# ADMINUX

Òscar Pérez Castillo

January 30, 2021

# Contents

# 1 ADMINUX

## 1.1 Process

### 1.1.1 Exercise 2.1

-1. By searching the pattern `/ppid` and then `n/N` for moving upwards/backwards, we find **7** ocurrences

-2. Command:

```
opc :: ~ » ps -o cmd,tty,pid
CMD                         TT           PID
/usr/bin/zsh                pts/3      45474
ps -o cmd,tty,pid           pts/3      45521
```

vs

```
opc :: ~ » ps -o cmd,tty,pid
CMD                         TT           PID
/usr/bin/zsh                tty2        45476
ps -o cmd,tty,pid           tty2        45556
```

- The TT column is different (terminal emulator vs virtual console)

-3. Command

```
opc :: ~ » ps -o pid,comm
    PID COMMAND
  45474 zsh
  45926 ps
opc :: ~ » ps -o pid,cmd
    PID CMD
  45474 /usr/bin/zsh
  45936 ps -o pid,cmd
opc :: ~ »
```

- Cmd: shows the name of the executable, including also the argument passed. If the command is not a bash built-in, the command show the absolute path of the executable. See the diference using bash instead of zsh:

```
[op@opc ~]$ ps -o pid,cmd
    PID CMD
  45474 /usr/bin/zsh
  46074 bash
  46113 ps -o pid,cmd
[op@opc ~]$
```

- Comm: shows the name of the command

-4. Command:

```
systemd
|
.
.
.
|--kdeinit5---file.so
|          |--kaccess---2*[{kaccess}]
|          |--kded5---konsole---zsh---tmux: client
|          |       |         '--7*[{konsole}]
|          |       |--konsole---zsh
|          |       |         '--6*[{konsole}]
|          |       |--konsole---zsh---pstree
|          |       |         '--6*[{konsole}]
|          |       '--6*[{kded5}]
```

We have the following structure:

- pstree process
- zsh: parent process acting as the interpreter who executes de pstree command
- konsole: terminal emulator used
- kded5: consolidates several small services in one process.
- kdeinit5: kdeinit5 is a process launcher somewhat similar to the famous init used for booting UNIX. It executes KDE programs and kdeinit loadable modules (KLMs) starting them more efficiently.

And by looking a the whole structure more precisely:

```
kdeinit5(1436)---file.so(2095)
               |--kaccess(1447)---{kaccess}(1448)
               |               '--{kaccess}(1460)
               |--kded5(1440)---konsole(26378)---zsh(26387)---tmux: client(29455)
               |             |--konsole(45467)---zsh(45474)---pstree(48850)
               |             |                |--{konsole}(45468)
               |             |                |--{konsole}(45469)
               |             |                |--{konsole}(45470)
               |             |                |--{konsole}(45471)
               |             |                |--{konsole}(45472)
               |             |                '--{konsole}(45473)
```

we can see the whole hierarchy with the pid associated to them with the threads associated ({})

-5. By looking, in my case, through a tmux instance, we see the following structure:

```
opc :: /tmp » pstree -c -g -p 29457
tmux: server(29457,29457)---zsh(29458,29458)
                          |--zsh(29500,29500)---evince(35564,35564)---{evince}(35568,35564)
                          |                                        |--{evince}(35569,35564)
                          |                                        |--{evince}(35571,35564)
                          |                                        '--{evince}(35588,35564)
                          |--zsh(37609,37609)---vim(37739,37739)
                          |--zsh(37756,37756)---vim(37994,37994)
                          '--zsh(38636,38636)
opc :: /tmp »
```

where:

- Each tab is represented as a new child process, where all of them have a common parent -tmux: server

- We can see the same structure in each ”tab-window”. An instance of zsh (alternative of bash) that executes the corresponding application (evince, vim)

-6. By opening an xterm console, we see the following:

```
|-plasmashell(1498,1496)
|                        |-xterm(54003,1496)---zsh(54006,54006) ...
```

- The parent process is not the same as we are not using a native kde application, which were controlled by a kde5 process.

