GESTIÓ DE SISTEMES I XARXES	GRUP: 2F Cristòfol Daudén Esmel Aleix Mariné Tena Josep Marín Llaó Data de lliurament: 20/03/2017

Lab: Shell Scripts i Administració

Només ficarem les capçaleres dels scripts i la funció ajuda en l'explicació del primer script, ja que ocupa molt espai i és una part del codi comuna.

Nom:	**gpgp.sh**	Grau d'assoliment	Alt
Permisos:	rwxr-xr-x	Propietari:	Creador de l'script
Ubicació dels fitxers:	Grup2F/	Grup:	Creador de l'script

Descripció

Aquest script mostrarà per pantalla el path absolut des fitxers (els paths dels quals es troben en cada línia d'un fitxer que passarem com a argument d'entrada), seguit del propietari, grup i permisos que actualment tenen.
Necessita permisos per a llegir el fitxer rebut per paràmetre i permisos per a llegir els fitxers indicats a dins d'aquest primer.
S'ha de tenir en compte que un directori és un fitxer, i guardarem només la informació del directori, no els fitxers sota del directori.

Decisions de Disseny

En aquest script només necessitem permisos de lectura sobre el fitxer passat per paràmetre.
Si l'usuari no introdueix cap argument d'entrada o si l'argument no és un fitxer, se li enviarà un missatge per la sortida d'error i es mostrarà per pantalla la funció ajuda (indicada en la part del codi).
En l'script bàsicament és va llegint el fitxer d'entrada línia a línia (path a path) i es realitza la sortida pertinent.
Hem suposat que el paths indicats en el fitxer que ens passen per paràmetre realment existeixen.

Codi

```
#!/bin/bash
################################################################################
# Autors: Cristòfol Daudèn, Aleix Marine i Josep Marin
# Data d'implementació: 7/2/2018
# Versió 1.0
# Permisos: Aquest script necessita permisos per a llegir el fitxer rebut per paràmetre i
# permisos per a llegir els fitxers indicats a dins del fitxer.
# Descripció i paràmetres: Retorna el propietari, grup i permisos dels fitxers els paths
dels # quals es troben en cada línia del fitxer d'entrada.
# -Argument 1: Ruta completa fins al fitxer que conté la ruta a la resta de fitxers
################################################################################

function ayuda {
```

```
    echo "
################################################################################
# Autors: Cristòfol Daudèn, Aleix Marine i Josep Marin
# Data d'implementació: 7/2/2018
# Versio 1.0
# Permisos: Aquest script necessita permisos per a llegir el fitxer rebut per paràmetre i
# permisos per a llegir els fitxers indicats a dins del fitxer.
# Descripció i paràmetres: Retorna el propietari, grup i permisos dels fitxers els paths
dels # quals es troben en cada línia del fitxer d'entrada.
# -Argument 1: Ruta completa fins al fitxer que conté la ruta a la resta de fitxers
################################################################################
"
}

if [ "$1" = "-h" ]; then
    ayuda
    exit 0
fi

if [ $# -lt 1 ]
then
    echo "ERROR, no file detected" >&2
    ayuda
    exit 1
fi

if [ ! -f "$1" ]
then
  echo "ERROR, parameter is not a file" >&2
  ayuda
  exit 1
fi

IFS=$'\n'
for file in $(cat $1)
do
    echo $file $(stat -c "%U %G %a" $file)
done
exit 0
```

Joc de Proves

Per provar el codi hem executat l'script JocProves.sh on primer s'executa sense arguments de forma que el missatge d'error es redirecciona al fitxer std.err i posteriorment és torna a córrer havent creat una sèrie de carpetes i fitxers, el resultat es guarda en un fitxer std.out.

Output:

Correm el script gpgp.sh sense arguments

```
################################################################################
#
# Autors: Cristòfol Daudèn, Aleix Marine i Josep Marin
# Data d'implementació: 7/2/2018
# Versio 1.0
# Permisos: Aquest script necessita permisos per a llegir el fitxer rebut per paràmetre i
# permisos per a llegir els fitxers indicats a dins del fitxer.
# Descripció i paràmetres: Retorna el propietari, grup i permisos dels fitxers els paths dels
# quals es troben en cada línia del fitxer d'entrada.
```

-Argument 1: Ruta completa fins al fitxer que conté la ruta a la resta de fitxers
##
#

ERROR, no file detected

Correm el script gpgp.sh:
Creem els fitxers dels que donem la direcció
Contingut del fitxer *files* passat per paràmetre:

JocProves
JocProves/a.txt
JocProves/b.txt
JocProves/c.txt
JocProves/d.txt
JocProves/prova

Output
JocProves/a.txt toful toful 644
JocProves/b.txt toful toful 644
JocProves/c.txt toful toful 644
JocProves/d.txt toful toful 644
JocProves/prova toful toful 755

Nom:	**rpgp.sh**	Grau d'assoliment	Alt
Permisos:	rwxr-xr-x	Proprietari:	Creador de l'script
Ubicació dels fitxers:	Grup2F/	Grup:	Creador de l'script

Descripció

Aquest script rep un fitxer per paràmetre (del tipus que generava l'script anterior, gpgp) i demana confirmació per restaurar el propietari, grup i/o permisos dels fitxers en que aquests camps s'hagin modificats.
Faran falta permisos de root quan es vulguin restaurar els atributs de usuari o grup.

Decisions de Disseny

Aquest script necessitarà permisos per llegir el fitxer passat per paràmetre, i en el cas que s'hagin modificat el propietari i/o l'usuari es requeriran permisos de root.
Es comprova que és passi un atribut per paràmetre i que aquest sigui un fitxer.
Es comprovarà també que els paths en el fitxer existeixin i enviarà un missatge d'error quan un s'hagi esborrat.
A més, preguntem (en el cas que s'hagin modificat) si es vol restaurar cada un dels atributs, i en el cas que es vulguin restaurar els atributs de usuari o grup i no es tinguin permisos de root també s'enviarà un missatge d'error.

Codi

```
if [ "$1" = "-h" ]; then
    ayuda
    exit 0
fi
#comprovem que es passi un argument com a entrada
if [ $# -lt 1 ]
then
    echo "ERROR, no file as an argument" >&2
    ayuda
    exit 1
fi
#comprovem que l'argument d'entrada sigui un fitxer
if [ ! -f "$1" ]
then
  echo "ERROR, parameter is not a file" >&2
  ayuda
  exit 1
fi

#per tots els paths en el fitxer d'entrada:
IFS=$'\n'
for file in $(cat $1)
do
    path=$(echo $file | cut -d ' ' -f1)
    #Comprovo si el fitxer existeix
    if [ -e $path ]
    then
         #si existeix comprovo si s'ha modificat algún dels paràmetres i li dino a
l'usuari la
             #possibilitat de restaurar-los
```

```
        user=$(echo $file | cut -d ' ' -f2)
        group=$(echo $file | cut -d ' ' -f3)
        perm=$(echo $file | cut -d ' ' -f4)
        echo $path $(stat -c "%U %G %a" $path)

        if [ $(stat -c "%a" $path) != $perm ]
        then
                echo "S'ha detectat que s'han modificat els permisos, els vols tornar al
seu estat anterior: $perm? (S/N)"
                read r
                if [ $r = 'S' -o $r = 's' ]
                then
                        chmod $perm $path
                        echo "S'han modificat els permisos a: $perm"
                fi
        fi

        if [ $(stat -c "%U" $path) != $user ]
        then
                echo "S'ha detectat que s'ha modificat el propietari, el vols tornar al
seu estat anterior: $user (S/N)"
                read r
                if [ $r = 'S' -o $r = 's' ]
                then
                        if [ $(whoami) != "root" ]
                        then
                                echo "Per poder canviar el propierari del fitxer necessites
permisos de root" >&2
                        else
                                chown $user $path
                                echo "S'ha modificat el propietari a: $user"
                        fi
                fi
                echo "S'ha modificat el propietari a: $user $(ls -l $path)"
        fi

        if [ $(stat -c "%G" $path) != $group ]
        then
                echo "S'ha detectat que s'han modificat el grup, el vols tornar al seu
estat anterior: $group? (S/N)"
                read r
                if [ r = 'S' -o r = 's' ]
                then
                        if [ $(whoami) != "root" ]
                        then
                                echo "Per poder canviar el grup del fitxer necessites
permisos de root" >&2
                        else
                                chgrp $group $path
                                echo "S'ha modificat el grup a: $group"
                        fi
                fi
        fi
    else
        echo "ERROR, el fitxer $path ja no existeix" >&2
    fi
done
exit 0
```

