Universitat Politècnica De Catalunya

Escuela Técnica Superior de Ingeniería de Telecomunicación de Barcelona

ADMINISTRACIÓ DE SISTEMES LINUX

Autors:
Gokarna Dumre & Jaume Quero

May 6, 2023

Contents

1 Introducció

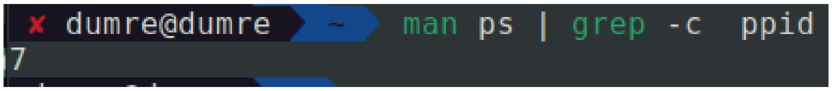
In this Seminar we are going to work in Linux basics. The main objective is to enhance our Linux knowledge to understand and to be capable to face Linux issues or challenges.

2 Chapter 2: Processes

2.1 Exercise 2.1

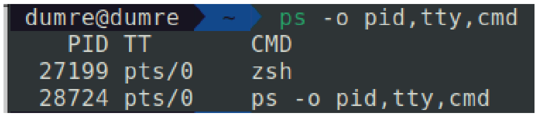
In this exercise you will practice with process execution and signals.

1. Open a pseudo-terminal and execute the command to see the manual of ps. Once in the manual of the ps command, search and count the number of times that appears the pattern ppid.

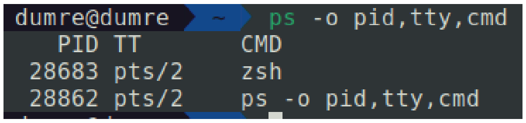


```
✘ dumre@dumre  ~  man ps | grep -c  ppid
7
```

2. Within the same pseudo-terminal, execute ps with the appropriate parameters in order to show the PID, the tty and the command of the currently active processes that have been executed from the terminal. Do the same in the virtual console number two (/dev/tty2).



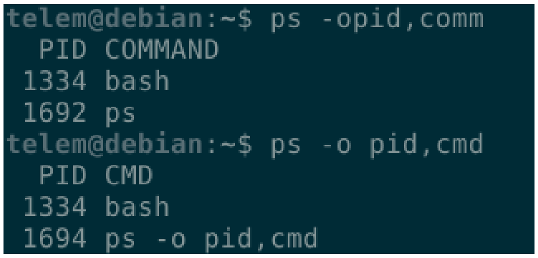
```
dumre@dumre  ~  ps -o pid,tty,cmd
  PID TT       CMD
27199 pts/0    zsh
28724 pts/0    ps -o pid,tty,cmd
```



```
dumre@dumre  ~  ps -o pid,tty,cmd
  PID TT       CMD
28683 pts/2    zsh
28862 pts/2    ps -o pid,tty,cmd
```

3. Execute the following commands:
   $ps -o pid,comm
   $ps -o pid,commd

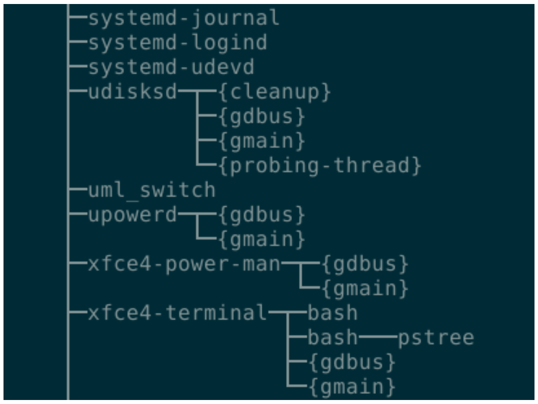
   Comment the differences between the options: cmd and comm



```
telem@debian:~$ ps -opid,comm
  PID COMMAND
 1334 bash
 1692 ps
telem@debian:~$ ps -o pid,cmd
  PID CMD
 1334 bash
 1694 ps -o pid,cmd
```

*The main difference is that **cmd** display the process ID and the command used to start the process, but **comm** displays the process ID (pid) and the name of the command (comm) for each process.*

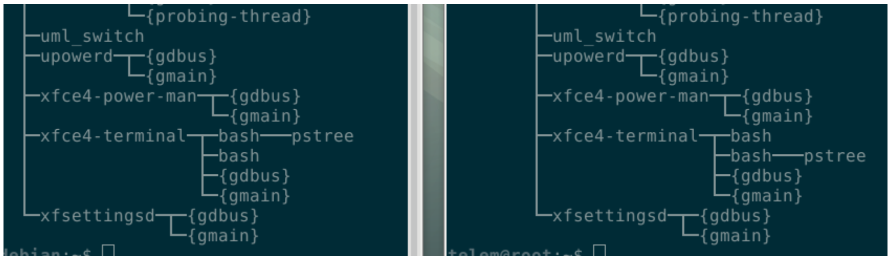
4. Use the pstree command to see the process tree of the system. Which process is the father of pstree? and its grandfather? and who are the rest of its relatives?



```
 ├─systemd-journal
 ├─systemd-logind
 ├─systemd-udevd
 ├─udisksd─┬─{cleanup}
 │         ├─{gdbus}
 │         ├─{gmain}
 │         └─{probing-thread}
 ├─uml_switch
 ├─upowerd─┬─{gdbus}
 │         └─{gmain}
 ├─xfce4-power-man─┬─{gdbus}
 │                 └─{gmain}
 ├─xfce4-terminal─┬─bash
 │                ├─bash───pstree
 │                ├─{gdbus}
 │                └─{gmain}
```

The process father is ***bash*** *and the grandfather is* ***xfce4-terminal*** *and the relatives are the rest.*

5. Open a gnome-terminal and then open a new "TAB" typing ctrl+shit+t. Now open another gnome-terminal in a new window. Using pstree, you have to comment the relationships between the processes related to the terminals that we opened.

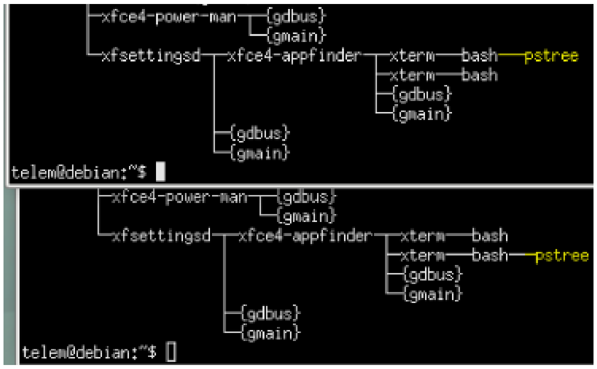


```
 │                 └─{probing-thread}
 ├─uml_switch
 ├─upowerd─┬─{gdbus}
 │         └─{gmain}
 ├─xfce4-power-man─┬─{gdbus}
 │                 └─{gmain}
 ├─xfce4-terminal─┬─bash───pstree
 │                ├─bash
 │                ├─{gdbus}
 │                └─{gmain}
 └─xfsettingsd─┬─{gdbus}
               └─{gmain}
debian:~$ 
```

```
 │                 └─{probing-thread}
 ├─uml_switch
 ├─upowerd─┬─{gdbus}
 │         └─{gmain}
 ├─xfce4-power-man─┬─{gdbus}
 │                 └─{gmain}
 ├─xfce4-terminal─┬─bash
 │                ├─bash───pstree
 │                ├─{gdbus}
 │                └─{gmain}
 └─xfsettingsd─┬─{gdbus}
               └─{gmain}
tolem@root:~$ 
```

In each tab there is a different bash process. In the first tab, the process is same as previous exercise. But in the second tab, the process is new.

6. Type ALT+F2 and then type xterm. Notice that this sequence opens another type of terminal. Repeat the same sequence to open a new xterm. Now, view the process tree and comment the differences with respect to the results of the previous case of gnome terminals.



```
  ├─xfce4-power-man─┬─{gdbus}
  │                 └─{gmain}
  └─xfsettingsd─┬─xfce4-appfinder─┬─xterm───bash───pstree
                │                 ├─xterm───bash
                │                 ├─{gdbus}
                │                 └─{gmain}
                ├─{gdbus}
                └─{gmain}
telem@debian:~$ 
```

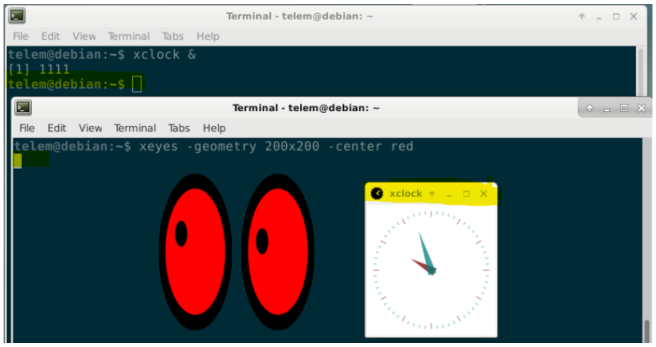
```
  ├─xfce4-power-man─┬─{gdbus}
  │                 └─{gmain}
  └─xfsettingsd─┬─xfce4-appfinder─┬─xterm───bash
                │                 ├─xterm───bash───pstree
                │                 ├─{gdbus}
                │                 └─{gmain}
                ├─{gdbus}
                └─{gmain}
telem@debian:~$ 
```

The situation is identical as in previous exercise, each xterm process is working in their individual bash.

7. Open three gnome-terminals. These will be noted as t1, t2 and t3. Then, type the following:

```
t1$ xeyes -geometry 200x200 -center red
t2$ xclock &
```

Comment what you see and also which is the type of execution (foreground/background) on each terminal.

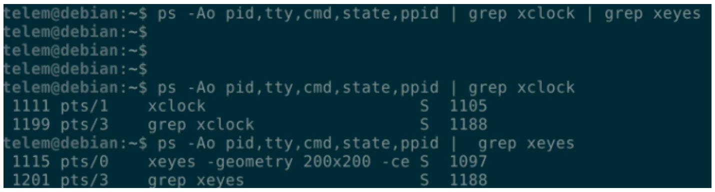

Terminal - telem@debian: ~
File Edit View Terminal Tabs Help
Terminal - telem@debian: ~
File Edit View Terminal Tabs Help
xclock

```
telem@debian:~$ xclock &
[1] 1111
telem@debian:~$ 
```

```
telem@debian:~$ xeyes -geometry 200x200 -center red
```

The first command is runing in foreground and the second one is being executed in background. As we can see in the image, we don't have any option to kill the eyes, but we can close, minimize or maximizee the clock.

8. For each process of the previous applications (xeyes and xclock), try to find out the PID, the execution state, the tty and the parent PID (PPID). To do so, use the third terminal (t3).



```
telem@debian:~$ ps -Ao pid,tty,cmd,state,ppid | grep xclock | grep xeyes
telem@debian:~$
telem@debian:~$
telem@debian:~$
telem@debian:~$ ps -Ao pid,tty,cmd,state,ppid | grep xclock
 1111 pts/1    xclock                      S  1105
 1199 pts/3    grep xclock                 S  1188
telem@debian:~$ ps -Ao pid,tty,cmd,state,ppid |  grep xeyes
 1115 pts/0    xeyes -geometry 200x200 -ce S  1097
 1201 pts/3    grep xeyes                  S  1188
```

Using the command that we saw in the beginning we obtain the PID, the State, the tty & the PPID.