-7. We have the following:

- t1: running in foreground. The terminal is ”blocked” and we can not execute another program
- t2: running in the background. We can execute other programs in the terminal and we have the pid associated with the xclock program (pid: 54875)

-8. Where the possible states are the following:

```
PROCESS STATE CODES
       Here are the different values that the s, stat and state output specifiers (header "
           STAT" or "S") will display to describe the state of
       a process:

               D    uninterruptible sleep (usually IO)
               I    Idle kernel thread
               R    running or runnable (on run queue)
               S    interruptible sleep (waiting for an event to complete)
               T    stopped by job control signal
               t    stopped by debugger during the tracing
               W    paging (not valid since the 2.6.xx kernel)
               X    dead (should never be seen)
               Z    defunct ("zombie") process, terminated but not reaped by its parent
```

- t1: by executing

```
opc :: ~ » ps -C xeyes -o pid,state,tty,ppid
    PID S TT          PPID
  54854 S pts/10     37756
opc :: ~ »
```

where the states a

- t2:

```
opc :: ~ » ps -C xclock -o pid,state,tty,ppid
    PID S TT          PPID
  54797 S pts/3      45474
opc :: ~ »
```

We can see that:

- We have differents tty, as we are using differents ”konsole” windows
- Each one has a different PPID, as they are created from differents zsh instances

-9.

```
opc :: ~ » kill -9 37756
```

-10. First we tried to do it with zsh, but zsh does not enable orphan process, as when you type exit(terminating the zsh processs) zsh terminates all the background process iniciated by zsh. So we then do it with bash:

- PPID: 57901

Then:

```
opc :: ~ » ps -C xclock -o pid,state,tty,ppid
    PID S TT          PPID
  57936 S pts/2          1
opc :: ~ »
```

- As we see it, now the process has the init process (PID=1) as a parent

-11.

```
opc :: ~ » kill -9 57936
```

-12,13,14.

- Using signals:

```
opc :: ~ » ps -C xclock -o pid,state,tty,ppid
    PID S TT          PPID
  89002 S pts/6      88397
opc :: ~ » kill -SIGSTOP 89002
opc :: ~ » ps -C xclock -o pid,state,tty,ppid
    PID S TT          PPID
  89002 T pts/6      88397
opc :: ~ » kill -SIGCONT 89002
opc :: ~ » ps -C xclock -o pid,state,tty,ppid
    PID S TT          PPID
  89002 S pts/6      88397
opc :: ~ » kill -SIGKILL 89002
```

- Using job control

```
opc :: ~ » xclock
Warning: Missing charsets in String to FontSet conversion
^Z
[1]  + 90314 suspended  xclock
opc :: ~ » jobs

    148 ⊠
[1]  + suspended  xclock
opc :: ~ » kill -SIGCONT %1
opc :: ~ » jobs
[1]  + running    xclock
opc :: ~ » kill -SIGKILL %1
[1]  + 90314 killed     xclock
opc :: ~ »
```

-15.

```
opc :: ~ » jobs
[1]    running    xclock
[2]    running    xclock
[3]  - running    xeyes
[4]  + running    xeyes
opc :: ~ » fg %1
^Z
opc :: ~ » bg %1
opc :: ~ » killall xclock
opc :: ~ » killall xeyes
```

-16.

```
opc :: ~ » ps && sleep 3 && ps
    PID TTY          TIME CMD
  87876 pts/2    00:00:06 zsh
  91167 pts/2    00:00:00 ps
    PID TTY          TIME CMD
  87876 pts/2    00:00:06 zsh
  91170 pts/2    00:00:00 ps
opc :: ~ »
```

- We can see that the PID of the ps instance is different after the sleep(3), as we are creating different processes

-17.

```
opc :: ~ » ps -nothing || ps
error: unsupported SysV option

Usage:
 ps [options]

 Try 'ps --help <simple|list|output|threads|misc|all>'
  or 'ps --help <s|l|o|t|m|a>'
 for additional help text.

For more details see ps(1).
    PID TTY          TIME CMD
  87876 pts/2    00:00:06 zsh
  91301 pts/2    00:00:00 ps
opc :: ~ »
```

- The last command (ps) will only execute if the first one has and error exit code, which is the case

-18.