Joc de Proves

Per provar el codi hem exectutat l'script JocProves.sh on s'aprofita la sortida d'una primera execució de l'script gpgp.sh

Fitxers Creats i contingut del fitxer passat per paràmetre a gpgp.sh:
JocProves
JocProves/a.txt
JocProves/b.txt
JocProves/c.txt
JocProves/d.txt
JocProves/prova

Contingut del fitxer que es passarà per paràmetre en les diferents execucions de rpgp.sh:
JocProves/a.txt root root 644
JocProves/b.txt root root 644
JocProves/c.txt root root 644
JocProves/d.txt root root 644
JocProves/prova root root 755

Output:

Correm el script rpgp.sh sense arguments:

#Funció ajuda#

ERROR, no file as an argument

Correm el script rpgp.sh amb la sortida de gpgp.sh sense haver modificat cap fitxer:

Contingut del fitxer:

JocProves/a.txt toful toful 644
JocProves/b.txt toful toful 644
JocProves/c.txt toful toful 644
JocProves/d.txt toful toful 644
JocProves/prova toful toful 755

Correm el script rpgp.sh amb la sortida de gpgp.sh havent modificat el fitxer a.txt i borrat el d.txt.

JocProves/a.txt toful toful 744
S'ha detectat que s'han modificat els permisos, els vols tornar al seu estat anterior: 644? (S/N)
n
JocProves/b.txt toful toful 644
JocProves/c.txt toful toful 644
JocProves/prova toful toful 755

Correm el script rpgp.sh amb la sortida de gpgp.sh havent modificat el fitxer a.txt i b.txt i borrat el d.txt (amb atributs de root):
JocProves/a.txt root root 744

S'ha detectat que s'han modificat els permisos, els vols tornar al seu estat anterior: 644? (S/N)
n
JocProves/b.txt toful root 644
S'ha detectat que s'ha modificat el propietari, el vols tornar al seu estat anterior: root (S/N)
n
JocProves/c.txt root root 644
JocProves/prova root root 755

Correm el script rpgp.sh amb la sortida de gpgp.sh havent modificat el fitxer a.txt i b.txt i borrat el d.txt:
JocProves/a.txt root root 744
S'ha detectat que s'han modificat els permisos, els vols tornar al seu estat anterior: 644? (S/N)
s
S'han modificat els permisos a: 644
JocProves/b.txt toful root 644
S'ha detectat que s'ha modificat el propietari, el vols tornar al seu estat anterior: root (S/N)
s
S'ha modificat el propietari a: root
S'ha modificat el propietari a: root -rw-r--r-- 1 root root 2 mar 9 00:30 JocProves/b.txt
JocProves/c.txt root root 644
JocProves/prova root root 755

Nom:	**aif.sh**	Grau d'assoliment	Alt
Permisos:	rwxr-xr--	Proprietari:	Creador de l'script
Ubicació dels fitxers:	Grup2F/	Grup:	Creador de l'script

Descripció

Aquest script escriu en el fitxer /etc/network/interfaces les següents línies amb els arguments desitjats de configuració passats per paràmetre.

auto $1
iface $1 inet static
address $2
netmask $3
network $4
gateway $5

Decisions de Disseny

Bàsicament es realitzen dos comprovacions: Si s'introdueix el paràmetre "-h" com a primer argument es printa la funció d'ajuda i s'acaba l'execució, sinó es comprova que s'hagin introduït un mínim de 5 arguments per paràmetres.
Es necessitaran permisos de root per executar l'script, ja que el fitxer interfaces és un fitxer del sistema i es requereixen aquests permisos per modificar-lo.

Codi

```
#!/bin/bash

################################################################################################
# Autors: Cristòfol Daudèn, Aleix Mariné i Josep Marín                                         #
# Data d'implementació: 7/2/2018                                                               #
# Versió 1.0                                                                                   #
# Permisos: L'usuari que executa l'script necesita permisos per a modificar el fitxer          #
# /etc/network/interfaces, que normalment posseeix l'usuari root, pel que resulta gairebé      #
# imprescindible ser un usuari amb privilegis per a que funcioni.                              #
# Descripció i paràmetres: Aquest script configura una nova interfície de xarxa, afegint al    #
# fitxer /etc/network/interfaces la informació necessaria. Aquesta info serà aportada pels     #
# paràmetres de la següent manera:                                                             #
# - Argument 1: Nom de la interfície                                                           #
# - Argument 2: Adreça ip                                                                      #
# - Argument 3: Màscara de xarxa                                                               #
# - Argument 4: Adreça de xarxa                                                                #
# - Argument 5: Porta d'enllaç                                                                 #
################################################################################################
```

```
function ayuda {
        echo "
################################################################################
# Autors: Cristòfol Daudèn, Aleix Mariné i Josep Marín                         #
# Data d'implementació: 7/2/2018                                               #
# Versió 1.0                                                                   #
# Permisos: L'usuari que executa l'script necesita permisos per a modificar    #
# el fitxer /etc/network/interfaces, que normalment posseeix l'usuari root,    #
# pel que resulta gairebé imprescindible ser un usuari amb privilegis per a    #
# que funcioni.                                                                #
# Descripció i paràmetres: Aquest script configura una nova interfície de      #
# xarxa, afegint al fitxer /etc/network/interfaces la informació necessaria.   #
# Aquesta info serà aportada pels paràmetres de la següent manera:             #
# - Argument 1: Nom de la interfície                                           #
# - Argument 2: Adreça ip                                                      #
# - Argument 3: Màscara de xarxa                                               #
# - Argument 4: Adreça de xarxa                                                #
# - Argument 5: Porta d'enllaç                                                 #
################################################################################
"
}

function valid_ip()
{
    local  ip=$1
    local  stat=1

    if [[ $ip =~ ^[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}$ ]]; then
        OIFS=$IFS
        IFS='.'
        ip=($ip)
        IFS=$OIFS
        [[ ${ip[0]} -le 255 && ${ip[1]} -le 255 \
            && ${ip[2]} -le 255 && ${ip[3]} -le 255 ]]
        stat=$?
    fi
    return $stat
}

if [ "$1" = "-h" ]; then
       ayuda
       exit 0
fi
if [ $# -lt 5 ]; then
       echo "No has passat el nombre adequat de paràmetres! Mostrant la ajuda..."
       ayuda
       exit 1
fi

# IPs validations
if ! valid_ip $2; then
       >&2 echo "Format error in IP $2"
       exit
fi
if ! valid_ip $3; then
       >&2 echo "Format error in IP $3"
       exit
fi
if ! valid_ip $4; then
       >&2 echo "Format error in IP $4"
       exit
fi
if ! valid_ip $5; then
       >&2 echo "Format error in IP $5"
       exit
```

```
fi

echo -e "\nauto $1\niface $1 inet static\n\taddress $2\n\tnetmask $3\n\tnetwork $4\n\tgateway $5" >> /etc/network/interfaces
exit 0
```

Joc de Proves

S'ha executat l'script passant com a paràmetres els arguments:
enp1s0f0 192.168.17.2 255.255.255.0 192.196.17.0 192.196.17.1
I s'ha observat com en el fitxer interfaces s'han escrit les línies indicades amb els paràmetres passats per argument.