9. Using the third terminal (t3), send a signal to terminate the process xeyes.

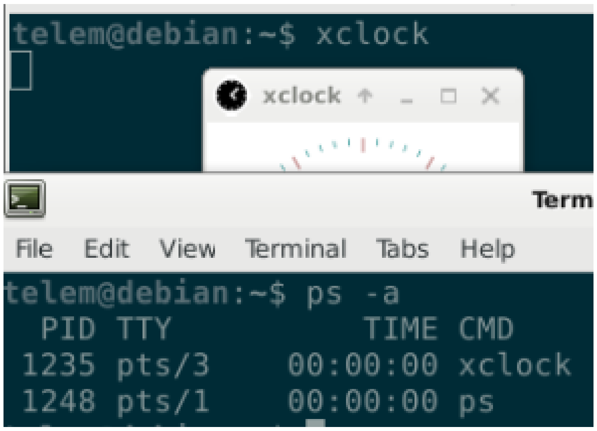
```
kill -9 1115 #pid of xeyes
```

10. Type exit in the terminal t2. After that, find out who is the parent process of xclock.
Atfer killing t2 and doing ps we can't know tty of xclock process and now the ppid is different.

11. Now send a signal to kill the process xclock using its PID.

```
kill -9 1111 #pid of xclock
```

12. Execute an xclock in foreground in the first terminal t1.



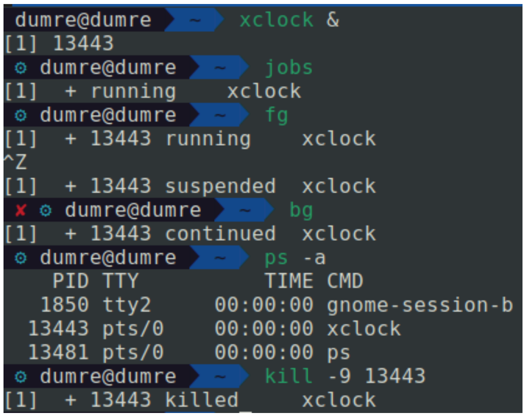
```
telem@debian:~$ xclock
```

```
telem@debian:~$ ps -a
  PID TTY          TIME CMD
 1235 pts/3    00:00:00 xclock
 1248 pts/1    00:00:00 ps
```

13. Send a signal from the third terminal to stop the process xclock and then send another signal to let this process to continue executing. Is the process executing in foreground or in background? Finally, send a signal to terminate the xclock process.

```
kill -SIGSTOP 1235
kill -SIGCONT 1235
kill -SIGKILL 1235 #kill -9 1235
```

14. Using the job control, repeat the same steps as before, that is, executing xclock in foreground and then stopping, resuming and killing. List the commands and the key combinations you have used.



```
dumre@dumre  ~  xclock &
[1] 13443
⚙ dumre@dumre  ~  jobs
[1]  + running    xclock
⚙ dumre@dumre  ~  fg
[1]  + 13443 running    xclock
^Z
[1]  + 13443 suspended  xclock
✘ ⚙ dumre@dumre  ~  bg
[1]  + 13443 continued  xclock
⚙ dumre@dumre  ~  ps -a
    PID TTY          TIME CMD
   1850 tty2     00:00:00 gnome-session-b
  13443 pts/0    00:00:00 xclock
  13481 pts/0    00:00:00 ps
⚙ dumre@dumre  ~  kill -9 13443
[1]  + 13443 killed     xclock
```

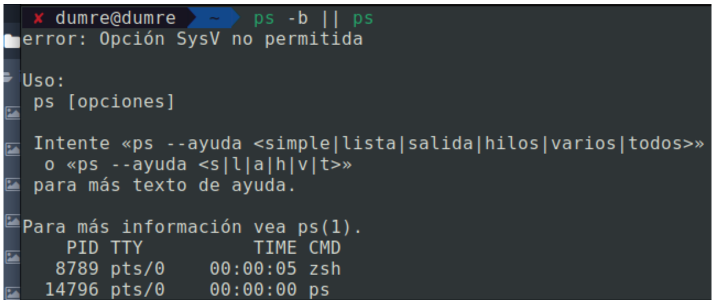
15. Execute the following commands in a pseudo-terminal:
Using the job control set the first xclock in foreground. Then place it back in background. Kill by name the two xclock processes and then the xeyes processes. List the commands and the key combinations you have used.

```
xclock &
xclock &
xeyes &
xeyes &
jobs
fg %1                #xclock toforeground
bg %1                #return to background
kill -s SIGTERM %1   #kill 1st xclock
kill -s SIGTERM %2   #kill 2n xclock
kill -s SIGTERM %3   #kill 1st xeyes
kill -s SIGTERM %4   #kill 2n xyes
```

16. Create a command line using execution of multiple commands that shows the processes launched from the terminal, then waits for 3 seconds and finally shows again the processes launched from the terminal.

```
ps; sleep 3; ps
```

17. Using multiple commands execution (&&, ||, etc.) create a command line that executes a ps command with an unsuccessful exit state and then as a result another ps command without parameters is executed.



```
x dumre@dumre  ~  ps -b || ps
error: Opción SysV no permitida

Uso:
 ps [opciones]

 Intente «ps --ayuda <simple|lista|salida|hilos|varios|todos>»
  o «ps --ayuda <s|l|a|h|v|t>»
 para más texto de ayuda.

Para más información vea ps(1).
    PID TTY          TIME CMD
   8789 pts/0    00:00:05 zsh
  14796 pts/0    00:00:00 ps
```

18. Discuss the results of the following multiple command executions:

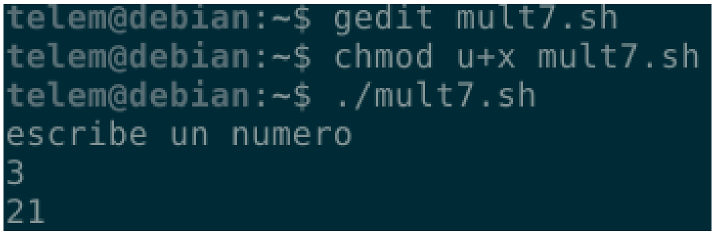
```
$ sleep || sleep || ls
#1st & 2nd sleep are lacked of parameter (time). only ls is
    executed.

$ sleep && sleep --help || ls && ps
#sleep is same es previous command but now the 2nd sleep is
    correct but there is the and condition, so only ls and ps are
    executed.

$ sleep && sleep --help || ls || ps
#sleep is same es previous command but now only ls is executed
    because the condition is or.
```

2.2 Exercise 2.2

This exercise deals with additional aspects about processes.

1. Create a script that asks for a number and displays the number multiplied by 7. Note. If you use the variable VAR to read, you can use $[VAR*7] to display its multiplication.

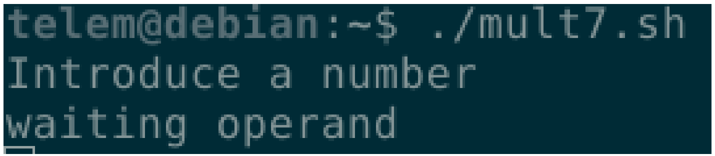
```
#Script code
echo Introduce a number
read NUMERO
let "RESULT = $NUMERO*7"
echo $RESULT
```



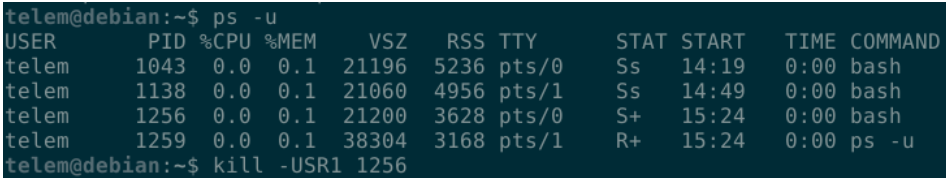
```
telem@debian:~$ gedit mult7.sh
telem@debian:~$ chmod u+x mult7.sh
telem@debian:~$ ./mult7.sh
escribe un numero
3
21
```

2. Add signal managment to the previous script so that when the USR1 signal is received, the script prints the sentence “waiting operand”. Tips: use trap to capture USR1 and kill -USR1 PID to send this signal.

```
#Script Code
trap "echo waiting operand" USR1
echo Introduce a number
read NUMERO
let "RESULT = $NUMERO*7"
echo $RESULT
```

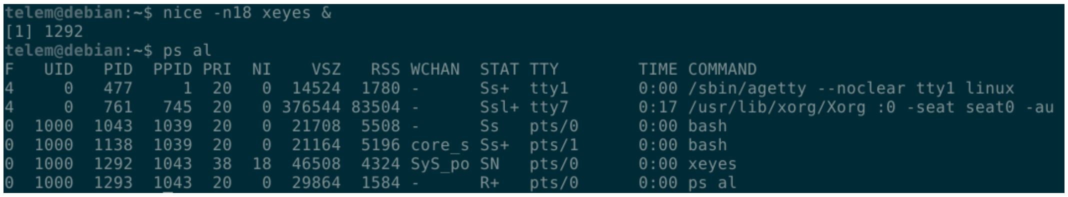


```
telem@debian:~$ ./mult7.sh
Introduce a number
waiting operand
```



```
telem@debian:~$ ps -u
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
telem     1043  0.0  0.1  21196  5236 pts/0    Ss   14:19   0:00 bash
telem     1138  0.0  0.1  21060  4956 pts/1    Ss   14:49   0:00 bash
telem     1256  0.0  0.1  21200  3628 pts/0    S+   15:24   0:00 bash
telem     1259  0.0  0.1  38304  3168 pts/1    R+   15:24   0:00 ps -u
telem@debian:~$ kill -USR1 1256
```

3. Type a command to execute an xeyes application in background with “niceness” (priority) equal to 18. Then, type a command to view the command, the PID and the priority of the xeyes process that you have just executed.