- 1:

```
opc :: ~ » sleep || sleep || ls
sleep: falta un operando
Pruebe 'sleep --help' para más información.
sleep: falta un operando
Pruebe 'sleep --help' para más información.
bin  Descargas  Documentos  Escritorio  Imágenes  kwin-tiling  m  Python  Universidad  VM
opc :: ~ »
```

- As "||" will only execute if the first one has an error, the first one sleep have errors, so the ls commands is executed

- 2:

```
opc :: ~ » sleep && sleep --help || ls && ps
sleep: falta un operando
Pruebe 'sleep --help' para más información.
bin  Descargas  Documentos  Escritorio  Imágenes  kwin-tiling  m  Python  Universidad  VM
    PID TTY          TIME CMD
  87876 pts/2    00:00:07 zsh
  91700 pts/2    00:00:00 ps
opc :: ~ »
```

- The first sleep has an error, so the second one will not execute
- As the last executed was exit successfully, the ls command will execute
- The last ps will execute, as the previous command did not have any error
- 3:

```
pc :: ~ » sleep && sleep --help || ls || ps
sleep: falta un operando
Pruebe 'sleep --help' para más información.
bin  Descargas  Documentos  Escritorio  Imágenes  kwin-tiling  m  Python  Universidad  VM
opc :: ~ »
```

- The same as before, but the last ps will not execute as the ”ls” command did run successfully

### 1.1.2 Exercise 2.2

-1.

- script.sh:

```
#!/bin/sh
# script.sh

read NUMBER
echo "Number: $NUMBER"
echo "Result: $[NUMBER * 7]"
```

- result:

```
opc :: /tmp » ./script.sh
8
Number: 8
Result: 56
opc :: /tmp »
```

-2.

- script2.sh

```
#!/bin/sh
# script2.sh

# Trap function
trap "echo signal captured" USR1

# Main function
while true
do
        sleep 1
done
```

- result:

```
opc :: /tmp » kill -USR1 93268
opc :: /tmp » signal captured
```

## 1.2 File system

### 1.2.1 Exercise 3.1

-1,2,3,4,5.

```
# Moving directories
opc :: /tmp/ADMINUX » cd ~
opc :: ~ » cd ../../etc
opc :: /etc »
opc :: /etc » cd /home/op
opc :: ~ »
# Copying files
opc :: ~ » cp ../../etc/passwd .
# Deleting directories
opc :: /tmp » rm -rf dir{A,B}{1,2}
opc :: /tmp » rm -rf dir{A,B}*
opc :: /tmp » rm -rf dirC?
```

-6,7,8,9.

```
# Creation files
opc :: /tmp » touch temp.txt
opc :: /tmp » cat temp.txt
opc :: /tmp » stat temp.txt

    Fichero: temp.txt
  Tamaño: 0             Bloques: 0          Bloque E/S: 4096   fichero regular vacío
Dispositivo: 24h/36d    Nodo-i: 1392        Enlaces: 1
Acceso: (0664/-rw-rw-r--)  Uid: ( 1000/      op)   Gid: ( 1000/      op)
Contexto: unconfined_u:object_r:user_tmp_t:s0
      Acceso: 2021-01-26 16:43:56.936980808 +0100
Modificación: 2021-01-26 16:43:54.511964491 +0100
      Cambio: 2021-01-26 16:43:54.511964491 +0100
    Creación: -
opc :: /tmp »
```

