ASO – FIB – LAB EXAM GUIDE

`files` **`Commands between text`**

`commands <parameters>` Files, paths, **command names**

1. Identify system's HW:

```
# lspci && lsusb
```

These commands allow to identify the devices directly connected to PCI and USB buses.

2. map the found hardware parts with the actual devices, plus try to find the devicesnot found using the previous commands. Free command to find RAM memory usage and total (useful for SWAP partition).

```
# dmesg
```

```
# free
```

3. Check if there are already made partitions, and if the sectors are contiguous. See spacing and filesystem:

```
# fdisk -f
```

```
# lsblk
```

PARTITIONS

1. Find whichis the device corresponding to the usb drive, it is in the form:/dev/usb.

* To avoid issues, unmount the USB device.

```
# umount /dev/usb
```

2. Partition the disk using **`fdisk`**.

Determine the geometry and the size of your disk. Create the partitions indicated in Table. If MBR partition table, remember that partitions 1 to 4 are "primary", and that if any of them is "extended", the "logical" partitions inside it are numbered starting at 5. Change the type of the swap partition to "Linux Swap", with the **`t`** command inside **`fdisk`**. Write the contents of the partition table to disk (remember to do so before leaving from the **`fdisk`** command). For the system to be able to boot later, the/partition must have the **bootable flag enabled ("`a`" option in `fdisk`)**.

Device	Type	File system	Size	Mount-point	Comments
/dev/usb1	Primary	ext4	100G	/	At lesat 10Gb
/dev/usb3	Primary	-swap-	2xram		
/dev/usb5	Logical	ext4	20G	/usr/local	
/dev/usb6	Logical	ext4	50G	/home	
/dev/usb4	Primary	free	20GB		Reserved for future use

CREATING THE FILE SYSTEMS

Initialize the file system on those partitions and prepare the swap area for use.

Initialize the swap:

```
# mkswap device
# swapon device
```

Create the filesystems:

```
# mkfs.fstype device
```

MOUNTING THE FILESYSTEMS

1. Mount the filesystems in a temporary directory to be able to install the software.

```
# mkdir /linux
```

2. Mount all filesystems inside such a mount point (`/linux`), creating the appropriate directories inside, at the same time. always inside `/linux`; for instance, `/` partition into `/linux`, `/home` partition into `/linux/home`…

```
# mount part dir
```

* It is important to create the directories after mounting /linux but before mounting the rest.

INSTALLATION OF THE BASE SYSTEM

In our case, we will install from a prepackaged system image that is in the ASO FTP server: asoserver.pc.ac.upc.edu. You will have to decompress it into the USB disk. Cd into the filesystem that will become the root(/) of your new installation (remember, mounted on `/linux`).

```
# cd /linux
# sftp aso@asoserver.pc.ac.upc.edu
sftp> get <filename>
# tar xzf aso-install.tar.gz
```

*remove the tar file! **You can use **lcd** (change directory on local machine) and **cd** (change directory on sftp server).

MOUNTING AUXILIARY SYSTEMS

We have to **bind-mount** the directories `/dev`, `/sys`, and `/proc` inside `/linux`, to temporarily expose the current existing devices to the new system.

```
# mount -o bind /dev /linux/dev
# mount -o bind /sys /linux/sys
# mount -o bind /proc /linux/proc
```

*bind-mounting: A *bind mount* is an alternate view of a directory tree. Classically, mounting creates a view of a storage device as a directory tree. A bind mount instead takes an existing directory tree and replicates it under a different point. The directories and files in the bind mount are the same as the original. Any modification on one side is immediately reflected on the other side, since the two views show the same data.

Use **`mount`** command without any parameters to see which filesystems are mounted, and verify that all USB disk partitions are mounted correctly in the appropriate directories, including the `/dev` directory with the system device files.

SYSTEM CONFIGURATION

Before rebooting, you should configure the system mount points through the `/etc/fstab` file and install a boot loader. configuration files in operating systems based on Unix/Linux are on the default directory `/etc` and almost always in text format.

