### Paràmetres d'entrada

En primer lloc detectarem l'opció d'entrada i farem un control d'errors. Només acceptarem un paràmetre d'entrada que sigui “-p”

```
#!/usr/bin/perl
$numArgs = @ARGV;
$p=0;
$usage=”Usage: BadUsers.pl [-p] \n”;
if ( $numArgs != 0){
    if ( $numArgs == 1 ){
        if ( __$ARGV[0] eq ‘-p’__ ) { # si el primer argument es -p
             $p=1;                 # activem el flag p
         } else { print $usage; exit (1); }
    } else { print $usage; exit (1); }
}
```

### Llegir el fitxer de passwd

Ara farem que l'script obri la base de dades d'usuaris i que guardi totes les dades dels usuaris en una llista.

```
$pass_db_file="/etc/passwd";
open (FILE,$pass_db_file) or die "no es pot obrir el fitxer $pass_db_file: $!";
@password_db= ____<FILE>____; # llegir tot el fitxer de password
close FILE;
```

Què vol dir exactament aquesta sentència?

*open (FILE,$pass_db_file) or die "no es pot obrir el fitxer $pass_db_file: $!";*

Obre a FILE el fitxer amb ruta $pass_db_file

Si no existeix termina (excepcio) i imprimeix error “no es pot obrir”

### Recorregut de la llista d'usuaris

Ara recorrerem la llista d'usuaris i buscarem per a cadascun si tenen fitxers al sistema tot usant la comanda **find** *(*veure **man find***)*. El usuaris sense fitxers els guardarem a un *hash* com invàlids per després utilitzar-los per processar la opció -p.

```
foreach $user_line (@password_db) {
      _chomp $user_line; # eliminar el salt de línia  //borra todos los matches
     @fields = split(':', $user_line);
     $user_id = $fields[0];
     $user_home = $fields[5];
```

```
      if ( -d $user_home ) {
            $comand=sprintf("find %s -type f -user %s | wc -l",
                              $user_home, $user_id);
            $find_out=`$comand`;
            ____chomp $find_out__; # eliminar el salt de línia
      } else {
          $find_out = 0;
      }
      if ($find_out == 0){
            $invalid_users{$user_id} = "invalid";
      }
}
```

Quin és el significat dels paràmetres de la comanda **find**?

*$comand=sprintf("find %s -type f -user %s | wc -l", $user_home, $user_id);*

Formatear un string de forma que quede un comando find en string con parametros del nombre usuario y home

Quina és la diferència entre les funcions **chomp** y **chop**?

Chop borra el ultimo carácter del string
Chomp borra todos los matches del string de la variable $/ (input separator) que por defecto es \n

Quin és el significat de la sentència?

*$invalid_users{$user_id} = "invalid";*

Crear un elemento del hash %invalid_users con clave $user_id y valor invalid

**Usuaris que tenen processos en execució**

Ara afegirem l'opció per no incloure com invàlids al usuaris que tenen processos en execució. Per trobar el usuaris que tenen processos en execució usarem la comanda **ps aux --no-headers** (veure **man ps**) i els eliminarem del *hash* de usuaris invàlids. Els usuaris que queden a la llista com invàlids seran aquells que no tenen cap fitxer ni procés en execució.

Utilitzeu una expressió regular com delimitador dels camps de la sortida de la comanda **ps**. Tingueu en compte que us podeu trobar amb un o més espais buits i amb tabuladors.

```
if ( $p == 1 ){
      @process_list=`ps aux --no-headers`;
      foreach $process_list_line (@process_list) {
            chomp($process_list_line);
            @fields_proc = split(_____________ , “$process_list_line”);
            $user_proc = $fields_proc[0];
            delete($invalid_users{$user_proc});
      }
}
foreach $user_inv_id (sort((keys%invalid_users))){
      print "$user_inv_id\n”;
}
```

Què fa exactament la comanda delete($invalid_users{$user_proc});

Borra el elemento del hash %invalid_users con clave $user_proc

i la comanda: sort( (keys%invalid_users) )

Devuelve las claves ordenadas del hash %invalid_users (keys devuelve una lista con claves = cut

## *El script en Bash*

Ara teniu el mateix script fet en Bash. Completeu els espais buits amb les comandes apropiades.

```
#!/bin/bash
p=0
usage=”Usage: BadUser.sh [-p]”
# detecció de opcions d'entrada: només son vàlids: sense paràmetres i -p
if [ $# -ne 0 ]; then
    if  [ $# -eq 1 ]; then
        if [ $1 == "-p" ]; then
            p=1
        else
            echo $usage; exit 1
        fi
    else
```

```
        echo $usage; exit 1
    fi
fi

# afegiu una comanda per llegir el fitxer de password i només agafar el camp de
# nom de l'usuari
for user in `cat /etc/passwd | cut  -d “:” -f1`; do
    home=`cat /etc/passwd | grep "^$user\>" | cut -d: -f6`
    if [ -d $home ]; then
        num_fich=`find $home -type f -user $user | wc -l`
    else
        num_fich=0
    fi

    if [ $num_fich -eq 0 ] ; then
        if [ $p -eq 1 ]; then

# afegiu una comanda per detectar si l'usuari te processos en execució,
# si no te ningú la variable $user_proc ha de ser 0
            user_proc=`____ps -eo user | grep “$user” | wc -l_`

            if [ $user_proc -eq 0 ]; then
                echo "$user"
            fi
        else
            echo "$user"
        fi
    fi
done
```

Què vol dir exactament aquesta comanda:

*`cat /etc/passwd | grep "^$user\>" | cut -d: -f6`?*

Selecciona el sisè field de les linies que comencen per $user del fitxer /etc/*passwd*

Quina diferencia hi ha amb la comanda:

*`cat /etc/passwd | grep "$user" | cut -d: -f6`?*

Selecciona el 6è field de les linies que contenen user

## *Detecció de usuaris inactius*

Ara esteneu aquest script per detectar usuaris *inactius*. Es defineix com inactius als que no executen cap procés (veure comanda **ps** al punt anterior), que fa molt de temps que no han fet login (ver comandes **finger**, **last** i **lastlog)**, i que fa molt de temps que no han modificat cap dels seus fitxers (veure opcions *time* a la comanda **find**). El període d'inactivitat s'indicarà a través d'un paràmetre:

```
$./BadUsers.pl -t 2d (indica 2 dies)
alvarez
aduran
xavim
marcg

$./BadUsers.pl -t 4m (indica 4 mesos)
xavim
marcg
```

Indiqueu les modificacions necessàries per donar suport a la nova opció d'usuaris inactius