-10.

```
opc :: /tmp » cp temp.txt /usr
cp: no se puede crear el fichero regular '/usr/temp.txt': Permiso denegado
opc :: /tmp »
```

- We can not write anything to the /usr directory
- As everything is a ”file”, we can look the permisiosn related to the /usr directory:

```
drwxr-xr-x.  13 root root  4096 nov  6 21:54 usr
```

- We can see that only the root user and root groupd have the basics permisions. The other only have the ”x” permision i.e. execution of programs

-11,12.

```
opc :: /tmp/practices » mkdir permission
opc :: /tmp/practices » mkdir no_permission
opc :: /tmp/practices » chmod u-w no_permission
opc :: /tmp/practices »
opc :: /tmp/practices » ls -l
total 0
dr-xrwxr-x. 2 op op 40 ene 26 16:53 no_permission
drwxrwxr-x. 2 op op 40 ene 26 16:53 permission
opc :: /tmp/practices »
```

- We don't have write permission to the no_permission directory, so we can not create any files inside it

```
opc :: /tmp/practices » touch no_permission/temp.txt
touch: no se puede efectuar `touch' sobre 'no_permission/temp.txt': Permiso denegado
opc :: /tmp/practices » touch permission/tempt.txt
opc :: /tmp/practices »
```

-13.

```
Commands                                read | write | execute
-------                                 ----------------------
cd no_permission;                       ok     no      no
cd no_permission; ls -l                 ok     no      no
cd temp ~/practices/no_permission       ok     ok      no
```

### 1.2.2 Exercise 3.2

-1,2,3.

```
opc :: /tmp » touch temp.txt
opc :: /tmp » touch origin.txt
opc :: /tmp » ln -s origin.txt link.txt
```

- Does not matter which file we modify.

-4.

```
opc :: /tmp » chmod u-rwx origin.txt
opc :: /tmp » cat origin.txt
cat: origin.txt: Permiso denegado
opc :: /tmp » cat link.txt

    1 ⊠
cat: link.txt: Permiso denegado
opc :: /tmp »
```

- We can not do anything from the symbolic link neither

-5.

```
opc :: /tmp/practices » chmod u-w link.txt
opc :: /tmp/practices » ls -l
total 4
lrwxrwxrwx. 1 op op 10 ene 26 17:17 link.txt -> origin.txt
d---rwxr-x. 2 op op 40 ene 26 16:53 no_permission
-r--rw-r--. 1 op op  6 ene 26 17:19 origin.txt
drwxrwxr-x. 2 op op 60 ene 26 16:55 permission
opc :: /tmp/practices »
```

- Removing the write permision into the symbolic link **implies removing access to the main file**

-6.

- In this case, the ”hello” is then transfered to the origin.txt. While the origin.txt does not exist, ”link.txt” has a broken link, but it ”has some content inside” that is ”transfered” once we create the archive by opening vi.

-7,8,9.

```
opc :: /tmp/practices » stat origin.txt
  Fichero: origin.txt
  Tamaño: 6             Bloques: 8          Bloque E/S: 4096   fichero regular
Dispositivo: 24h/36d    Nodo-i: 1449        Enlaces: 2
Acceso: (0464/-r--rw-r--)  Uid: ( 1000/      op)   Gid: ( 1000/      op)
Contexto: unconfined_u:object_r:user_tmp_t:s0
      Acceso: 2021-01-26 17:20:23.373068406 +0100
Modificación: 2021-01-26 17:19:59.209904892 +0100
      Cambio: 2021-01-26 17:41:29.568391461 +0100
    Creación: -
opc :: /tmp/practices »
```

- With the hard lin way, the stat command show us the ”links”

Then when we remove the origin.txt:

- Now the other one, as we have a hard link (ie. the same file with different links), the file can be modified as if nothign has happened.