Nom:	**bfit.sh**	Grau d'assoliment	Alt
Permisos:	rwxr-xr-x	Proprietari:	Creador de l'script
Ubicació dels fitxers:	Grup2F/	Grup:	Creador de l'script

Descripció

Aquest script senzillament buida un fitxer el path del qual és passat per paràmetre.

Decisions de Disseny

Bàsicament es realitzen dos comprovacions, si s'introdueix el paràmetre "-h" com a primer argument es printa la funció d'ajuda i s'acaba l'execució, sinó es comprova que s'hagi introduït com a mínim un argument per paràmetre.
L'estratègia empleada és la de realitzar una redirecció buida cap al path del fitxer per buidar-lo. Si el fitxer passat per paràmetre no existeix, es crea buit.
Es necessitaràn permisos d'escriptura sobre el fitxer a buidar, si no es tenen, s'haurà d'executar amb permisos de root.

Codi

```
if [ "$1" = "-h" ]; then
    ayuda
    exit 0
fi
if [ $# -lt 1 ]; then
    echo "No has passat el nombre adequat de paràmetres! Mostrant la ajuda..."
    ayuda
    exit 1
else
    > $1
    exit 0
fi
```

Joc de Proves

S'ha comprovat el seu correcte funcionament amb un fitxer d'exemple:

echo "Aquest fitxer és d'exemple" > exemple.txt
more exemple.txt
Output: `Aquest fitxer és d'exemple`
./bfit.sh exemple.txt
more exemple.txt
Output:
I un fitxer inexistent el crea:
./bfit.sh fitxerinexistent.txt
ls
Output:
`fitxerinexistent.txt`

Lab: Boot i Serveis

Nom:	**Backup**	Grau d'assoliment	Alt
Permisos:	rwxr--r--	Proprietari:	root
Ubicació dels fitxers:	Grup2F/	Grup:	root

Descripció

Servei que s'executa únicament en el nivell 3, de manera que al aturar el sistema es fa una copia de seguretat comprimida per cada usuari del sistema, d'aquells fitxers de l'usuari que estiguin sota del directori /home i que hagin estat modificats des de la còpia realitzada anteriorment.
Es conservaran els permisos i les dates d'accés, creació i modificació dels fitxers dels que fem la còpia
La còpia es deixarà sota un directori anomenat /back/dir on dir serà el nom de l'usuari i el nom del fitxer còpia de seguretat serà l'any, mes i dia en que s'ha fet la còpia en el format: 'aammdd'.

Decisions de Disseny

Per comprimir els fitxers s'ha usat la comanda tar.
Es farà una còpia de seguretat únicament si aquell dia no s'ha fet, d'aquí hem considerat quatre casos: un en que ja s'ha fet la copia el mateix dia, un que no existeix la carpeta /home/usuari, un altre en que existeix però esta buida i finalment el cas en que hi ha còpies de seguretat d'altres dies.
Per assegurar-nos que només l'usuari propietari dels fitxer pugui accedir a ells, només donem permisos a de lectura, escriptura i execució al camp usuari.
Farem que es pari el servei (execució de la funció de backup) quan el sistema canvii als nivells 0 i 6 (reboot i shutdown) i que s'engegui quan canvii a l'estat 3.
La funció backup, el primer que fa és comprobar si el nivell anterior era el 3 amb la comanda *runlevel,* però estan en el nivell 0 o 6 s'executa com si estes en el nivell anterior, de forma que només s'executaria el servei de backup si del nivell 3 canviéssim a un altre nivell diferent de 0, 3 o 6 i llavors tornéssim a canviar a l'estat 0 o 6.
Per solucionar-ho, hem considerat que la comanda *runlevel* s'executa com si s'estès en l'estat anterior (nivell des d'on apaguem l'ordinador), de forma que agafem el segon paràmetre que retorna enlloc del primer.
Per poder instalar el servei faran falta permisos de root.

Codi

```
#!/bin/bash
# Start/stop the backup service.
#
### BEGIN INIT INFO
# Provides:          backup
# Required-Start:    $remote_fs $syslog $time
# Required-Stop:     $remote_fs $syslog $time
# Should-Start:
# Should-Stop:       0,6
```

```
# Default-Start:    3
# Default-Stop:        0,6
# Short-Description: Backup service program
# Description:         backup is a program that runs when the machine is shutted
#               down creating a compressed system backup for every user in
#               the system.
### END INIT INFO

PATH=/bin:/usr/bin:/sbin:/usr/sbin
DESC="backup service"
NAME=backup
SCRIPTNAME=/etc/init.d/"$NAME"

backup_service(){
    echo "S'executa el codi $(date)" > /home/milax/execucio.txt
    echo "runlevel: $(runlevel)" >> /home/milax/execucio.txt
    #comprovem que l'estat d'origen sigui l'estat 3
    if [ $(runlevel | cut -d ' ' -f2) -eq 3 ]
    then
        #per tots els usuaris en la carpeta /home/:
        for user in $(ls /home)
        do
           #comprovem que sigui un usuari
           id $user > /dev/null >&2
           if [ $? -eq 0 ]
           then
               #comprovem que no hi hagi un backup del mateix dia de l'usuari
               if [ -e /back/$user/$(date +%y%m%d).tar.gz ]
               then
                  #cas en que ja s'ha fet un backup aquest dia
                  echo "Avui ja s'ha fet una copa de seguretat de l'usuari $user" >&2
                  else
                     if [ -d /back/$user ]
                     then
                         if [ $(ls -l /back/$user/ | wc -l) -gt 1 ]
                         then
                            #cas en que hi ha un backup d'un dia anterior
                            #obtenim el nom de l'últim backup realitzat
                            anterior=$( ls -t /back/$user/ | sed -n '1p' )
                            #comprimim  aquells  fitxers  que  s'han  actualitzat  des  de
l'últim backup
                            find    /home/$user    -type    f    -user    $user    -newer
/back/$user/$anterior > /tmp/new_files.txt
                            tar     -czf     /back/$user/$(date     +%y%m%d).tar.gz     -T
/tmp/new_files.txt
                            chmod 700 /back/$user/$(date +%y%m%d).tar.gz
                            chown $user:$user /back/$user/$(date +%y%m%d).tar.gz
                          else
                            #cas en que la carpeta amb el nom de l'usuari esta creada
pero no hi ha cap backup
                            find /home/$user -type f -user $user > /tmp/new_files.txt
                            tar     -czf     /back/$user/$(date     +%y%m%d).tar.gz     -T
/tmp/new_files.txt
                            chmod 700 /back/$user/$(date +%y%m%d).tar.gz
                            chown $user:$user /back/$user/$(date +%y%m%d).tar.gz
                         fi
                       else
                        #Cas en que no hi ha cap carpeta amb el nom de l'usuari
                         mkdir -p /back/$user
                        find /home/$user -type f -user $user > /tmp/new_files.txt
                                tar    -czf    /back/$user/$(date    +%y%m%d).tar.gz    -T
/tmp/new_files.txt
                        chmod 700 /back/$user/$(date +%y%m%d).tar.gz
                        chown $user:$user /back/$user/$(date +%y%m%d).tar.gz
```

```
                        fi
                fi
         fi
    Done
fi
}

case "$1" in
start)
        exit 0
    ;;
stop)
        backup_service
        ;;
status)
        ;;
*)
        exit 2
        ;;
esac
exit 0
```

Joc de Proves

Per provar el servei, he realitzat la seva instal·lació mitjançant el script backup_installer.sh (també adjuntat) en una màquina virtual amb Milax.
Durant una setmana s'ha anat engegant i apagant diversos cops, de forma que només crea una copia al dia i només amb aquells fitxers que s'han modificat des de l'últim backup. A més, el propietari dels fitxers creats és l'usuari al qual pertanyien, de la mateixa forma només ell té permisos per llegir-los, modificar-los o executar-los.