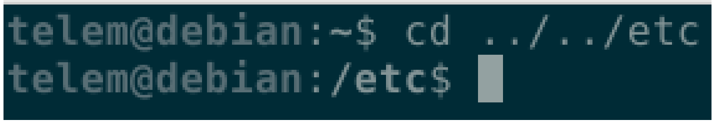
```
telem@debian:~$ nice -n18 xeyes &
[1] 1292
telem@debian:~$ ps al
F   UID   PID  PPID PRI  NI    VSZ   RSS WCHAN  STAT TTY        TIME COMMAND
4     0   477     1  20   0  14524  1780 -      Ss+  tty1       0:00 /sbin/agetty --noclear tty1 linux
4     0   761   745  20   0 376544 83504 -      Ssl+ tty7       0:17 /usr/lib/xorg/Xorg :0 -seat seat0 -au
0  1000  1043  1039  20   0  21708  5508 -      Ss   pts/0      0:00 bash
0  1000  1138  1039  20   0  21164  5196 core_s Ss+  pts/1      0:00 bash
0  1000  1292  1043  38  18  46508  4324 SyS_po SN   pts/0      0:00 xeyes
0  1000  1293  1043  20   0  29864  1584 -      R+   pts/0      0:00 ps al
```

3 Chapter 3: Filesystem

3.1 Exercise 3.1

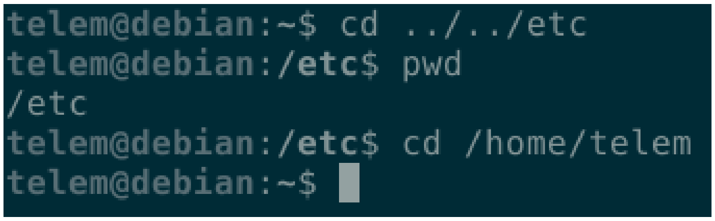
This exercise is related to the Linux filesystem and its basic permission system.

1. Open a terminal and navigate to your home directory (type cd or simply cd). Then, type the command that using a relative path changes your location into the directory /etc.



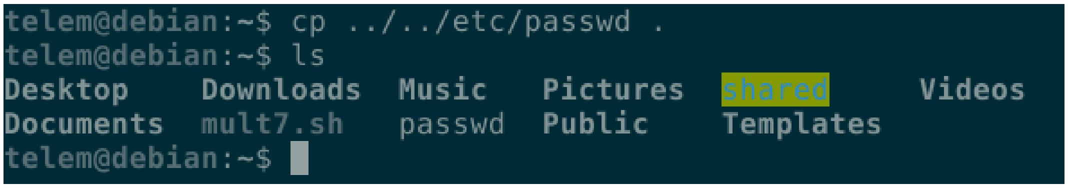
```
telem@debian:~$ cd ../../etc
telem@debian:/etc$ 
```

2. Type the command to return to home directory using an absolute path.



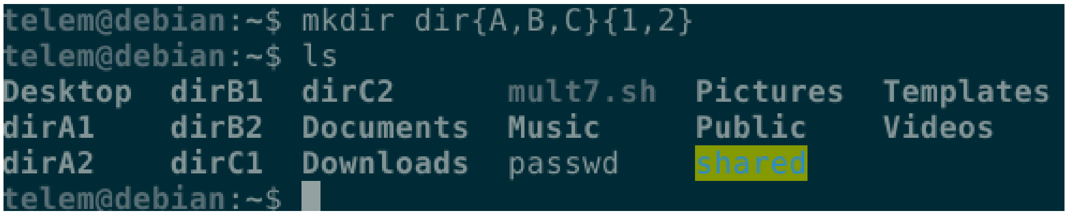
```
telem@debian:~$ cd ../../etc
telem@debian:/etc$ pwd
/etc
telem@debian:/etc$ cd /home/telem
telem@debian:~$ 
```

3. Once at your home directory, type a command to copy the file /etc/passwd in your working directory using only relative paths.



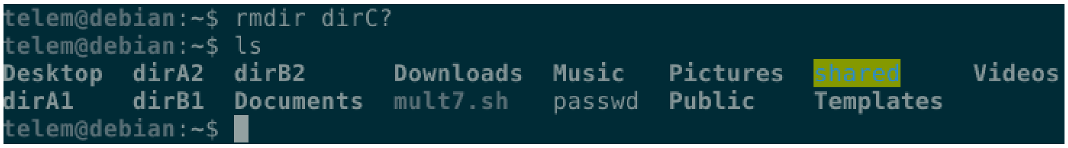
```
telem@debian:~$ cp ../../etc/passwd .
telem@debian:~$ ls
Desktop     Downloads  Music    Pictures  shared     Videos
Documents   mult7.sh   passwd   Public    Templates
telem@debian:~$ 
```

4. Create six directories named: dirA1, dirA2, dirB1, dirB2, dirC1 and dirC2 inside your home directory. You can do this with the command:



```
telem@debian:~$ mkdir dir{A,B,C}{1,2}
telem@debian:~$ ls
Desktop  dirB1  dirC2      mult7.sh  Pictures  Templates
dirA1    dirB2  Documents  Music     Public    Videos
dirA2    dirC1  Downloads  passwd    shared
telem@debian:~$ 
```

5. Delete directories dirC2 and dirC1 using the wildcard "?".



```
telem@debian:~$ rmdir dirC?
telem@debian:~$ ls
Desktop  dirA2  dirB2      Downloads  Music   Pictures  shared     Videos
dirA1    dirB1  Documents  mult7.sh   passwd  Public    Templates
telem@debian:~$ 
```

6. Create an empty file in your working directory called temp.

```
#We use the command
touch temp
```

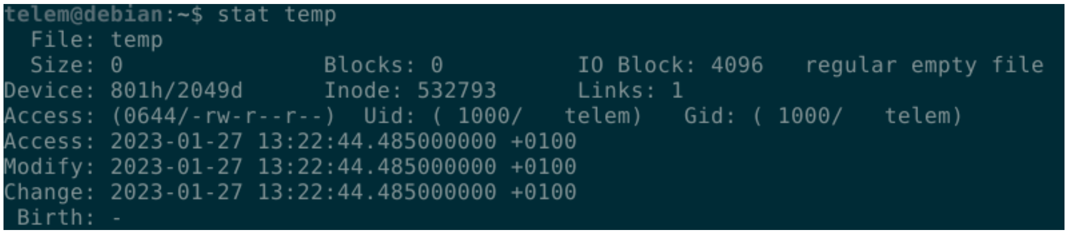
7. Type a command for viewing text to display the contents of the file, which obviously must be empty.

```
#We use the command
cat temp
```

8. Type a command to display the file metadata and properties (creation date, modification date, last access date, inode etc.).

```
#We use the command
stat temp
```

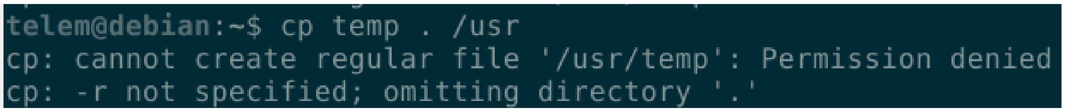
9. What kind of content is shown for the temp? and what kind basic file is?



```
telem@debian:~$ stat temp
  File: temp
  Size: 0               Blocks: 0          IO Block: 4096   regular empty file
Device: 801h/2049d      Inode: 532793      Links: 1
Access: (0644/-rw-r--r--)  Uid: ( 1000/   telem)   Gid: ( 1000/   telem)
Access: 2023-01-27 13:22:44.485000000 +0100
Modify: 2023-01-27 13:22:44.485000000 +0100
Change: 2023-01-27 13:22:44.485000000 +0100
 Birth: -
```

Shows the file properties and we can see its a regular empty file.

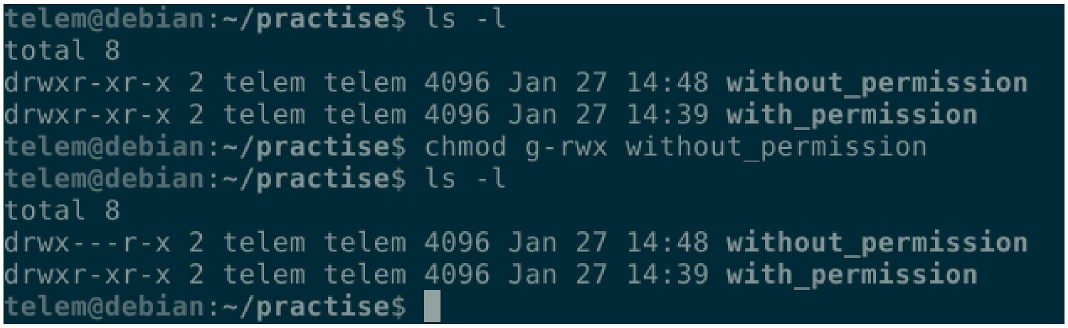
10. Change to your working directory. From there, type a command to try to copy the file temp to the /usr directory. What happened and why?



```
telem@debian:~$ cp temp . /usr
cp: cannot create regular file '/usr/temp': Permission denied
cp: -r not specified; omitting directory '.'
```

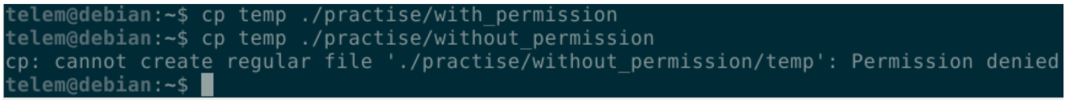
It can't be copied because its not in our personal area, we need to be superuser.

11. Create a directory called practices inside your home. Inside practices, create two directories called with permission and without permission. Then, remove your own permission to write into the directory without permission.



```
telem@debian:~/practise$ ls -l
total 8
drwxr-xr-x 2 telem telem 4096 Jan 27 14:48 without_permission
drwxr-xr-x 2 telem telem 4096 Jan 27 14:39 with_permission
telem@debian:~/practise$ chmod g-rwx without_permission
telem@debian:~/practise$ ls -l
total 8
drwx---r-x 2 telem telem 4096 Jan 27 14:48 without_permission
drwxr-xr-x 2 telem telem 4096 Jan 27 14:39 with_permission
telem@debian:~/practise$ 
```

12. Try to copy the temp file to the directories with permission and without permission. Explain what has happened in each case and why.



```
telem@debian:~$ cp temp ./practise/with_permission
telem@debian:~$ cp temp ./practise/without_permission
cp: cannot create regular file './practise/without_permission/temp': Permission denied
telem@debian:~$ 
```

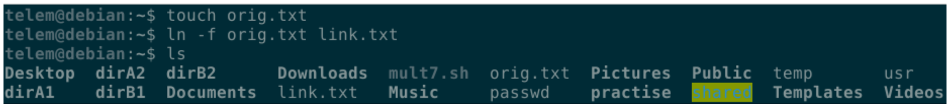
13. Figure out which is the minimum set of permissions (read, write, execute) that the owner has to have to execute the following commands

Commands	read	write	execute
`cd without_permission`	X		X
`cd without_permission; ls -l`	X		X
`cp temp ~/practices/without_permission`	X	X	X

3.2 Exercise 3.2

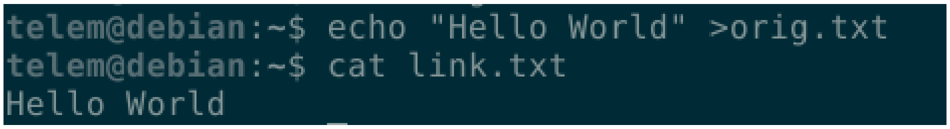
This exercise presents practices about text files and special files.