So in conclusion:

- Hard link: creation of the same file
- Soft link: link to a file that can o can not exist

-10.

```
opc :: /tmp » cat /etc/services| grep "HTTP" | wc -l
104
opc :: /tmp »
```

-11.

```
opc :: /tmp » cat /etc/group | cut -d ":" -f 1,4
```

- Where not all the groups have some user in them

-12.

```
opc :: /tmp » touch tex1.txt
opc :: /tmp » echo "abñ" > tex1.txt
opc :: /tmp » file tex1.txt
tex1.txt: UTF-8 Unicode text
opc :: /tmp »
```

-13. The encoding of ”ñ” is:

- ISO-8859-15: 0x00F1
- UTF-8: 0xC3 B1

So by looking at the hexadecimals:

- ISO-8859-15: we see ”b1c3” due to the endianess

```
opc :: /tmp » hexdump tex1.txt
0000000 6261 b1c3 000a
0000005
opc :: /tmp »
```

-14,15,16.

- 0x0a (UTF-8): correspons to the line feed (EOL)
- 0x0d (UTF-8): correspons to the carriage return (cr)

## 1.3 File descriptors

### 1.3.1 Exercise 4.1

-1.

```
opc :: /tmp/ADMINUX » touch mylist.txt
opc :: /tmp/ADMINUX » ls
mylist.txt
opc :: /tmp/ADMINUX » ls -R /etc > mylist.txt && echo "CONTENTS OF ETC" >> mylist.txt   opc ::
     /tmp/ADMINUX » tail --lines=10 < mylist.txt
rpmfusion-free-updates-testing.repo
rpmfusion-nonfree.repo
rpmfusion-nonfree-updates.repo
rpmfusion-nonfree-updates-testing.repo
tailscale.repo
vscodium.repo

/etc/zfs-fuse:
zfs_pool_alert
CONTENTS OF ETC
opc :: /tmp/ADMINUX »
```

-2. By concateneting the following commands:

```
opc :: /tmp/ADMINUX » sudo echo "CONTENT OF ETC" > mylist.txt && sudo ls -R /etc >> mylist.txt
     && head -n 10 < mylist.txt
CONTENT OF ETC
/etc:
abrt
adjtime
aliases
alsa
alternatives
anaconda
anacrontab
appstream.conf
opc :: /tmp/ADMINUX »
```

-3.

```
opc :: /tmp/ADMINUX » ls /bin | wc --lines
2668
opc :: /tmp/ADMINUX »
```

-4.

```
opc :: /tmp/ADMINUX » ls /bin | head -n 3 | sort --reverse
7za
7z
[
opc :: /tmp/ADMINUX »
```

-5.

```
opc :: /tmp/ADMINUX » ls /bin | sort --reverse | tail -n 3
7za
7z
[
opc :: /tmp/ADMINUX »
```

-6.

```
opc :: /tmp/ADMINUX » cat /etc/{group,passwd} | wc --lines
123
opc :: /tmp/ADMINUX »
```

-7.

```
opc :: /tmp/ADMINUX » ps -A -o pid,ppid,tty,time,cmd | head -n 2
    PID    PPID TT           TIME CMD
      1       0 ?        00:00:02 /usr/lib/systemd/systemd --switched-root --system --
          deserialize 30
```

- Where the process with PID seems to be the scheluder of the scheluder

### 1.3.2 Exercise 4.2

```
opc :: /tmp/ADMINUX » cat /etc/passwd > /dev/pts/4
```

and then execute in each terminal the redirection to the other /dev/pts/..