S'han provat els diferents escenaris considerats:

- Ja s'ha fet la copia el mateix dia: No es fa cap copia
- No existeix la carpeta /home/usuari: Es crea la carpeta i es fan les còpies de seguretat.
- Existeix /home/usuari però esta buida: Es fan les còpies de seguretat.
- Hi ha còpies de seguretat d'altres dies: Es fan còpies de seguretat de tot allò que hagi estat modificat.

Nom:	**Shutdown programat**	Grau d'assoliment	Alt
Permisos:	rwxr-xr-x	Proprietari:	Creador de l'script
Ubicació dels fitxers:	Grup2F/	Grup:	Creador de l'script

Descripció

El sistema s'atura cada divendres a les 10 de la nit. S'implementa de dos formes diferents, mitjançant el **daemon del cron** i utilitzant els **timers del *systemd***.

Comparativa:

La principal característica que diferencia els *timers* del *systemd,* és que aquests, al estar destinats en la gestió de recursos del sistema, són serveis sistemàtics amb totes les seves capacitats i permisos adients, com per a accions com programació de CPU IO...

Cron es remunta a finals de 1970 mentre que systemd més recent.

Per tant el crontab, en contrast, al estar destinat a l'execució programada de tot tipus de comandes, comporta moltes més configuracions per tasques que requereixin serveis addicionals, com la comunicació amb altres programes.

Decisions de Disseny

Daemon del cron:

Utilitzant el daemon del cron s'insereix una entrada a la taula 'crontab', la qual executa la comanda `shutdown`. Com que aquesta comanda requereix permisos d'administrador, es configura el crontab del `root`, que és diferent de la d'usuari i per tant té els permisos necessaris; s'accedeix a través del `sudo` i especificant el subdirectori estandard del `root`, `/sbin`.

Codi

Daemon del cron:

Accés al crontab del root.

```
$sudo crontab -l
```

Adició de l'entrada a la crontab. S'ha d'ex

```
00 22 * * 5 /sbin/shutdown -h now
```

Utilitzant els timers del systemd:

1- Al directori `/etc/systemd/system/` s'afegeix:

```
[Unit]
Description=Shutdown timer systemd
[Service]
ExecStart=/usr/local/bin/GSX/shutdown-timersystemd
```

2- Al directori on es troba l'Script, al mateix path:

```
[Unit]
Description= Every friday at 22, the computer will be shutted down.
[Timer]
# Time to wait.
OnCalendar=Fri *-*-* 22:00:00
```

```
OnUnitActiveSec=1h
Unit=myscript.service
[Install]
WantedBy=multi-user.target
```

3- Per activar-lo (seria el mateix per desactivar-lo), s'executa la comanda:

```
$ systemctl start myscript.timer
```

Quan es configura al timer del systemd, ja es disposen dels permisos pertinents.

Conclusions: crond is much easier to use. In comparison, for systemd you have to create 3 files (one for the timer, one for the service and one for the script that you want to execute) and then you have to start the timer. It's much easier to just modify the crontab file and you can even write a command directly there, you don't need an extra script file. This also means that less dependencies are added to the system with just one file, and less disk space is used.

Joc de Proves

Hem provat tant el daemon cron com els timers de systemctl per a executar s'script sd.sh en un temps pròxim i efectivament la màquina s'ha apagat.

Lab: Gestió de processos

Nom:	**limit_cpu_use.sh**	Grau d'assoliment	Alt
Permisos:	rwxr--r--	Proprietari:	root
Ubicació dels fitxers:	Grup2F/	Grup:	root

Descripció

Aquest script limitarà a l'usuari que li passem per paràmetre a l'ús d'una única CPU, també passada per paràmetre.

Decisions de Disseny

Es passa com a primer argument la CPU a la qual se li limitarà l'us a l'usuari i com a segon argument l'usuari.
Controlarem que es passin dos arguments per paràmetre, que la CPU indicada estigui dins del rang permès per la màquina i que l'usuari passat per paràmetre existeixi.
El que farem serà crear un nou cgroup amb el nom de l'usuari en /sys/fs/cgroup/, assignarem la CPU i memòria i escriurem en el fitxer tasks la shell pare de l'usuari.
Aquest script s'ha d'executar desde root, sinó no es pot escriure el PID de la primera bash shell en el fitxer tasks del cgroup creat.
Aquest script només serveix per aquells usuaris que inicien el sistema amb una shell bash, per fer-ho més genèric hauríem de buscar en el /etc/passwd la primera shell que s'executa i copiar el seu PID en el fitxer tasks.

Codi

```
if [ "$1" = "-h" ]; then
    ayuda
    exit 0
fi

#mirem que s'hagin introduit tots els arguments
if [ $# -lt 2 ]
then
    echo "ERROR, you have to indroduce two argumnets" >&2
    ayuda
    exit 1
fi

#mirem que la CPU indicada estigui entre les possibles de la màquina
cpus=$(cat /sys/fs/cgroup/cpuset/cpuset.cpus)
if [[ $1 =~ ^[^$cpus]$ ]]
then
    echo "ERROR, CPU especified isn't available" >&2
    ayuda
    exit 1
fi

#comprovem que l'usuari existeixi
id $2 > /dev/null >&2
if [ $? -eq 0 ]
then
    #creem un group per l'usuari
    mkdir /sys/fs/cgroup/cpuset/$2
    echo $1 > /sys/fs/cgroup/cpuset/$2/cpuset.cpus
```

```
    echo 0 > /sys/fs/cgroup/cpuset/$2/cpuset.mems
   echo $(ps -ef | egrep -e "^$2.*bash" | tr -s ' ' | sed -n '1p' | cut -d ' ' -f3) >
/sys/fs/cgroup/cpuset/$2/tasks
    exit 0
else
    echo "ERROR, user doesn't exists" >&2
    ayuda
    exit 1
fi
```

Joc de Proves

Executem l'script i es crea el nou grup en la carpeta /sys/fs/cgroup/cpuset i amb el fitxers cpuset.cpus, cpuset.mems i tasks amb el contingut desitjat.
Si l'usuari que introduïm no existeix es mostra missatge d'error. El mateix passa si seleccionem una CPU que no tenim (per exemple: en un quad-core, la màxima CPU seleccionable és la 3).

Nom:	**processes_limit.sh**	Grau d'assoliment	Alt
Permisos:	rwxr-xr-x	Proprietari:	root
Ubicació dels fitxers:	Grup2F/	Grup:	root

Descripció

Aquest *script* suspèn els processos que han consumit més de 5 minuts o ocupen més de 1GB de memòria resident. Es programara (amb el crontab) l'execució per aturar-los a les 8 del mati, i despres es tornara a cridar a les 9 de la nit per rependre'ls durant la nit.

Decisions de Disseny

Per gestionar els períodes de temps de aturar o/i rependre'ls s'utilitza el Daemon del crontab. D'aquesta manera, mitjançant un argument, s'indica quan s'ha de rependre i quan s'ha d'aturar: Al mati es programa una entrada al crontab amb l'argument s (suspend) i per la nit r (resume). Es controla que es passin l'argument correcte.
Llavors mitjançant la comanda ps aux, es capturen els processos i es filtren, consultant la columna corresponent de la informacio del proces i aquells que consumeixen superior al limit de memoria o de temps d'us de la CPU es guarden a un Array, tot imprimint-los per indicar-li a l'usuari, i a continuacio es suspenen. Llavors es guarda aquest array en un fitxer de text per tal que a l'hora de rependre'ls es consulten i es crida a la comanda kill -CONT , tot imprimint-los.