1. Create a file called orig.txt with the touch command and use the command ln to create a symbolic link to orig.txt called link.txt. Open the vi text editor and modify the file orig.txt entering some text.



```
telem@debian:~$ touch orig.txt
telem@debian:~$ ln -f orig.txt link.txt
telem@debian:~$ ls
Desktop  dirA2  dirB2      Downloads  mult7.sh  orig.txt  Pictures  Public  temp       usr
dirA1    dirB1  Documents  link.txt   Music     passwd    practise  shared  Templates  Videos
```

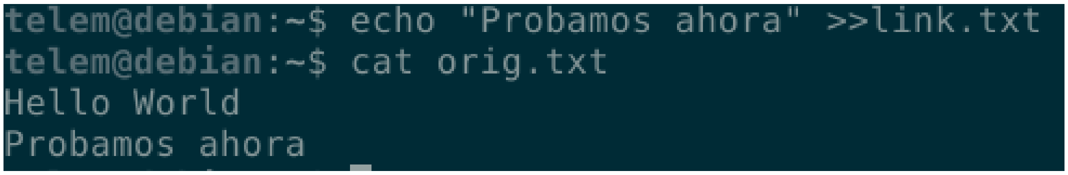
2. Use the command cat to view link.txt. What can you observe? why?.



```
telem@debian:~$ echo "Hello World" >orig.txt
telem@debian:~$ cat link.txt
Hello World
```

The things we write on orig.txt will be also at link.txt.

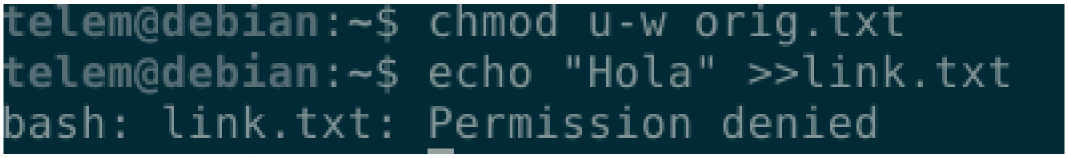
3. Repeat previous two steps but this time modifying first the link.txt file and then viewing the orig.txt file. Discuss the results.



```
telem@debian:~$ echo "Probamos ahora" >>link.txt
telem@debian:~$ cat orig.txt
Hello World
Probamos ahora
```

As before both files are linked so every thing we do in one, will apear on the other.

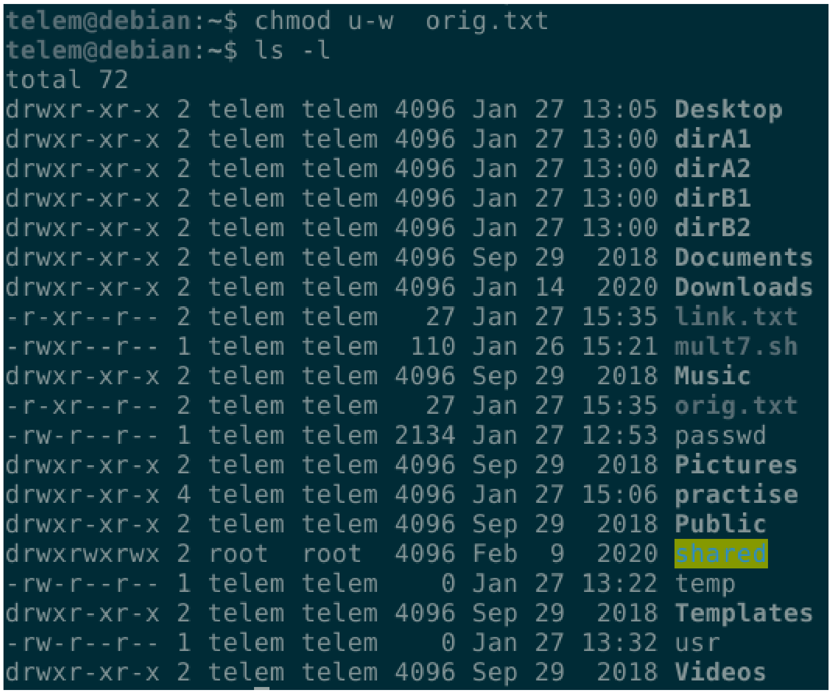
4. Remove all permissions from orig.txt and try to modify the link.txt file. What happened?



```
telem@debian:~$ chmod u-w orig.txt
telem@debian:~$ echo "Hola" >>link.txt
bash: link.txt: Permission denied
```

We are not allowed to modify it.

5. Give back the write permission to orig.txt. Then, try to remove the write permission to link.txt. Type ls -l and discuss the results.

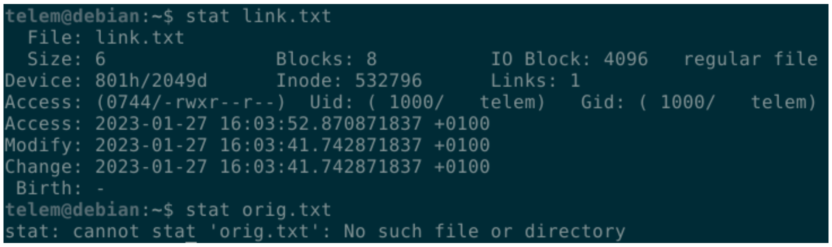


```
telem@debian:~$ chmod u-w  orig.txt
telem@debian:~$ ls -l
total 72
drwxr-xr-x 2 telem telem 4096 Jan 27 13:05 Desktop
drwxr-xr-x 2 telem telem 4096 Jan 27 13:00 dirA1
drwxr-xr-x 2 telem telem 4096 Jan 27 13:00 dirA2
drwxr-xr-x 2 telem telem 4096 Jan 27 13:00 dirB1
drwxr-xr-x 2 telem telem 4096 Jan 27 13:00 dirB2
drwxr-xr-x 2 telem telem 4096 Sep 29  2018 Documents
drwxr-xr-x 2 telem telem 4096 Jan 14  2020 Downloads
-r-xr--r-- 2 telem telem   27 Jan 27 15:35 link.txt
-rwxr--r-- 1 telem telem  110 Jan 26 15:21 mult7.sh
drwxr-xr-x 2 telem telem 4096 Sep 29  2018 Music
-r-xr--r-- 2 telem telem   27 Jan 27 15:35 orig.txt
-rw-r--r-- 1 telem telem 2134 Jan 27 12:53 passwd
drwxr-xr-x 2 telem telem 4096 Sep 29  2018 Pictures
drwxr-xr-x 4 telem telem 4096 Jan 27 15:06 practise
drwxr-xr-x 2 telem telem 4096 Sep 29  2018 Public
drwxrwxrwx 2 root  root  4096 Feb  9  2020 shared
-rw-r--r-- 1 telem telem    0 Jan 27 13:22 temp
drwxr-xr-x 2 telem telem 4096 Sep 29  2018 Templates
-rw-r--r-- 1 telem telem    0 Jan 27 13:32 usr
drwxr-xr-x 2 telem telem 4096 Sep 29  2018 Videos
```

We can see that both files had modified its permissions.

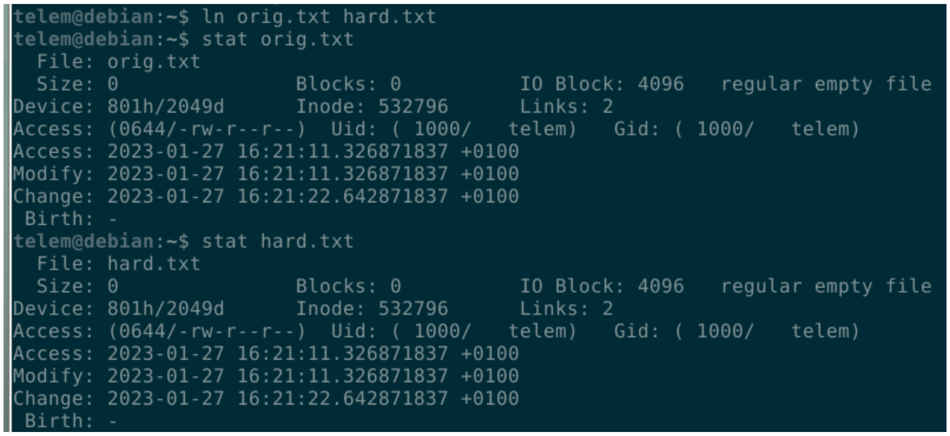
6. Delete the file orig.txt and try to display the contents of link.txt with the cat command. Then, in a terminal (t1) edit orig.txt with the command:
   Did not happened anything because we had removed the file.

7. Use the command stat to see the number of links that orig.txt and link.txt have.



```
telem@debian:~$ stat link.txt
  File: link.txt
  Size: 6               Blocks: 8          IO Block: 4096   regular file
Device: 801h/2049d      Inode: 532796      Links: 1
Access: (0744/-rwxr--r--)  Uid: ( 1000/   telem)   Gid: ( 1000/   telem)
Access: 2023-01-27 16:03:52.870871837 +0100
Modify: 2023-01-27 16:03:41.742871837 +0100
Change: 2023-01-27 16:03:41.742871837 +0100
 Birth: -
telem@debian:~$ stat orig.txt
stat: cannot stat 'orig.txt': No such file or directory
```

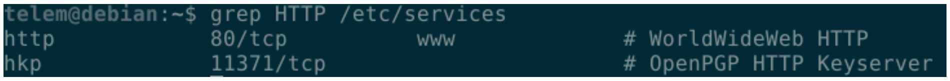
8. Now create a hard link for the orig.txt file called hard.txt. Then, using the command stat figure out the number of "Links" of orig.txt and hard.txt.



```
telem@debian:~$ ln orig.txt hard.txt
telem@debian:~$ stat orig.txt
  File: orig.txt
  Size: 0               Blocks: 0          IO Block: 4096   regular empty file
Device: 801h/2049d      Inode: 532796      Links: 2
Access: (0644/-rw-r--r--)  Uid: ( 1000/   telem)   Gid: ( 1000/   telem)
Access: 2023-01-27 16:21:11.326871837 +0100
Modify: 2023-01-27 16:21:11.326871837 +0100
Change: 2023-01-27 16:21:22.642871837 +0100
 Birth: -
telem@debian:~$ stat hard.txt
  File: hard.txt
  Size: 0               Blocks: 0          IO Block: 4096   regular empty file
Device: 801h/2049d      Inode: 532796      Links: 2
Access: (0644/-rw-r--r--)  Uid: ( 1000/   telem)   Gid: ( 1000/   telem)
Access: 2023-01-27 16:21:11.326871837 +0100
Modify: 2023-01-27 16:21:11.326871837 +0100
Change: 2023-01-27 16:21:22.642871837 +0100
 Birth: -
```

9. Delete the file orig.txt and try to modify with vi hard.txt. What happened?
   The hard.txt could have been edited but orig.txt not.

