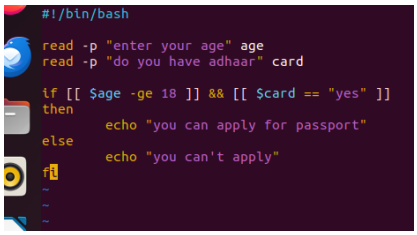
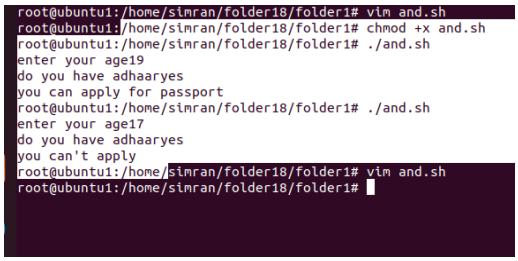
Usage of && operator

Example1:

This is the scenario where both the conditions should be true because of && operator



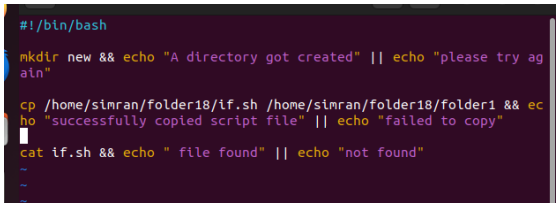
```
#!/bin/bash

read -p "enter your age" age
read -p "do you have adhaar" card

if [[ $age -ge 18 ]] && [[ $card == "yes" ]]
then
        echo "you can apply for passport"
else
        echo "you can't apply"
fi
~
~
~
```



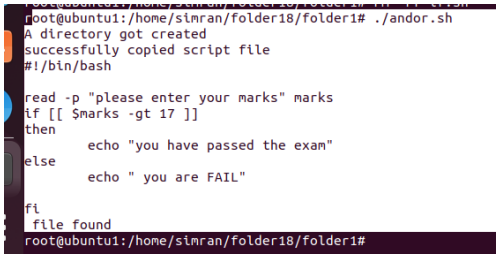
```
root@ubuntu1:/home/simran/folder18/folder1# vim and.sh
root@ubuntu1:/home/simran/folder18/folder1# chmod +x and.sh
root@ubuntu1:/home/simran/folder18/folder1# ./and.sh
enter your age19
do you have adhaaryes
you can apply for passport
root@ubuntu1:/home/simran/folder18/folder1# ./and.sh
enter your age17
do you have adhaaryes
you can't apply
root@ubuntu1:/home/simran/folder18/folder1# vim and.sh
root@ubuntu1:/home/simran/folder18/folder1#
```

Example2:

usage of && and ||(or) operator



```
#!/bin/bash

mkdir new && echo "A directory got created" || echo "please try ag
ain"

cp /home/simran/folder18/if.sh /home/simran/folder18/folder1 && ec
ho "successfully copied script file" || echo "failed to copy"

cat if.sh && echo " file found" || echo "not found"
~
~
```

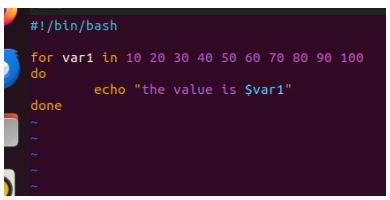


```
root@ubuntu1:/home/simran/folder18/folder1# ./andor.sh
A directory got created
successfully copied script file
#!/bin/bash

read -p "please enter your marks" marks
if [[ $marks -gt 17 ]]
then
        echo "you have passed the exam"
else
        echo " you are FAIL"

fi
 file found
root@ubuntu1:/home/simran/folder18/folder1#
```

For Loop:

Here we need loops whenever we want to run certain commands in repeatable manner

Example1: A script to print range of values:



```
#!/bin/bash

for var1 in 10 20 30 40 50 60 70 80 90 100
do
        echo "the value is $var1"
done
~
~
~
~
~
```

```
root@ubuntu1:/home/simran/folder18/folder1# vim for.sh
root@ubuntu1:/home/simran/folder18/folder1# chmod +x for.sh
root@ubuntu1:/home/simran/folder18/folder1# ./for.sh
the value is 10
the value is 20
the value is 30
the value is 40
the value is 50
the value is 60
the value is 70
the value is 80
the value is 90
the value is 100
root@ubuntu1:/home/simran/folder18/folder1#
```

Another format :

```
root@ubuntu1:/home/simran/folder18/folder1# ./for.sh
the value is 1
the value is 2
the value is 3
the value is 4
the value is 5
the value is 6
the value is 7
the value is 8
the value is 9
the value is 10
the value is 11
the value is 12
the value is 13
the value is 14
the value is 15
the value is 16
the value is 17
the value is 18
the value is 19
the value is 20
root@ubuntu1:/home/simran/folder18/folder1# cat for.sh
#!/bin/bash

for var1 in {1..20}
do
        echo "the value is $var1"
done
root@ubuntu1:/home/simran/folder18/folder1#
```

Example2: Loop through files in a directory

```
#!/bin/bash

for file in *.sh
do
        echo "the file is  $file"
done
~
~
~
~
~
```

This example demonstrates how to loop through files in the current directory with a .sh extension. The loop iterates over each matching file, and the filename is stored in the variable $file, which is then used in the loop body.

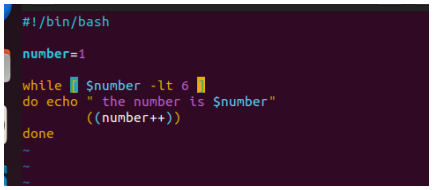
```
root@ubuntu1:/home/simran/folder18/folder1# ./for.sh
the file is  andor.sh
the file is  and.sh
the file is  for.sh
the file is  if.sh
```

Example3: Retrieving all values of array

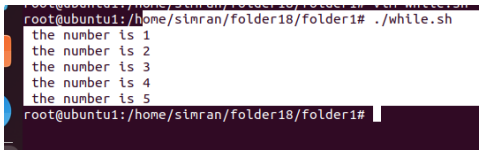
```
the element is  10
root@ubuntu1:/home/simran/folder18/folder1# cat for.sh
#!/bin/bash


list=(1 2 3 4 5 hello "hello there" 8 9 10)
length=${#list[*]}
for ((i=0;i<$length;i++))
do
        echo "the element is  ${list[$i]} "
done
root@ubuntu1:/home/simran/folder18/folder1#
```

```
root@ubuntu1:/home/simran/folder18/folder1# ./for.sh
the element is  1
the element is  2
the element is  3
the element is  4
the element is  5
the element is  hello
the element is  hello there
the element is  8
the element is  9
the element is  10
root@ubuntu1:/home/simran/folder18/folder1# cat for.sh
```

While Loop:



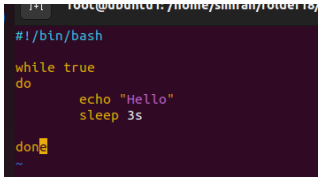
```
#!/bin/bash

number=1

while [ $number -lt 6 ]
do echo " the number is $number"
        ((number++))
done
~
~
~
```



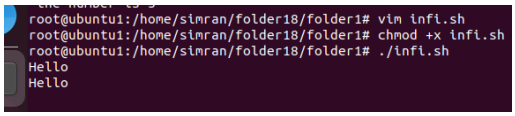
```
root@ubuntu1:/home/simran/folder18/folder1# ./while.sh
 the number is 1
 the number is 2
 the number is 3
 the number is 4
 the number is 5
root@ubuntu1:/home/simran/folder18/folder1#
```

Example: Making infinite script

After every 3 seconds it will print the text “hello”



```
#!/bin/bash

while true
do
        echo "Hello"
        sleep 3s

done
~
```