Codi

```
if [ "$1" = "-h" ]; then
	ayuda
	exit 0
fi

#mirem que s'hagi introduit l'argument.
if [ $# -lt 1 ]
then
	echo "ERROR, you have to indroduce one argumnet" >&1
	ayuda
	exit 1
fi

#Suspendre processos. Argument "-s".
if [ "$1" = "-s" ]
	then
	echo "The following processes will be suspended."
	ps aux | awk '$10>="5:00"||$5>=1000000{print $0}' # Filtra els processos que consumeixin mes de 1GB de mem o que hagin estat en execucio mes de 5 min, tot imprimint-los.
	processes=($(ps aux | awk '$10>="5:00"||$5>=1000000{print $2}')) # Filtra els processos amb el mateix criteri i guardar-los en un array.
	echo ${processes[@]} > $BASEDIR/suspended_process.txt   # Guarda l'array en un fitxer.
	for i in "${processes[@]:1}"
	do
		kill -STOP $i # Suspendre cada proces de l'Array.
	done
#Suspendre processos. Argument "-r".
```

```
elif [ "$1" = "-r" ]
then
	echo "The following processes (listed by PID) will be resumed."
	mapfile -t processes < $BASEDIR/suspended_process.txt    # Obten l'array de
processos suspesos a partir del fitxer de text.
	processes=($(echo ${processes[0]} | tr " " "\n"))
	for i in "${processes[@]:1}"
	do
		kill -CONT $i # Repen cada proces de l'Array.
		echo $i # Mostra el PID de cada proces repres
	done
else # Gesstio d'errors si s'introdueix un argument invalid.
	echo "ERROR, you have to introduce a valid argument: -s or -r"
	exit 1
fi

exit 0
```

Joc de Proves

Programem el crontab per executar l'script en un període determinat, passant-li l'argument pertinent, primer el de suspendre els processos, i comprovem com es suspenen els processos en qüestió. De la mateixa manera a l'hora de reprendre'ls.

Nom:	**nofork.sh**	Grau d'assoliment	Alt
Permisos:	rwxr-xr-x	Propietari:	Creador de l'script
Ubicació dels fitxers:	Grup2F/	Grup:	Creador de l'script

Descripció

Aquest script mostra UN SOL COP el nom de totes les comandes ordenades alfabéticament que s'han engegat entre les 13:00 i les 14:59 i que han dut a terme la crida al sistema fork sense haver dut a terme un exec a posteriori, així com l'usuari que ha executat aquestes comandes.

Decisions de Disseny

Sense cap argument aquest script mostra les comandes indicades un sol cop, de tal manera que si diferents execucions de la mateixa comanda compleixen els requeriments, aquesta comanda només apareixerà un cop junt al nom de l'usuari que l'ha executat.
D'aquesta manera, la informació que mostra l'script es menor i més fàcil per a treballar-hi.
Si es necessita més informació es pot especificar l'argument -r per a mostrar en quin moment exacte s'ha executat la comanda, així com les diferents possibles execucions de la mateixa comanda en aquest període de temps.

S'ha utilitzat la comanda lastcomm que llegeix un fitxer de log de sistema, es per això que lastcomm requereix privilegis elevats, de tal manera que al nostre script requereix ser executat en mode de superusuari..
Argument 1: -[h|r]

Codi

```
if [ $# -le 1 ]; then
        if [ "$1" = "-h" ]; then
                ayuda
                exit 0
        elif [ "$1" = "-r" ]; then
                lastcomm | grep -E "^.* .F... .* 1[3,4]:..$" | cut -c1-17,24-32,65-69 |
sort
                exit 0
        elif [ "$1" = "" ]; then
                lastcomm | grep -E "^.* .F... .* 1[3,4]:..$" | cut -c1-17,24-32 | sort |
uniq
                exit 0
        else
                echo -E "L'argument $1 no és reconeix. Mostrant l'ajuda..."
                ayuda
                exit 1
        fi
else
        echo "Hi ha massa arguments. Mostrant l'ajuda..."
        ayuda
        exit 1
fi
```

Joc de Proves

```
sudo ./nofork.sh
```

Output

```
apache2ctl       root
apache2          root
bash             milax
console-kit-dae  root
cron             root
dbus-daemon      messageb
dbus-daemon      milax
dbus-launch      milax
debian-start     root
Default          root
dndHGCM          milax
ethtool          root
file-roller      milax
git-pull         milax
gnome-settings-  milax
gnome-sound-app  milax
gvfsd            milax
iceweasel        milax
isc-dhcp-server  root
kworker/0:0      root
kworker/0:1      root
kworker/0:2      root
kworker/u:0      root
kworker/u:10     root
kworker/u:11     root
kworker/u:12     root
kworker/u:13     root
kworker/u:14     root
kworker/u:15     root
kworker/u:16     root
kworker/u:17     root
kworker/u:18     root
kworker/u:19     root
kworker/u:1      root
kworker/u:20     root
kworker/u:21     root
kworker/u:22     root
kworker/u:23     root
kworker/u:24     root
kworker/u:25     root
kworker/u:26     root
kworker/u:27     root
kworker/u:29     root
kworker/u:2      root
kworker/u:30     root
kworker/u:3      root
kworker/u:4      root
kworker/u:5      root
kworker/u:6      root
kworker/u:7      root
kworker/u:8      root
kworker/u:9      root
laptop-mode      root
lsof             root
mountnfs         root
mysqld_safe      root
mysql            root
nautilus         milax
ntop             root
```

```
pam-foreground-  root
pm-is-supported  root
pm-powersave     root
powerbtn-acpi-s  root
quagga           root
rcvboxadd-x11    root
sh               root
shutdown         root
snmpd            root
ssh-agent        milax
ssh              root
start-stop-daem  root
udevd            root
upowerd          root
VBoxClient       milax
VBoxService      root
wpasupplicant    root
xscreensaver     milax
Xsession         milax

sudo ./nofork.sh -r
```