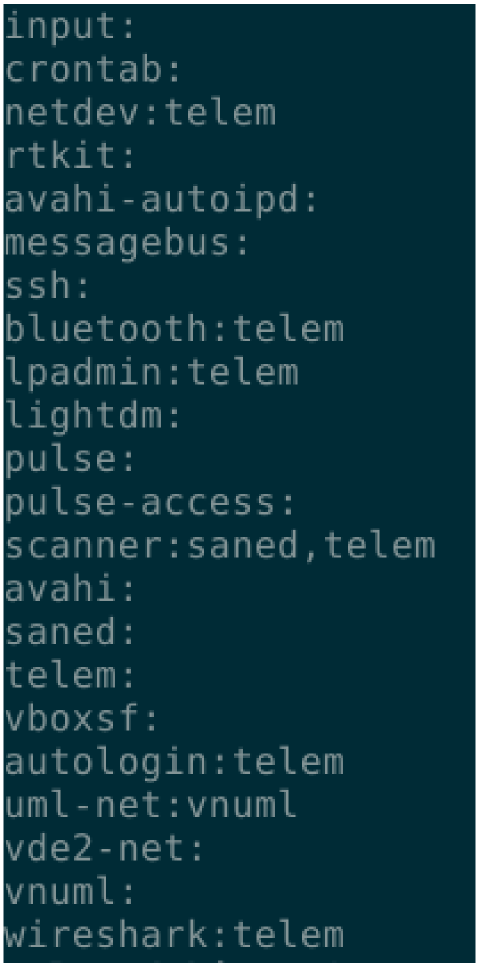
10. Use the grep command to find all the information about the HTTP protocol present in the file /etc/services (remember that Unix commands are case-sensitive).



```
telem@debian:~$ grep HTTP /etc/services
http            80/tcp          www             # WorldWideWeb HTTP
hkp             11371/tcp                       # OpenPGP HTTP Keyserver
```

11. Use the cut command over the file /etc/group to display the name of each group and its members (last field).

```
cut -d ":" -f 1,4 /etc/group
```

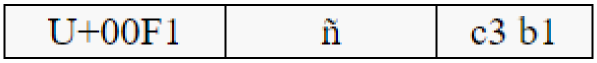


```
input:
crontab:
netdev:telem
rtkit:
avahi-autoipd:
messagebus:
ssh:
bluetooth:telem
lpadmin:telem
lightdm:
pulse:
pulse-access:
scanner:saned,telem
avahi:
saned:
telem:
vboxsf:
autologin:telem
uml-net:vnuml
vde2-net:
vnuml:
wireshark:telem
```

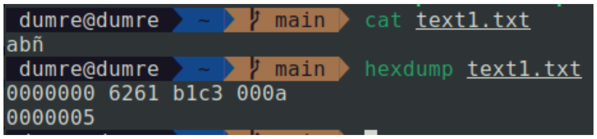
12. Create an empty file called text1.txt. Use text editor vi abn to introduce "abñ" in the file, save and exit. Type a command to figure out the type of content of the file.

```
nano text1.txt
file text1.txt
#text1.txt: Unicode text, UTF-8 text
```

13. Search in the Web the hexadecimal encoding of the letter “ñ” in ISO-8859-15 and UTF8. Use the command hexdump to view the content in hexadecimal of text1.txt. Which encoding have you found?
The character ñ is in UTF8, and the bits are inverted. The ñ its the b1c3 as we can see in the image.



U+00F1	ñ	c3 b1



```
 dumre@dumre  ~  main  cat text1.txt
abñ
 dumre@dumre  ~  main  hexdump text1.txt
0000000 6261 b1c3 000a
0000005
```

14. Find out what the character is “0x0a“, which also appears in the file.
0x0A is LF (Line feed, ”\n“, 10 in decimal). We can see figure the UTF8 enconding of abñ and linebreak which is 0x0a.

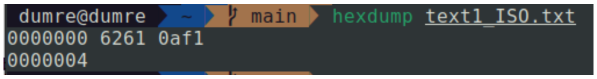
15. Open the gedit text editor and type ”abñ“. Go to the menu and use the option “Save As” to save the file with the name text2.txt and “Line Ending” type Windows. Again with the hexdump examine the contents of the file. Find out which is the character encoded as “0x0d”.

```
hexdump text1.txt  #linux
0000000 6261 b1c3 000a
0000005

hexdump text2.txt #windows
0000000 6261 b1c3 0a0d   #0d=Carriage Return,
0000006

hexdump text3.txt #classic mac
0000000 6261 b1c3 000d
0000005
```

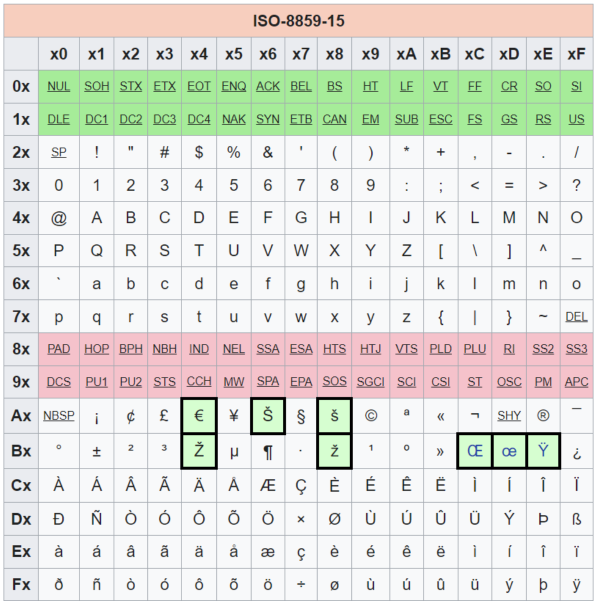
The document control character has changed, since it is saved as Windows. And then on mac Line Ending also we can appreciate the change. And the 0x0d is referred to the carriage return.

16. Explain the different types of line breaks for Unix (new Mac), Windows and classical Mac.
We see that unix and mac end in 0000005 and windows in 0000006, in unix(000a) we don't have a carry, on mac(000d) we don't have a newline and on windows(0a0d) we have both.

17. Open the gedit text editor and type "abñ". Go to the menu and use the option "Save As" to save the file with the name text3.txt and "Character Encoding" ISO-8859-15. Recheck the contents of the text file with hexdump and discuss the results.



```
dumre@dumre  ~  main  hexdump text1_ISO.txt
0000000 6261 0af1
0000004
```



ISO-8859-15

	x0	x1	x2	x3	x4	x5	x6	x7	x8	x9	xA	xB	xC	xD	xE	xF
0x	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1x	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2x	SP	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3x	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4x	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5x	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6x	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7x	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL
8x	PAD	HOP	BPH	NBH	IND	NEL	SSA	ESA	HTS	HTJ	VTS	PLD	PLU	RI	SS2	SS3
9x	DCS	PU1	PU2	STS	CCH	MW	SPA	EPA	SOS	SGCI	SCI	CSI	ST	OSC	PM	APC
Ax	NBSP	¡	¢	£	€	¥	Š	§	š	©	ª	«	¬	SHY	®	¯
Bx	°	±	²	³	Ž	µ	¶	·	ž	¹	º	»	Œ	œ	Ÿ	¿
Cx	À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Î	Ï
Dx	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
Ex	à	á	â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	î	ï
Fx	ð	ñ	ò	ó	ô	õ	ö	÷	ø	ù	ú	û	ü	ý	þ	ÿ

In this case we can see also the inverted bits. The "ñ" appears as f1 and 0a is the newline.

4 Chapter 4: File Descriptors

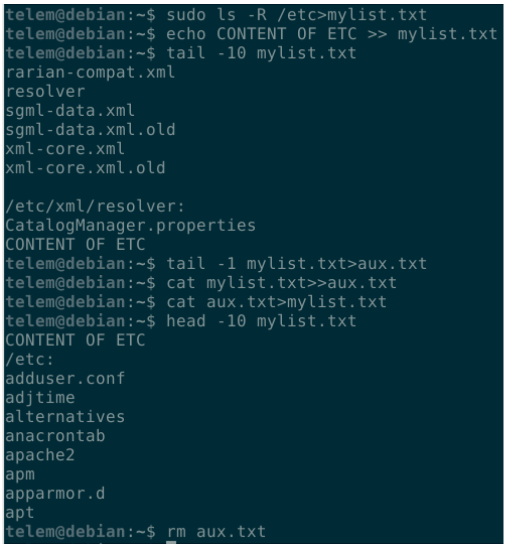
4.1 Exercise 4.1

In this exercise, we will practice with file redirections using several filter commands.

1. Without using any text editor, you have to create a file called mylist.txt in your home directory that contains the recursive list of contents of the /etc directory. Hint: use ls -R. Then, "append" the sentence "CONTENTS OF ETC" at the end of the file mylist.txt. Finally, type a command to view the last 10 lines of mylist.txt to check that you obtained the expected result.

```
touch mylist.txt
sudo ls -R /etc >mylist.txt
#redirect to file mylist the output of command ls -R /etc. sudo
   because permission needed.
echo CONTENTS OF ETC>>mylist.txt
#open file mylist in append mood and add CONTENTS OF ETC
tail -10 mylist.txt
#shows the last 10 lines of the file
```

2. Without using any text editor, you have to "prepend" the sentence "CONTENTS OF ETC" at the beginning of mylist.txt. You can use auxiliary files but when you achieve the desired result, you have to remove them. Finally, check the result typing a command to view the first 10 lines of mylist.txt.



```
telem@debian:~$ sudo ls -R /etc>mylist.txt
telem@debian:~$ echo CONTENT OF ETC >> mylist.txt
telem@debian:~$ tail -10 mylist.txt
rarian-compat.xml
resolver
sgml-data.xml
sgml-data.xml.old
xml-core.xml
xml-core.xml.old

/etc/xml/resolver:
CatalogManager.properties
CONTENT OF ETC
telem@debian:~$ tail -1 mylist.txt>aux.txt
telem@debian:~$ cat mylist.txt>>aux.txt
telem@debian:~$ cat aux.txt>mylist.txt
telem@debian:~$ head -10 mylist.txt
CONTENT OF ETC
/etc:
adduser.conf
adjtime
alternatives
anacrontab
apache2
apm
apparmor.d
apt
telem@debian:~$ rm aux.txt
```

3. Type a command-line using pipes to count the number of files in the /bin directory.

```
ls /bin |wc -w
```

4. Type a command-line using pipes that shows the list of the first 3 commands in the /bin directory. Then, type another command-line to show this list in reverse alphabetical order. Hint: use the commands ls, sort and head.

```
ls /bin | sort | tail -3
ls /bin | sort -r | tail -3
#sort shows ordered list
#short -r is reverse ordered
```

5. Type the command-lines that achieve the same results but using tail instead of head.

```
ls /bin | sort | head -3
ls /bin | sort -r | head -3
```

6. Type a command-line using pipes that shows the "number" of users and groups defined in the system (the sum of both). Hint: use the files /etc/passwd and /etc/group.

```
cat /etc/passwd /etc/group | wc
```

7. Type a command line using pipes that shows one text line containing the PID and the PPID of the init process.

```
ps -A -o comm,pid,ppid | head -2
```

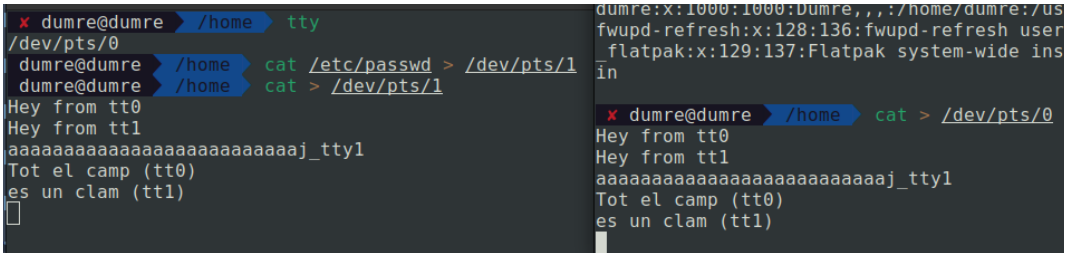
4.2 Exercise 4.2

In this exercise, we are going to practice with the special files of pseudo-terminals (/dev/pts/X).