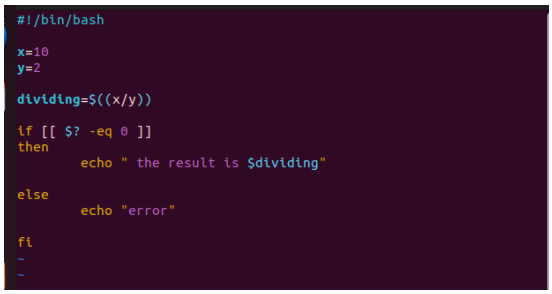


```
root@ubuntu1:/home/simran/folder18/folder1# vim infi.sh
root@ubuntu1:/home/simran/folder18/folder1# chmod +x infi.sh
root@ubuntu1:/home/simran/folder18/folder1# ./infi.sh
Hello
Hello
```

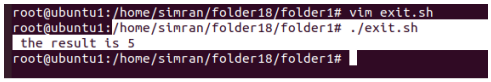
Exit Status

In Bash scripts, the exit status of a command is stored in the special variable $?. A common convention is to exit with a status of 0 for success and a non-zero status for failure.

Example:



```
#!/bin/bash

x=10
y=2

dividing=$((x/y))

if [[ $? -eq 0 ]]
then
        echo " the result is $dividing"

else
        echo "error"

fi
~
~
```



```
root@ubuntu1:/home/simran/folder18/folder1# vim exit.sh
root@ubuntu1:/home/simran/folder18/folder1# ./exit.sh
 the result is 5
root@ubuntu1:/home/simran/folder18/folder1#
```

Cronjob Command:

To automate the task on Linux we use cronjob command

Crontab -e to edit and make a new job

Crontab -l to show all current jobs

Format:

* * * * *

Minute hour day month dayofWeek

Service cron start

Service cron status

```
untu1:/home/simran/folder18/folder1# service cron start
untu1:/home/simran/folder18/folder1# service cron status
service - Regular background program processing daemon
aded: loaded (/lib/systemd/system/cron.service; enabled; vendor pr>
tive: active (running) since Thu 2023-12-07 13:37:20 IST; 1h 53min>
Docs: man:cron(8)
 PID: 638 (cron)
asks: 1 (limit: 4599)
mory: 456.0K
 CPU: 527ms
roup: /system.slice/cron.service
      └─638 /usr/sbin/cron -f -P

5:19:27 ubuntu1 CRON[13337]: (root) CMD (command -v debian-sa1 > >
5:19:27 ubuntu1 CRON[13332]: pam_unix(cron:session): session clos>
5:19:27 ubuntu1 CRON[13339]: pam_unix(cron:session): session open>
5:19:27 ubuntu1 CRON[13340]: (root) CMD (   cd / && run-parts --r>
5:19:27 ubuntu1 CRON[13339]: pam_unix(cron:session): session clos>
5:25:01 ubuntu1 CRON[13358]: pam_unix(cron:session): session open>
5:25:01 ubuntu1 CRON[13359]: (root) CMD (command -v debian-sa1 > >
5:25:01 ubuntu1 CRON[13358]: pam_unix(cron:session): session clos>
5:30:01 ubuntu1 CRON[13382]: pam_unix(cron:session): session open>
5:30:01 ubuntu1 CRON[13382]: pam_unix(cron:session): session clos>
21/21 (END)
```

To list existing cron jobs:

Crontab -l

To create a cronjob

Crontab -e

Then choose the editor:

Vim/nano

```
root@ubuntu1:/home/simran/folder18/folder1# crontab -e
no crontab for root - using an empty one

Select an editor.  To change later, run 'select-editor'.
  1. /bin/nano        <---- easiest
  2. /usr/bin/vim.basic
  3. /usr/bin/vim.tiny
  4. /bin/ed

Choose 1-4 [1]: 2
crontab: installing new crontab
```

```
* * * * * echo "newfile" >> /home/simran/folder18/file111
root@ubuntu1:/home/simran/folder18#
```

When we have used *** it says after every minute this text will be added in this file “file111”**

```
root@ubuntu1:/home/simran/folder18# cat file111
newfile
newfile
newfile
newfile
root@ubuntu1:/home/simran/folder18#
```