Output

```
...
VBoxClient       milax    13:26
VBoxClient       milax    13:26
VBoxClient       milax    13:26
VBoxClient       milax    13:26
VBoxClient       milax    13:26
VBoxClient       milax    13:26
VBoxClient       milax    13:26
VBoxClient       milax    13:26
VBoxClient       milax    13:26
VBoxClient       milax    13:26
VBoxClient       milax    13:26
VBoxClient       milax    13:26
VBoxClient       milax    13:26
VBoxClient       milax    13:26
VBoxClient       milax    13:26
VBoxClient       milax    13:26
VBoxClient       milax    13:26
VBoxClient       milax    13:26
VBoxClient       milax    13:26
VBoxClient       milax    13:26
VBoxClient       milax    13:26
VBoxClient       milax    13:26
VBoxClient       milax    13:41
VBoxClient       milax    13:41
VBoxClient       milax    13:41
VBoxClient       milax    13:41
VBoxClient       milax    13:41
VBoxClient       milax    13:41
VBoxClient       milax    13:41
VBoxClient       milax    13:42
VBoxClient       milax    13:42
VBoxClient       milax    13:42
VBoxClient       milax    13:42
VBoxClient       milax    13:42
VBoxClient       milax    13:42
VBoxClient       milax    13:42
VBoxClient       milax    13:42
VBoxClient       milax    13:42
VBoxClient       milax    13:42
VBoxClient       milax    13:42
```

```
VBoxClient       milax    13:42
VBoxClient       milax    13:42
VBoxClient       milax    13:42
VBoxClient       milax    13:42
VBoxClient       milax    14:16
VBoxClient       milax    14:16
VBoxClient       milax    14:16
VBoxClient       milax    14:16
VBoxClient       milax    14:16
VBoxClient       milax    14:16
VBoxClient       milax    14:16
VBoxClient       milax    14:16
VBoxClient       milax    14:16
VBoxClient       milax    14:16
VBoxClient       milax    14:16
VBoxClient       milax    14:16
VBoxClient       milax    14:16
VBoxClient       milax    14:16
VBoxClient       milax    14:16
VBoxClient       milax    14:16
VBoxClient       milax    14:16
VBoxClient       milax    14:16
VBoxClient       milax    14:16
VBoxClient       milax    14:16
VBoxClient       milax    14:16
VBoxClient       milax    14:16
VBoxClient       milax    14:17
VBoxClient       milax    14:17
VBoxClient       milax    14:17
VBoxClient       milax    14:17
VBoxClient       milax    14:17
VBoxClient       milax    14:17
VBoxClient       milax    14:17
VBoxClient       milax    14:18
VBoxClient       milax    14:18
VBoxClient       milax    14:18
VBoxClient       milax    14:18
VBoxClient       milax    14:18
VBoxClient       milax    14:18
VBoxClient       milax    14:18
VBoxClient       milax    14:18
VBoxClient       milax    14:18
VBoxClient       milax    14:18
VBoxClient       milax    14:18
VBoxClient       milax    14:18
VBoxClient       milax    14:18
VBoxClient       milax    14:18
VBoxClient       milax    14:18
VBoxClient       milax    14:51
VBoxClient       milax    14:51
VBoxClient       milax    14:51
VBoxClient       milax    14:51
VBoxClient       milax    14:51
VBoxClient       milax    14:51
VBoxClient       milax    14:51
VBoxClient       milax    14:53
VBoxClient       milax    14:53
VBoxClient       milax    14:53
VBoxClient       milax    14:53
VBoxClient       milax    14:53
VBoxClient       milax    14:53
VBoxClient       milax    14:53
VBoxClient       milax    14:53
VBoxClient       milax    14:53
VBoxClient       milax    14:53
```

```
VBoxClient       milax    14:53
VBoxClient       milax    14:53
VBoxClient       milax    14:53
VBoxClient       milax    14:53
VBoxClient       milax    14:53
VBoxClient       milax    14:53
VBoxClient       milax    14:53
VBoxClient       milax    14:53
VBoxClient       milax    14:53
VBoxClient       milax    14:53
VBoxClient       milax    14:53
VBoxClient       milax    14:53
VBoxClient       milax    14:55
VBoxClient       milax    14:55
VBoxClient       milax    14:55
VBoxClient       milax    14:55
VBoxClient       milax    14:55
VBoxClient       milax    14:55
VBoxClient       milax    14:55
VBoxClient       milax    14:55
VBoxClient       milax    14:55
VBoxClient       milax    14:55
VBoxClient       milax    14:55
VBoxClient       milax    14:55
VBoxClient       milax    14:55
VBoxClient       milax    14:55
VBoxClient       milax    14:55
VBoxClient       milax    14:55
VBoxClient       milax    14:55
VBoxClient       milax    14:55
VBoxClient       milax    14:55
VBoxClient       milax    14:55
VBoxClient       milax    14:55
VBoxClient       milax    14:55
VBoxService      root     13:26
VBoxService      root     13:42
VBoxService      root     14:16
VBoxService      root     14:18
VBoxService      root     14:53
VBoxService      root     14:55
wpasupplicant    root     13:25
wpasupplicant    root     13:41
wpasupplicant    root     14:16
wpasupplicant    root     14:17
wpasupplicant    root     14:53
wpasupplicant    root     14:54
xscreensaver     milax    13:04
xscreensaver     milax    13:14
xscreensaver     milax    14:28
Xsession         milax    13:26
Xsession         milax    13:26
Xsession         milax    13:41
Xsession         milax    13:41
Xsession         milax    14:16
Xsession         milax    14:16
Xsession         milax    14:17
Xsession         milax    14:17
Xsession         milax    14:53
Xsession         milax    14:53
Xsession         milax    14:55
Xsession         milax    14:55
...
```

Lab: Gestió d'usuaris

Nom:	**altausers.sh**	Grau d'assoliment	Mitjà
Permisos:	rwx------	Proprietari:	root
Ubicació dels fitxers:	Grup2F/	Grup:	root

Descripció

L'script altausers.sh llegeix del fitxer passat per paràmetre. De cada línia extreu la informació d'un usuari, que inclou DNI, telèfon, nom, cognom i una llista de grups i CPUs assignades a cada grup. Com a mínim hi haurà un grup i una CPU en aquesta llista, que serà el grup primari de l'usuari.

El nom de l'usuari serà el seu nom acompanyat per les inicials dels seus cognoms. Si hi ha un usuari que necessita el mateix nom d'usuari llavors se li afegeix un nombre que es va incrementant.

Per a comprovar que no hi hagi usuaris repetits es comprova el DNI de la persona, que s'emmagatzema com a comentari en el fitxer /etc/passwd. No hi pot haver dues persones amb el mateix DNI.

Decisions de Disseny

Hem considerat que les línies d'entrada del fitxer són correctes. Es consulten diferents fitxers per a dur a terme les accions, per això s'han parametritzat utilitzant variables que es troben a dins del codi.
La variable CONF_PATH per exemple, inclou la direcció d'un fitxer d'on podem extreure el directori base per defecte, els limits del gid i uid i la shell predeterminada. Aquesta serà /root/conf/.usuaris per defecte
La variable LOGIN_DEFAULT inclou la direcció /etc/login.defs que conté els valors predetrminats de login per als usuaris.
La variable BASE_DIR indica el directori base per establir el directori dels usuaris.
L'script s'encarrega de modificar els valors dels límits per a GID i UID en /etc/login.defs i de tractar les dades que rep del fitxer per a crear l'usuari i els grups corresponents.
Es necessiten privilegis de superusuari degut a que la creació d'usuaris ho requereix.
S'han complit tots els objectius que demanava l'enunciat excepte la limitació de la CPU associada a un determinat grup.