### 1.3.3 Exercise 4.3

```
# We create to FIFO pipes
$: mkfifo pipe1 pipe2

# We concatenate cat to the input of the pipe (i.e: x) +
# the output of the FIFO pipe
# And all of it's output is redirected to the FIFO pipe2
# and we set it to the background
# Problems:
# - pipe1: will wait for someone to insert data
# - pipe2: will wait for someone to read data
$: echo -n x | cat - pipe1 > pipe2 &

# We connect:
# input: pipe2
# output: pipe1
# Problems:
# pipe2: waiting for someone to put data
# pipe1: waiting for someone to reading data
$: cat <pipe2 > pipe1
```

### 1.3.4 Exercise 4.4

-1.

```
opc :: /tmp/ADMINUX » lsof /etc/passwd
COMMAND   PID USER   FD   TYPE DEVICE SIZE/OFF    NODE NAME
less    19118   op    4r   REG  253,0     2628 2099117 /etc/passwd
less    19124   op    4r   REG  253,0     2628 2099117 /etc/passwd
opc :: /tmp/ADMINUX » ps -p 19118
    PID TTY          TIME CMD
  19118 pts/2    00:00:00 less
opc :: /tmp/ADMINUX » ps -p 19124
    PID TTY          TIME CMD
  19124 pts/4    00:00:00 less
opc :: /tmp/ADMINUX »
```

-2.

```
opc :: /tmp/ADMINUX » fuser /etc/passwd
/etc/passwd:         19118 19124
opc :: /tmp/ADMINUX » kill -9 {19118,19124}
opc :: /tmp/ADMINUX »
```

## 1.4 Scripts

### 1.4.1 Exercise 11.1

```bash
#!/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

# Print PID
echo "My PID is $$"
# Print name of file $0=shellinfo
echo "The name of the script is $0"
# Print the number of parameters passed
# after the shellinfo.sh ...
echo "The number of parameters received is $#"

# If number of parameters is greater than 0
if [ $# -gt 0 ]; then
        # Declare variable I
        I=1
        # For over the element passed
        for PARAM in $@
        do
                # We echo:
                # "\$" = $
                # "$I" = I (value)
                echo "Parameter \$$I is $PARAM"

                # We do arithmetic operation
                # so we need the double "(())""
                # And we simply sum + 1
                ((I++))
        done
fi
```

Output:

```
opc :: 4A#ADMINUX#scripts ‹main›* » .#shellinfo.sh 1 2 3 4 5
My PID is 69933
The name of the script is .#shellinfo.sh
The number of parameters received is 5
Parameter $1 is 1
Parameter $2 is 2
Parameter $3 is 3
Parameter $4 is 4
Parameter $5 is 5
opc :: 4A#ADMINUX#scripts ‹main›* »
```

### 1.4.2 Exercise 11.2

```
#!/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?"
# Read and assign it to var YORN
read YORN
# Case statement of var YORN
case $YORN in
        # In case Yorn equal y#Y#s#S
        y|Y|s|S) ACTION=0;;


        # In case Yorn equal n#N
        n|N) ACTION=1;;

        # Otherwise
        *) ACTION=2;;
esac

if [ $ACTION -eq 0 ]; then
        # Remove all files that match the patterns
        # It also looks for invisible files
        rm -f \#* *~ .*~ *.bak .*.bak *.backup *.tmp .*.tmp core a.out
        echo "Cleaned"
        exit 0
elif [ $ACTION -eq 1 ]; then
        echo "Not Cleaned"
        exit 0
elif [ $ACTION -eq 2 ]; then
        echo "$YORN is no an allowed answer. Bye bye"
        exit 1
else
        echo "Uaggg!! Symptomatic Error"
        exit 2
fi
```

### 1.4.3 Exercise 11.3

```
opc :: 4A#ADMINUX#scripts ‹main›* » cat root.sh
#!/bin/bash
# AUTHOR: opc
# DATE: 28#1#2021
# NAME: squareroot.sh
# SYNOPSIS: root.sh number
# DESCRIPTION: Computes root square of number given
# HISTORY: First version

square() {
  local CAT1 CAT2 RESULT
  CAT1=$1
  CAT2=$2
  ((RESULT = (CAT1**2 + CAT2**2) ** 1#2))
  echo $RESULT
}
if [ ! $# -eq 2 ]; then
        echo "Incorrect number of arguments"
        exit 1
fi

echo "Square root of "
echo "$1"
echo "$2"
echo "Result"
square "$1" "$2"
exit 0
```

