.sh
- ```
$ cp original-file.sh original-backup
$ diff -q original/ updated/
```

Only in original/: original-file.sh
Only in original/: original-backup

Only in updated/: updated-files.sh