Codi

```
 #!/bin/bash

################################################################################
# Autors: Cristòfol Daudèn, Aleix Marine i Josep Marín Llaó
# Data d'implementació: 13/3/2018
# Versio 1.0
# Permisos:    root
# Descripció i paràmetres: Es vol implementar un script altausers per donar
# d'alta varis usuaris en una sola execució. La informació dels usuaris serà
```

```
# rebuda per paràmetre a través d'un fitxer
################################################################################

function ayuda {
        echo "
################################################################################
# Autors: Cristòfol Daudèn, Aleix Marine i Josep Marín Llaó
# Data d'implementació: 13/3/2018
# Versio 1.0
# Permisos:    root
# Descripció i paràmetres: Es vol implementar un script altausers per donar
# d'alta varis usuaris en una sola execució. La informació dels usuaris serà
# rebuda per paràmetre a través d'un fitxer
################################################################################
"
}

getline () { awk -vlnum=${1} 'NR==lnum {print; exit}'; }

CONF_PATH=/root/conf/.usuaris
LOGIN_DEFAULT=/etc/login.defs
BASE_DIR=/usuaris

#mirem que s'hagin introduit tots els arguments
if [ $# -lt 1 ]; then
        echo "Error, you need to introduce an argument!"
        ayuda
        exit 1
fi

#Mirem parametre d'ajuda
if [ "$1" = "-h" ]; then
        ayuda
        exit 0
fi

# Comprovem que es tinguin permisos de root
if [ $(whoami) != "root" ]; then
        echo "ERROR, Root permissions needed"
        exit 1
fi

# Comprovem existencia de directoris
if [ ! -d "/root" ]; then
        echo -e "\nError, directory /root does not exist. Creating..."
        mkdir /root
fi

if [ ! -d "/root/conf" ]; then
        echo -e "\nError, directory /root/conf does not exist. Creating..."
        mkdir /root/conf
fi

if [ ! -f "/root/conf/.usuaris" ]; then
        read
        echo -e "\nError, file /root/conf/.usuaris does not exist. Creating..."
        echo "" > /root/conf/.usuaris
        echo -e "\nNow /root/conf/.usuaris is empty, introduce correct data as specified
in headers and execute the script later."
        exit 1
fi

if [ ! -f "$1" ]; then
        echo -e "\nError, file $1 specified in argument, does not exist or is not a valid
file. "
```

```
        echo -e "\nSpecify a valid file and execute this script"
        exit 2
fi

if [ ! -d $BASE_DIR ]; then
        mkdir $BASE_DIR
fi

# reading input from file CONF_PATH and defining limits of uid and gid in login.defs
skel=$(more $CONF_PATH | getline 1)

if [ ! -d $skel ]; then
        skel="/home"
fi

uidmin=$(more $CONF_PATH | getline 2 | cut -d "-" -f1)
if [ -z $uidmin ]; then
        uidmin=100
fi
# comprovar els parametres i assignar per defecte.
num=$(grep ^"UID_MIN" /etc/login.defs | tr -s "\t" | tr '\t' ' ' | tr -s " " | cut -d ' '
-f2)
sed -i "/UID_MIN/ s/$num/$uidmin/g" "$LOGIN_DEFAULT"

read
uidmax=$(more $CONF_PATH | getline 2 | cut -d "-" -f2)
if [ -z $uidmax ]; then
        uidmax=1000
fi


num=$(grep ^"UID_MAX" /etc/login.defs | tr -s "\t" | tr '\t' ' ' | tr -s " " | cut -d ' '
-f2)
sed -i "/UID_MAX/ s/$num/$uidmax/g" "$LOGIN_DEFAULT"


gidmin=$(more $CONF_PATH | getline 3 | cut -d "-" -f1)
if [ -z $gidmin ]; then
        gidmin=200
fi
num=$(grep ^"GID_MIN" /etc/login.defs | tr -s "\t" | tr '\t' ' ' | tr -s " " | cut -d ' '
-f2)
sed -i "/GID_MIN/ s/$num/$gidmin/g" "$LOGIN_DEFAULT"

gidmax=$(more $CONF_PATH | getline 3 | cut -d "-" -f2)
if [ -z $gidmax ]; then
        gidmax=2000
fi
num=$(grep ^"GID_MAX" /etc/login.defs | tr -s "\t" | tr '\t' ' ' | tr -s " " | cut -d ' '
-f2)
sed -i "/GID_MAX/ s/$num/$gidmax/g" "$LOGIN_DEFAULT"

shell=$(more $CONF_PATH | getline 4)
if [ ! -f $shell ]; then
        shell="/bin/bash"
fi

# Add user/s
while IFS="," read DNI NAME SUR1 SUR2 TLFN GRUPS
do
        USR_NAME="$NAME${SUR1:0:1}${SUR2:0:1}"
        aux="$USR_NAME"
        cont=0

        # Check if user is already registered
```

```
	if [ ! -z $(cat /etc/passwd | grep -w $DNI) ]; then #Si la linia on trobem el DNI
de
		echo "The user with DNI $DNI is already registered." >&2
		continue # instruccio break; i passem al seguent user
	fi

	# Creacio del nom d'usuari
	id $aux > /dev/null 2> /dev/null

	while [ "$?" -eq "0" ]; do # Mirem si aux es un nom adequat i sino incrementem
comptador i generem nou nom
		aux="$USR_NAME$cont"
		cont=$((cont + 1))
		id $aux > /dev/null 2> /dev/null
	done

	# tractem el primer grup de manera especial degut a que es el grup primari
	if [ ! -z $GRUPS ]; then
		PRIMARY_GROUP=$(echo -e $GRUPS | cut -d ',' -f1)
		groupadd -f $PRIMARY_GROUP
		PRIMARY_CPU=$(echo -e $GRUPS | cut -d ',' -f2)
		#./limit_cpu_use.sh $PRIMARY_GROUP $PRIMARY_CPU
	fi
	# creem la resta de grups de cpu amb la cpu que ens passen aprofitant la
implementacio de l'altre script
	SECONDARY_GROUPS=""
	GRUPS=$(echo $GRUPS | cut -d ',' -f3-)
	while [ ! -z $GRUPS ]; do
		GROUP=$(echo $GRUPS | cut -d ',' -f1)
		groupadd -f $GROUP
		SECONDARY_GROUPS="$SECONDARY_GROUPS,$GROUP"
		CPU=$(echo $GRUPS | cut -d ',' -f2)
		GRUPS=$(echo $GRUPS | cut -d ',' -f3-)
		#./limit_cpu_use.sh $GROUP $CPU
	done

	SECONDARY_GROUPS="${SECONDARY_GROUPS:1}" # eliminem la coma inicial

	# b basename path a on situarem la nostra carpeta (vindria a er home per defecte)
	# c password
	# d directori de l'usuari sol ser /home/[nom usuari]
	# g grup primari de l'usuari (ha d'existir)
	# G grup de directoris secundaris (separats per coma)
	# k directori esquelet (es copiara tot el contingut d'aquest directori al
direcotori d'entrada)
	# m força la creacio de del direcotri de l'usuari
	# p password
	# s shell per defecte

	useradd -b "/usuaris" -c "$DNI" -d "/usuaris/$aux" -g "$PRIMARY_GROUP" -G
"$SECONDARY_GROUPS" -k "$skel" -m -p "$TLFN" -s "$shell" "$aux"
	#echo "$aux:$TLFN" | chpasswd
done < $1
exit 0
```

Joc de Proves

S'ha escrit el fitxer users.txt amb el següent contingut

```
39933490-M,Aleix,sarine,Tena,977352524,panolitos,1,la,2,vida,3,es,4,una,2,merda,3
39932490,Aleix,Marine,Tena,977352524,poia,1
32222222,Tofulet,el,fdp,6666666666,poiancre,1
362782282,oscarsekando,mama,papa,7878,poia,2
44544,nosufras,mas,mas,7,poiancre,0
```

```
39933490-M,Aleix,Marine,Antena,87678,pillastre,2
3993344444,demon,in,sky,666,demon,5,hell,6,adolf'sresidence,4
45,Aleix,Makina,Torrencial,666,demoniacoespapa,4
45,Aleix,Motorista,Tempestuoso,32,panolitos,5,vida,4
23456-1,Anacleto,Maquinista,Tomahawk,789878987,indiosalvalle,7,una,7,merda,2
2345,Aleix,Mosquetero,Tronista,789878876,hola,3,poni,3
```

Aquest fitxer conté usuaris inventats. Al cridar al nostre script i passarli per paràmetre ens dòna la següent sortida.

```
milax@casa:~/Escriptori/GSX_Labs/Prac1/altausers.sh$ sudo ./altausers.sh /users.txt
The user with DNI 39933490-M is already registered.
The user with DNI 45 is already registered.
```

El fitxer /etc/passwd té el següent contingut

```
AleixsT:x:111:1002:39933490-M:/usuaris/AleixsT:/bin/bash
AleixMT:x:112:1001:39932490:/usuaris/AleixMT:/bin/bash
Tofuletef:x:113:1008:32222222:/usuaris/Tofuletef:/bin/bash
oscarsekandomp:x:114:1001:362782282:/usuaris/oscarsekandomp:/bin/bash
nosufrasmm:x:115:1008:44544:/usuaris/nosufrasmm:/bin/bash
demonis:x:116:1009:3993344444:/usuaris/demonis:/bin/bash
AleixMT0:x:117:1012:45:/usuaris/AleixMT0:/bin/bash
AnacletoMT:x:118:1013:23456-1:/usuaris/AnacletoMT:/bin/bash
```