### 1.4.4 Exercise 11.4

```
#!/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
# RETURN CODES: 0-success 1-bad-number-of-args 2-not-a-printable-character.
# HISTORY: First version

# Function declaration
fill_terminal() {

# If number of argument not equal to 1 return 1
# Otherwhise continue with the function
[ $# -ne 1 ] && return 1

        # Local declarations
        local HEXCHAR DECCHAR i j
        # First argument of function
        HEXCHAR=$1
        # Print with format HEXCHAR var
        DECCHAR‘=printf "%d" 0‘x$HEXCHAR
        # var DECCHAR [33,127] limit return 2
        # Otherwise continue function
        if [ $DECCHAR -lt 33 -o $DECCHAR -gt 127 ]; then
                return 2
        fi
        # If columns is a string assign 80
        [ -z "$COLUMNS" ] && COLUMNS=80
        # If lines is a string assign 24
        [ -z "$LINES" ] && LINES=24
        # Minus 2 to linux
        ((LINES-=2))
        # for c style bver number of colums
        for((i=0; i< COLUMNS; i++))
        do
                # for c style of number of lines
                for ((j=0; j< LINES; j++))
                do
                  printf "\x$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

# Using the other script, source all the variables
# to the current terminal (bash) process
# As it has defined one function, the function
# is available to our current context
source fill_terminal_procedure.sh

# Execute function with parameters to the general function
```

### 1.4.5 Exercise 11.5

```
#!/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

#
function search_in_dir() {
        # local var definition
        local fileitem
        # If DEBUG is set print degut message
        [ $DEBUG -eq 1 ] && echo "Entrant a $1"
        # Change directory to the argument passed
        cd $1
        # for over all the elements of the directory
        for fileitem in *
        do
                # If file is a directory
                # run function recursively
                if [ -d $fileitem ]; then
                        search_in_dir $fileitem
                # Not a directory, just print full path
                # of the file, in case is the file we
                # are looking for (FILE_IN_SEARCH)
                elif [ "$fileitem" = "$FILE_IN_SEARCH" ]; then
                        echo ‘‘pwd#$fileitem
                fi
        done
        # If DEBUG is set print degut message
        [ $DEBUG -eq 1 ] && echo "Sortint de $1"
        # Change directory one level above
        cd ..
        }

DEBUG=0
# If no arguments are passed print error and exit
if [ $# -ne 1 ]; then
        echo "Usage: $0 file_to_search"
        exit 1
fi

# File we are looking to search
FILE_IN_SEARCH=$1
# Execute function with pwd command result as argument
# Search will look into our directory the FILE_IN_SEARCH
# file
search_in_dir ‘‘pwd
```

### 1.4.6 Exercise 11.6

```bash
#!/bin/bash
# AUTHOR: opc
# DATE: 29#1#21
# NAME: factorial.sh
# SYNOPSIS: factorial.sh number
# DESCRIPTION: Computes factorial of number given
# HISTORY: First version


if [ ! $# -eq 1 ]; then
       echo "Incorrect number of arguments"
       echo "Usage: .#factorial.sh x"
       exit 1
fi
RESULT=1
NUMBER=$1

while [ $NUMBER -gt 1 ]
do
        RESULT=$(( RESULT * NUMBER ))
        NUMBER=$(( NUMBER - 1))
done
echo $RESULT
```

### 1.4.7 Exercise 11.7

-1.

```
opc :: 4A#ADMINUX#scripts ‹main›* » tail re.txt | grep --line-buffered kernel | cut -c 1-10
```

- I could not acomplish it with the ”tail -f” options, as it seems that the cut command does not flush is contents and nnot output is shown