1. Open two pseudo-terminals. In one pseudo-terminal type a command-line to display the the content of the file /etc/passwd in the other terminal.

```
#First we open 2 terminals, and with the command tty, we obtain the
    the virtul terminal number
tty
#1st terminal -->tty0
#2nd terminal --->tty1
# We want to display content of tty0 in tty1
#from tty0
sudo cat /etc/passwd > /dev/pts/tt1
```

2. You have to build a chat between two pseudo-terminals. That is to say, what you type in one pseudo-terminal must appear in the other pseudo-terminal and vice-versa. Hint: use cat and a redirection to the TTY file of the pseudo-terminal.



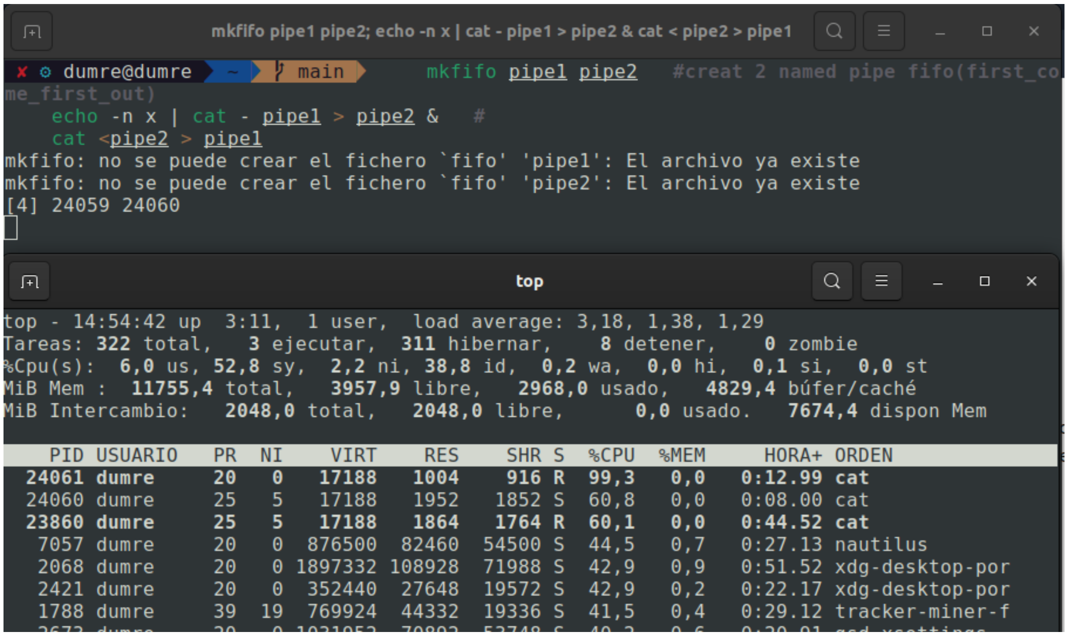
```
✘ dumre@dumre  /home  tty
/dev/pts/0
 dumre@dumre  /home  cat /etc/passwd > /dev/pts/1
 dumre@dumre  /home  cat > /dev/pts/1
Hey from tt0
Hey from tt1
aaaaaaaaaaaaaaaaaaaaaaaaaaj_tty1
Tot el camp (tt0)
es un clam (tt1)
```

```
dumre:x:1000:1000:Dumre,,,:/home/dumre:/us
fwupd-refresh:x:128:136:fwupd-refresh user
_flatpak:x:129:137:Flatpak system-wide ins
in

✘ dumre@dumre  /home  cat > /dev/pts/0
Hey from tt0
Hey from tt1
aaaaaaaaaaaaaaaaaaaaaaaaaaj_tty1
Tot el camp (tt0)
es un clam (tt1)
```

4.3 Exercise 4.3

Explain in detail what happens when you type the following command lines:

```
mkfifo pipe1 pipe2 #creat 2 named pipe fifo(first_come_first_out)
echo -n x | cat - pipe1 > pipe2 & #redirect to pipe2 from pipe1
cat <pipe2 > pipe1 #redirect to pipe1 all input of pipe2
```



```
mkfifo pipe1 pipe2; echo -n x | cat - pipe1 > pipe2 & cat < pipe2 > pipe1

x ⚙ dumre@dumre  ~  main       mkfifo pipe1 pipe2    #creat 2 named pipe fifo(first_co
me_first_out)
    echo -n x | cat - pipe1 > pipe2 &    #
    cat <pipe2 > pipe1
mkfifo: no se puede crear el fichero `fifo' 'pipe1': El archivo ya existe
mkfifo: no se puede crear el fichero `fifo' 'pipe2': El archivo ya existe
[4] 24059 24060
```

```
top

top - 14:54:42 up  3:11,  1 user,  load average: 3,18, 1,38, 1,29
Tareas: 322 total,   3 ejecutar, 311 hibernar,    8 detener,    0 zombie
%Cpu(s):  6,0 us, 52,8 sy,  2,2 ni, 38,8 id,  0,2 wa,  0,0 hi,  0,1 si,  0,0 st
MiB Mem :  11755,4 total,   3957,9 libre,   2968,0 usado,   4829,4 búfer/caché
MiB Intercambio:   2048,0 total,   2048,0 libre,      0,0 usado.   7674,4 dispon Mem

    PID USUARIO   PR  NI    VIRT    RES    SHR S  %CPU  %MEM     HORA+ ORDEN
  24061 dumre     20   0   17188   1004    916 R  99,3   0,0   0:12.99 cat
  24060 dumre     25   5   17188   1952   1852 S  60,8   0,0   0:08.00 cat
  23860 dumre     25   5   17188   1864   1764 R  60,1   0,0   0:44.52 cat
   7057 dumre     20   0  876500  82460  54500 S  44,5   0,7   0:27.13 nautilus
   2068 dumre     20   0 1897332 108928  71988 S  42,9   0,9   0:51.52 xdg-desktop-por
   2421 dumre     20   0  352440  27648  19572 S  42,9   0,2   0:22.17 xdg-desktop-por
   1788 dumre     39  19  769924  44332  19336 S  41,5   0,4   0:29.12 tracker-miner-f
```

Do you see any output? Hint. Use top in another terminal to see CPU usage.

There is a loop of input & output, for that reason the command cat is always in the top.

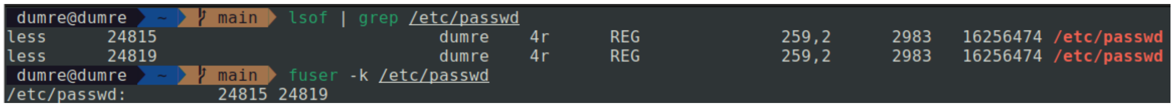
4.4 Exercise 4.4

In this exercise, we deal with inheritance of file descriptors.

1. Execute the command less /etc/passwd in two different pseudo-terminals. Then, from a third terminal list all processes that have opened /etc/ passwd and check their PIDs. Hint: use lsof.

   ```
   lsof | grep /etc/passwd
   ```

2. Using the fuser command, kill all processes that have the file /etc/passwd open.



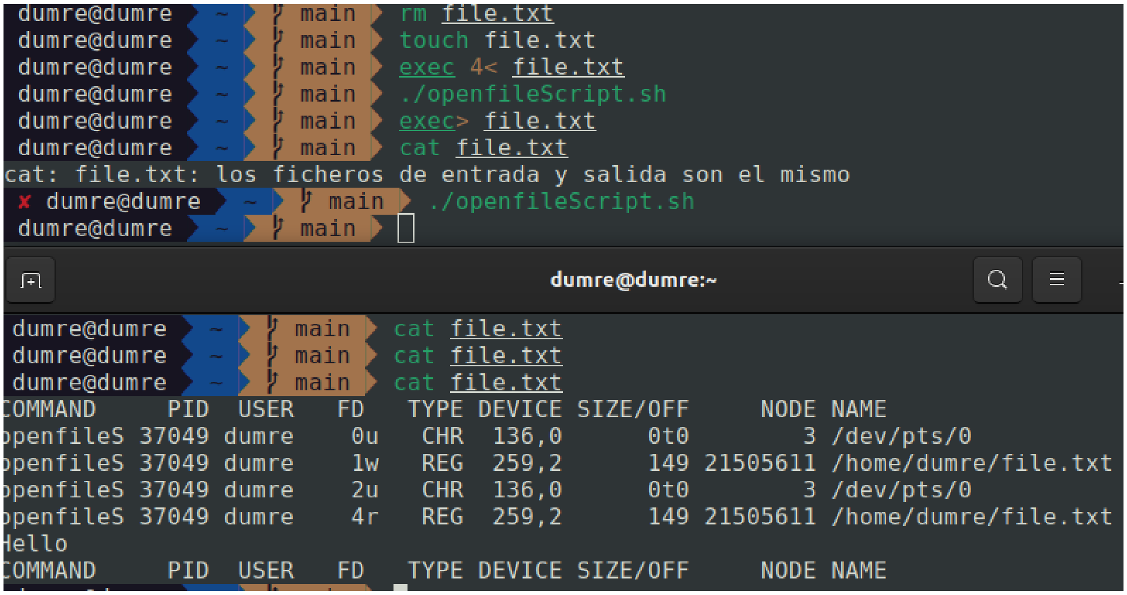
```
 dumre@dumre  ~  main  lsof | grep /etc/passwd
less       24815                              dumre    4r      REG                    259,2       2983   16256474 /etc/passwd
less       24819                              dumre    4r      REG                    259,2       2983   16256474 /etc/passwd
 dumre@dumre  ~  main  fuser -k /etc/passwd
/etc/passwd:             24815 24819
```

3. Open a pseudo-terminal (t1) and create an empty file called file.txt. Open file.txt only for reading with exec using fd=4. Create the following script called “openfilescript.sh”:

   ```
   #!/bin/bash
   # Scriptname: openfilescript.sh
   lsof -a -p $$ -d0-10
   echo "Hello"
   read "TEXT_LINE" <&4
   echo "$TEXT_LINE"
   ```

   Redirect ”stdout“ permanently (with exec) to file.txt in t1 and explain what happens when you execute the previous script in this terminal. Explain what file descriptors has inherited the child bash that executes the commands of the script.