I el fitxer /etc/group té el següent contingut

```
poia:x:1001:
panolitos:x:1002:
la:x:1003:AleixsT
vida:x:1004:AleixsT
es:x:1005:AleixsT
una:x:1006:AleixsT,AnacletoMT
merda:x:1007:AleixsT,AnacletoMT
poiancre:x:1008:
demon:x:1009:
hell:x:1010:demonis
adolf'sresidence:x:1011:demonis
demoniacoespapa:x:1012:
indiosalvalle:x:1013:
```

La carpeta /usuaris s'ha plenat dels directoris home

```
ls /usuaris
AleixMT   AleixsT     demonis     oscarsekandomp
AleixMT0  AnacletoMT  nosufrasmm  Tofuletef
```

Per tant podem veure com l'script ha creat els grups així com l'estructura de directoris per a cada usuari. A dins de la carpeta d'usuari s'hi troben els fitxers a la carpeta skel tal i com es demana. Els GID i UID també són els desitjats ja que s'ha modificat el fitxer de configuració login.defs.

De la mateixa manera, si executem l'script de nou:

```
milax@casa:~/Escriptori/GSX_Labs/Prac1/altausers.sh$ sudo ./altausers.sh /users.txt
The user with DNI 39933490-M is already registered.
The user with DNI 39932490 is already registered.
The user with DNI 32222222 is already registered.
The user with DNI 362782282 is already registered.
```

```
The user with DNI 44544 is already registered.
The user with DNI 39933490-M is already registered.
The user with DNI 3993344444 is already registered.
The user with DNI 45 is already registered.
The user with DNI 45 is already registered.
The user with DNI 23456-1 is already registered.
```

Podem obervar que l'script es capaç de detectar aquells usuaris que ja han sigut registrats utilitzant el seu DNI.

Per últim cal afegir que s'ha dut a terme altre proves menors per a comprovar els parámetres d'entrada, permisos, etc.
L'script comprova que té permisos de root, el nombre adequat de paràmetres, comprova que existeixi el fitxer d'usuaris i comprova que existeixin alguns dels directoris necessaris. Si falla algun d'aquests factors l'script acaba o bé en el cas dels directoris els crea per a poder continuar amb l'execució.

Nom:	**canviausers.sh**	Grau d'assoliment	Alt
Permisos:	rwx------	Proprietari:	root
Ubicació dels fitxers:	Grup2F/	Grup:	root

Descripció

Aquest script deshabilitarà o habilitarà el compte dels usuaris que entrem per paràmetre segons estigui habilitat o deshabilitat.

Decisions de Disseny

Comprovem que que es tinguin permisos de root, que s'hagi introduit almenys un argument sinó es llança un missatge per la sortida d'error i acaba l'execució.
A més, comprovem que cada un dels usuaris que es passen per paràmetre existeixi, sinó s'enviarà també un missatge d'error per la sortida d'error i es continuarà tractant la resta.
El codi consisteix en una primera comprovació per veure si l'usuari ja estava deshabilitat o no, això ho fem mirant si el primer símbol de la contrasenya codificada de l'usuari en el /etc/shadow és un ! o no, i després habilitem o deshabilitem l'usuari mitjançant la comanda passwd amb l'argument idoni (-u o -l).

Codi

```
if [ "$1" = "-h" ]; then
    ayuda
    exit 0
fi

# Comprovem que es tinguin permisos de root
if [ $(whoami) != "root" ]
then
  echo "ERROR, Root permissions needed" >&2
  exit 1
fi

#mirem que s'hagin introduit tots els arguments
if [ $# -lt 1 ]
then
    echo "ERROR, you have to indroduce at least one argumnet" >&2
    ayuda
    exit 1
fi

for user in "$@"
do
    #comprovem que l'usuari existeixi
    id $user > /dev/null >&2
    if [ $? -eq 0 ]
    then
       #obtenim el hashcode de l'usuari del fitxer /etc/shadow
       encrypt_pass=$(cat /etc/shadow | egrep $user | cut -d ":" -f2 )
       #comprovem si l'usuari estava bloquejat o no
       if [ ${encrypt_pass:0:1} == "!" ]
       then
               #desbloquegem l'usuari
               passwd -u $user
```

```
                echo "S'ha desbloquejat l'usuari $user"
        else
                #bloquegem l'usuari
                passwd -u $user
                echo "S'ha bloquejat l'usuari $user"
        fi
    else
        echo "ERROR, user $user doesn't exists" >&2
    fi
done
exit 0
```

Joc de Proves

Hem executat l'script en la màquina virtual milax des de l'usuari root passant com a paràmetre a l'usuari milax, després en intentar loguejar-nos com a milax mitjançant su milax i ficar la contrasenya correcta, aquesta no és acceptada.
En tornar a realitzar el mateix procés, ja ens permet loguejar-nos amb normalitat.
També hem comprovat que en el fitxer /etc/shadow s'hagi escrit un ! davant de la contrasenya encriptada.

Nom:	**canviagrup.sh**	Grau d'assoliment	Alt
Permisos:	root	Proprietari:	root
Ubicació dels fitxers:	Grup2F/	Grup:	root

Descripció

Aquest script rep per parametre un possible grup secundari i el posa com a grup actiu.

Decisions de Disseny

Comprovem que que es tinguin permisos de root, que s'hagi introduit almenys un argument sinó es llança un missatge per la sortida d'error i acaba l'execució.
Es guarda el grup principal actual, per a l'hora de sortir de l'execució restablir-lo. A continuació es fa el canvi de grup principal al secundari donat. Es comprova que l'indicat contingui com a secundari el que s'ha passat per paràmetre, sinó s'enviarà també un missatge d'error per la sortida d'error.
Quan es cridi la comanda exit es reverteix al grup principal anterior mitançant l'ús de trap, que permet executar les ordres que s'indiquen quan es surt de l'execució l'script per qualsevol motiu amb exit (les atrapa).

Codi

```
if [ "$1" = "-h" ]; then
    ayuda
    exit 0
fi

# Comprovem que es tinguin permisos de root
if [ $(whoami) != "root" ]
then
  echo "ERROR, Root permissions needed" >&2
  exit 1
fi

#mirem que s'hagin introduit tots els arguments
if [ $# -lt 1 ]
then
    echo "ERROR, you have to indroduce at least one argumnet" >&2
    ayuda
    exit 1
fi

for user in "$@"
do
    #comprovem que l'usuari existeixi
    id $user > /dev/null >&2
    if [ $? -eq 0 ]
    then
```

```
        #obtenim el hashcode de l'usuari del fitxer /etc/shadow
        encrypt_pass=$(cat /etc/shadow | egrep $user | cut -d ":" -f2 )
        #comprovem si l'usuari estava bloquejat o no
        if [ ${encrypt_pass:0:1} == "!" ]
        then
                #desbloquegem l'usuari
                passwd -u $user
                echo "S'ha desbloquejat l'usuari $user"
        else
                #bloquegem l'usuari
                passwd -u $user
                echo "S'ha bloquejat l'usuari $user"
        fi
   else
        echo "ERROR, user $user doesn't exists" >&2
   Fi

done
exit 0
```

Joc de Proves

Hem executat l'script en la màquina virtual milax des de l'usuari root passant com a paràmetre a l'usuari milax, després en intentar loguejar-nos com a milax mitjançant su milax i ficar la contrasenya correcta, aquesta no és acceptada.
En tornar a realitzar el mateix procés, ja ens permet loguejar-nos amb normalitat.
També hem comprovat que en el fitxer /etc/shadow s'hagi escrit un ! davant de la contrasenya encriptada.