   The aim is put in file.txt content of openfilescript, which shows the list of openfiles. To do that, first we create empty file, then we execute it with reading mode. After we creat bash script where we put the losf command. After that with exec, redirect the output. To see the output we have to do with another terminal, because in the same terminal, file.txt is the stdin & stdout.



```
dumre@dumre  ~  main  rm file.txt
dumre@dumre  ~  main  touch file.txt
dumre@dumre  ~  main  exec 4< file.txt
dumre@dumre  ~  main  ./openfileScript.sh
dumre@dumre  ~  main  exec> file.txt
dumre@dumre  ~  main  cat file.txt
cat: file.txt: los ficheros de entrada y salida son el mismo
 ✘ dumre@dumre  ~  main  ./openfileScript.sh
dumre@dumre  ~  main  
```

```
dumre@dumre  ~  main  cat file.txt
dumre@dumre  ~  main  cat file.txt
dumre@dumre  ~  main  cat file.txt
COMMAND     PID  USER    FD   TYPE DEVICE SIZE/OFF     NODE NAME
openfileS 37049 dumre    0u    CHR  136,0      0t0        3 /dev/pts/0
openfileS 37049 dumre    1w    REG  259,2      149 21505611 /home/dumre/file.txt
openfileS 37049 dumre    2u    CHR  136,0      0t0        3 /dev/pts/0
openfileS 37049 dumre    4r    REG  259,2      149 21505611 /home/dumre/file.txt
Hello
COMMAND     PID  USER    FD   TYPE DEVICE SIZE/OFF     NODE NAME
```

4. From the second pseudo-terminal (t2) remove and create again file.txt. Then, execute "openfilescript.sh" in t1. Explain what happened and why.

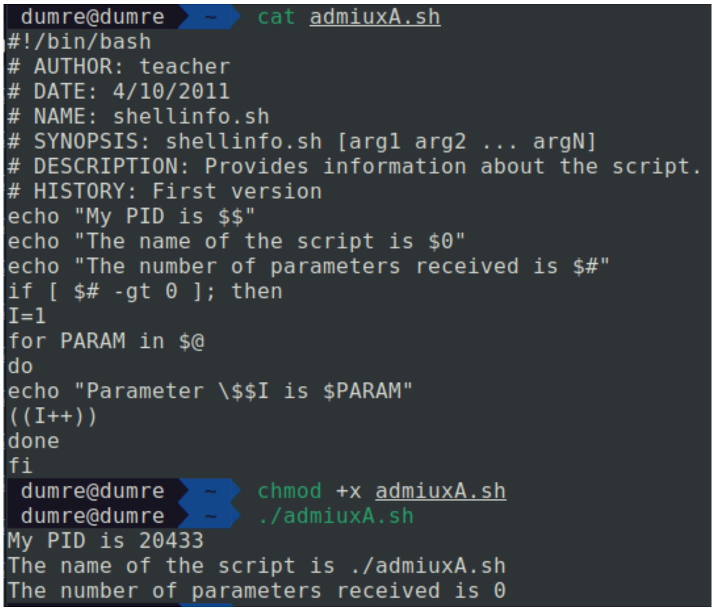
   There is no output because althought the name if same, the file is different.

5 Chapter 11: Shell Scripts

5.1 Exercise 11.1

Describe in detail line by line the following script:

```
#!/bin/bash       #bash script, mandatory to start a .sh file with this
    command.
# AUTHOR: teacher
# DATE: 4/10/2011
# NAME: shellinfo.sh
# SYNOPSIS: shellinfo.sh [arg1 arg2 ... argN]
# DESCRIPTION: Provides information about the script.
# HISTORY: First version
echo "My PID is $$"                   #$$ is PID of the script.
echo "The name of the script is $0"    #$0 is script name.
echo "The number of parameters received is $#" #total number of parameters
    received.
if [ $# -gt 0 ]; then
#If number of parametres is greater than 0, then its returns PID of each
    parameters.

I=1
for PARAM in $@
do
echo "Parameter \$$I is $PARAM"
((I++))
done
fi
```

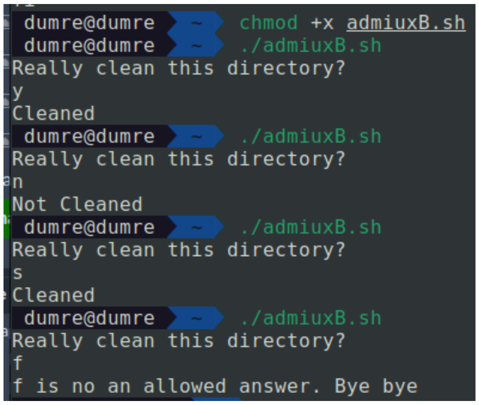


```
dumre@dumre  ~  cat admiuxA.sh
#!/bin/bash
# AUTHOR: teacher
# DATE: 4/10/2011
# NAME: shellinfo.sh
# SYNOPSIS: shellinfo.sh [arg1 arg2 ... argN]
# DESCRIPTION: Provides information about the script.
# HISTORY: First version
echo "My PID is $$"
echo "The name of the script is $0"
echo "The number of parameters received is $#"
if [ $# -gt 0 ]; then
I=1
for PARAM in $@
do
echo "Parameter \$$I is $PARAM"
((I++))
done
fi
dumre@dumre  ~  chmod +x admiuxA.sh
dumre@dumre  ~  ./admiuxA.sh
My PID is 20433
The name of the script is ./admiuxA.sh
The number of parameters received is 0
```

5.2 Exercise 11.2

Describe in detail line by line the following script:

```
#!/bin/bash
# AUTHOR: teacher
# DATE: 4/10/2011
# NAME: clean.sh
# SYNOPSIS: clean.sh (without parameters)
# DESCRIPTION: Removes temporal files in your working directory:
# HISTORY: First version
echo "Really clean this directory?" #Print the string
read YORN                          #Save the answer/input parameters as YORN
case $YORN in                      #treate YORN depending the input
y|Y|s|S) ACTION=0;;                #if YORN is yes/si then action is 0.
n|N) ACTION=1;;                    #if YORN is No then action is 1.
*) ACTION=2;;                      #if YORN is other value then action is 2.
esac
if [ $ACTION -eq 0 ]; then        #The action 0 removes the files and print
    Cleaned.
rm -f \#* *~ .*~ *.bak .*.bak *.backup *.tmp .*.tmp core a.out
echo "Cleaned"
exit 0
elif [ $ACTION -eq 1 ]; then     #The action 1 doesn't removes the files and
    print Not Cleaned.
echo "Not Cleaned"
exit 0
elif [ $ACTION -eq 2 ]; then     #The action 2 print the input YORN isn't
    allowd answer.Bye bye.
echo "$YORN is no an allowed answer. Bye bye"
exit 1
else
echo "Uaggg!! Symptomatic Error"
exit 2
fi
```

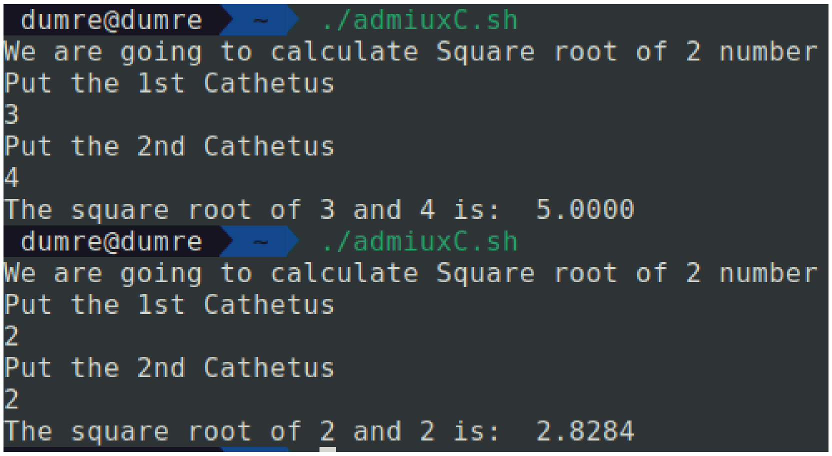


```
 dumre@dumre  ~  chmod +x admiuxB.sh
 dumre@dumre  ~  ./admiuxB.sh
Really clean this directory?
y
Cleaned
 dumre@dumre  ~  ./admiuxB.sh
Really clean this directory?
n
Not Cleaned
 dumre@dumre  ~  ./admiuxB.sh
Really clean this directory?
s
Cleaned
 dumre@dumre  ~  ./admiuxB.sh
Really clean this directory?
f
f is no an allowed answer. Bye bye
```

5.3 Exercise 11.3

Develop a script that calculates the square root of two cathetus. Use a function with local variables and arithmetic expansions.

```
#!/bin/bash
# AUTHOR: Jaume&Gokarna
# DATE:‘date‘
#NAME:SquareRoot.sh
echo "We are going to calculate Square root of 2 number"
echo "Put the 1st Cathetus"
read x
echo "Put the 2nd Cathetus"
read y

#es_numero=’ˆ-?[0-9]+([.][0-9]+)?$’
#if ! [[ $x =˜ $es_numero ]] ; then
#   echo "ERROR:Put number pls " >&amp;2; exit 1
#fi
#elif ! [[ $y =˜ $es_numero ]] ; then
#   echo "ERROR:Put number pls " >&amp;2; exit 1
#fi
#This segment of code was for the control of inputs those weren’t numbers.

((sum = x*x+y*y))
sqr=‘echo "scale=4; sqrt($sum)" | bc‘
echo "The square root of $x and $y is: $sqr"
```



```
 dumre@dumre  ~  ./admiuxC.sh
We are going to calculate Square root of 2 number
Put the 1st Cathetus
3
Put the 2nd Cathetus
4
The square root of 3 and 4 is:  5.0000
 dumre@dumre  ~  ./admiuxC.sh
We are going to calculate Square root of 2 number
Put the 1st Cathetus
2
Put the 2nd Cathetus
2
The square root of 2 and 2 is:  2.8284
```

5.4 Exercise 11.4

Describe in detail line by line the following script:

```
#!/bin/bash
# AUTHOR: teacher
# DATE: 4/10/2011
# NAME: fill_terminal_procedure.sh
# SYNOPSIS: fill_terminal arg
# DESCRIPTION: Procedure to fill the terminal with a printable character
# FUNCTION NAME: fill_terminal:
# OUTPUT: none         #output variable
# RETURN CODES: 0-success 1-bad-number-of-args 2-not-a-printable-character.
    #output codes
# HISTORY: First version
fill_terminal() {
[ $# -ne 1 ] && return 1
local HEXCHAR DECCHAR i j  #local variables
HEXCHAR=$1                  #set first parameters as a local variable
    Hexchar.
DECCHAR= printf  "%d" 0 x$HEXCHAR  #shows the decimal of HEXCHAR variables.
if [ $DECCHAR -lt 33 -o $DECCHAR -gt 127 ]; then
return 2
#if DECCHAR value is in out of range [33-127] shows output code 2.(non
    printable char.)
fi
[ -z "$COLUMNS" ] && COLUMNS=80 #if the variable COLUMNS is empty then the
    value is 80.
[ -z "$LINES" ] && LINES=24    #if the variable COLUMNS is empty then the
    value is 24.
((LINES-=2))
for((i=0; i< COLUMNS; i++))     #The aim of this for, is to print HEXCHAR
    is all COLUMNS and LINES.
do
for ((j=0; j< LINES; j++))
do
printf "\x$HEXCHAR"
done
done
return 0
}


#!/bin/bash
# AUTHOR: teacher
# DATE: 4/10/2011
# NAME: procedure.sh
# SYNOPSIS: procedure.sh arg
# DESCRIPTION: Use the fill_terminal procedure
# HISTORY: First version
source fill_terminal_procedure.sh      #execute the
   fill_terminal_procedure.sh file
fill_terminal $@                       #generate string with recieved
   parameters.
case $? in
0)
exit 0 ;;                       #in case of 0 no parameters is nedded & the
   process is killed.
1)
echo "I need one argument (an hex value)" >&2 ; exit 1 ;; #in case of 1,
   parameters are nedded, at least one.
2)
echo "Not printable character. Try one between 0x21 and 0x7F" >&2 ; exit 1
   ;; #in case of 2, no char is printed.
*)
echo "Internal error" >&2 ; exit 1 #otherwise error.
esac
```

5.5 Exercise 11.5

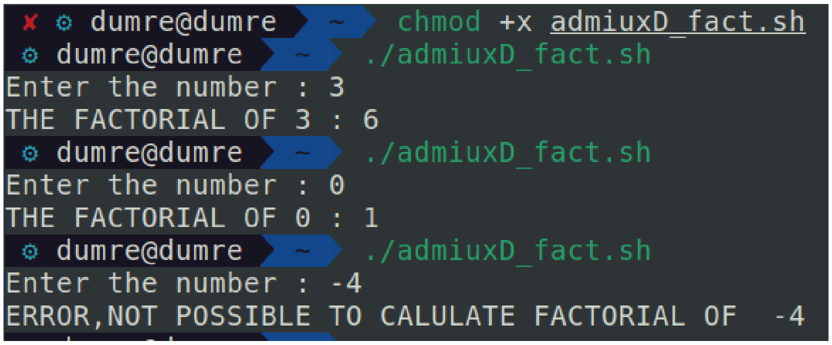
The following script illustrates how to use functions recursively. Describe it in detail line by line.

```
#!/bin/bash
# AUTHOR: teacher
# DATE: 4/10/2011
# NAME: recfind.sh
# SYNOPSIS: recfind.sh file_to_be_found
# DESCRIPTION: Search recursively a file from the working directory
# HISTORY: First version
# Function: search_in_dir
# Arguments: search directory #parameter used in the function
function search_in_dir() {
local fileitem       #local variable
[ $DEBUG -eq 1 ] && echo "Entrant a $1" #if DEBUG is 1, its shows the string
   indicating the entry in the directory.
cd $1                               #entry to the directory
for fileitem in *                   #iteration of fileitem in all actual
   files and directories.
do
if [ -d $fileitem ]; then #if fileitem is directory.
search_in_dir $fileitem
elif [ "$fileitem" = "$FILE_IN_SEARCH" ]; then # if fileitem is a file which
   we are looking for.
echo   pwd  /$fileitem   #shows the route where the fileitem is located.
fi
done
[ $DEBUG -eq 1 ] && echo "Sortint de $1"
cd ..
}
DEBUG=0 #0 because not needed more.
if [ $# -ne 1 ]; then #if the parameters number is different than 1
echo "Usage: $0 file_to_search" #shows the variable
exit 1
fi
FILE_IN_SEARCH=$1 #we assigned the parameters to the FILE_IN_SEARCH
search_in_dir  pwd   #the function search_in_dir get the route of actual
    directory as a parameter
```

5.6 Exercise 11.6

Using a function recursively, develop a script to calculate the factorial of a number.

```
#!/bin/bash
fact(){
    i=$1       #save the input
    if [ $i -eq 0 -o $i -eq 1 ] #if input is 0 or 1, the result is 1.
    then
        echo 1
    else
        f=‘expr $i \- 1‘ #in other case, fact=n*(n-1).....*1
        f=$(fact $f)
        f=‘expr $i \* $f‘
        echo $f
    fi
}
read -p "Enter the number : " n
if [ $n -lt 0 ] #if input is lower than 0 return error
then
  echo "ERROR,NOT POSSIBLE FACTORIAL CALULATTION OF $n "
else
  echo "THE FACTORIAL OF $n : $(fact $n) "
fi
```

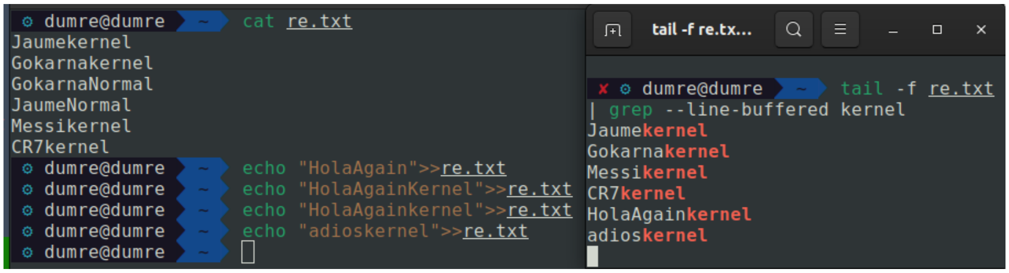


```
✘ ⚙ dumre@dumre  ~  chmod +x admiuxD_fact.sh
⚙ dumre@dumre  ~  ./admiuxD_fact.sh
Enter the number : 3
THE FACTORIAL OF 3 : 6
⚙ dumre@dumre  ~  ./admiuxD_fact.sh
Enter the number : 0
THE FACTORIAL OF 0 : 1
⚙ dumre@dumre  ~  ./admiuxD_fact.sh
Enter the number : -4
ERROR,NOT POSSIBLE TO CALULATE FACTORIAL OF  -4
```

5.7 Exercise 11.7

In this exercise, we will practice with Regular Expressions (regex).

1. Create a file called re.txt and type a command-line that continuously ”follows“ this file to display text lines added to this file in the following way: Display only the text lines containing the word ”kernel“. From these lines, display only the first 10 characters. Try your command-line by writing from another pseudo-terminal some text lines to the file re.txt. Hint: use tail, grep and cut. Note. You must use the grep command with the option –line-buffered. This option prevents grep from using its internal buffer for text lines. If you do not use this option you will not see anything displayed on the terminal.

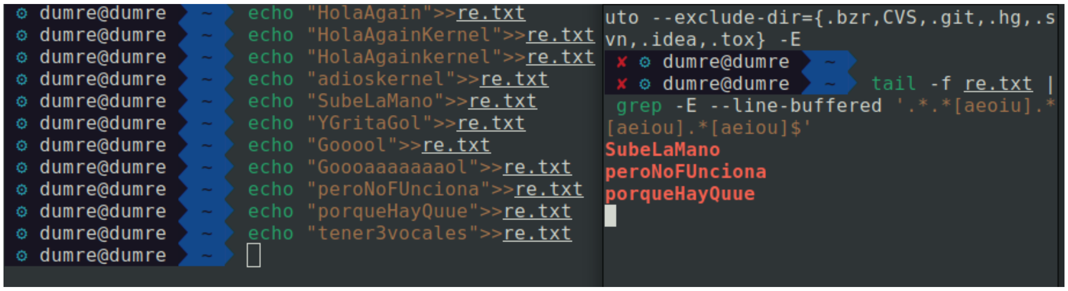


```
⚙ dumre@dumre  ~  cat re.txt
Jaumekernel
Gokarnakernel
GokarnaNormal
JaumeNormal
Messikernel
CR7kernel
⚙ dumre@dumre  ~  echo "HolaAgain">>re.txt
⚙ dumre@dumre  ~  echo "HolaAgainKernel">>re.txt
⚙ dumre@dumre  ~  echo "HolaAgainkernel">>re.txt
⚙ dumre@dumre  ~  echo "adioskernel">>re.txt
⚙ dumre@dumre  ~
```

```
✘ ⚙ dumre@dumre  ~  tail -f re.txt
| grep --line-buffered kernel
Jaumekernel
Gokarnakernel
Messikernel
CR7kernel
HolaAgainkernel
adioskernel
```

2. Type a command-line that continuously ”follows“ the re.txt file to display the new text lines in this file in the following way: Display only the text lines ending with a word containing three vowels. Try your command-line by sending from another pseudo-terminal some text lines to re.txt.

```
tail -f re.txt | grep -E --line-buffered '.* .*[aeoiu].*[aeiou].*[aeiou]$'
```



```
⚙ dumre@dumre  ~  echo "HolaAgain">>re.txt
⚙ dumre@dumre  ~  echo "HolaAgainKernel">>re.txt
⚙ dumre@dumre  ~  echo "HolaAgainkernel">>re.txt
⚙ dumre@dumre  ~  echo "adioskernel">>re.txt
⚙ dumre@dumre  ~  echo "SubeLaMano">>re.txt
⚙ dumre@dumre  ~  echo "YGritaGol">>re.txt
⚙ dumre@dumre  ~  echo "Gooool">>re.txt
⚙ dumre@dumre  ~  echo "Goooaaaaaaaol">>re.txt
⚙ dumre@dumre  ~  echo "peroNoFUnciona">>re.txt
⚙ dumre@dumre  ~  echo "porqueHayQuue">>re.txt
⚙ dumre@dumre  ~  echo "tener3vocales">>re.txt
⚙ dumre@dumre  ~
```

```
uto --exclude-dir={.bzr,CVS,.git,.hg,.s
vn,.idea,.tox} -E
 ✗ ⚙ dumre@dumre  ~
 ✗ ⚙ dumre@dumre  ~  tail -f re.txt |
 grep -E --line-buffered '.*.*[aeoiu].*
[aeiou].*[aeiou]$'
SubeLaMano
peroNoFUnciona
porqueHayQuue
```