CONFIGURING THE /ETC/FSTAB

- Add your swap partition: `device none swap defaults 0 0`
- Add root partition: `device / ext4 defaults 0 1`
- Add the rest of filesystems you created previously:

  `device mountpoint fstype defaults 0 2`
- Leave the entry `/proc` and `/sys` without change

CHANGING ROOT DIRECTORY

You can change the root directory of your system, and temporarily use the software that you installed in the system, instead of the one currently available on the system.

```
# chroot /linux
```

CONFIGURING THE KEYBOARD

```
# dpkg-reconfigure locales
# dpkg-reconfigure console-data
# dpkg-reconfigure keyboard-configuration
```

CONFIGURING THE BOOT PROCESS

Currently, we have the system installed, but we need to somehow point where our OS is to the BIOS so that next time you boot the computer it is done properly. To this end we will install the GRUB boot manager.

```
# grub-install /dev/sdb
```

This script prepares the `/boot` directory in order to contain the information needed to boot the machine. The steps it performs:
- Creates the directory `/boot/grub`
- Copy the files needed for GRUB to `/boot`
- Installs the bootloader in the MBR of the USB disk.

The system must inform GRUB which kernel must be used to properly boot. this can be accomplished by editing the file `/boot/grub/grub.cfg`. Debian comes with system tools that allow the automatic creation of such file in order to automatically update the `/boot/grub/grub.cfg` file to contain the UUID of the particular partition you are using. Run the script:

```
# update-grub
```

*get UUIDs with **`blkid`** (note this is just a comment, not necessary for the process).

SETTING UP THE PASSWORDS

To change the password we need to update the file `/etc/shadow`, look at it. As expected the file has hashed passwords rather than plain text. In order to be able to change the passwords we can use the command **`passwd [LOGIN]`**. [LOGIN] is optional, if we're not using the root user we will not be able to use it, if we are we can change other user's password (but never see their password). We should use passwd only if we are not a privileged user, this way we will change OUR password.

---

*You can now exit the **`chroot`** shell

Unmount all filesystems and reboot using the **`shutdown -r now`** command.

POST-CONFIGURATION

CONFIGURING FILESYSTEMS

In file systems ext3 and ext4 we find certain properties that can be changed after formatting, with the **`tune2fs`** command. To change the frequency of checks of the filesystem in the usb1 partition to 28 days:

```
# tune2fs /dev/sdb1 -i 28d
```

CONFIGURING SYSTEM LOGIN MESSAGES

There are several configuration files that handle messages that appear during the process of login into the system. We want to change some of these messages. Usually the configuration files are located in `/etc`.

Before the login prompt ”asoclient login:”, it appears a message similar to ”Debian GNU/Linux 7”.Often, we want to change this message. Now we want to change it. This file is the one called “issue” located at `/etc/issue`. Another file we might want to change is the “motd” “Message-Of-The-Day”, this is usually used to give users lats-minute information about the system status or news. Locate it at `/etc/motd`.

SETTING UP NETWORK

First, we will dothe network configuration by hand and then we will use DHCP to configure it permanently. Before starting, and to avoid mistakes we will flush the current network status running:

```
# ip link show
# ip link set dev <thernetiface> down
```

Manual configuration

1. Configure the network interface using **`ip address`** command.

2. Configure the routing table using **`ip route`** command.

3. Setup name resolution in `/etc/resolv.conf`

- Bring interface up:

```
# ip link set dev <eth iface> up
```

- Add default getway to routing table

```
# ip route add default via <gateway> dev <eth iface>
```

Permanent configuration

We want the network to be configured at bootime. Delete previows config from “Manual Configuration”.

```
# ip link set dev <eth iface> down
# ip address del ??.??.??.??/24 dev <eth if>
```

Debian saves several network config files on `/etc/network`. Now we will use systemctl list-units so we can see all available services and status. You can see the networking.service. In the directory `/etc/network` there is a file `interfaces` that is where you configure different interfaces. Right now there is only configured the loopback interface. Add an entry in the `/etc/network/interfaces` file to configure your network interface.

```
# FOR MANUAL CONFIG
auto <eth if>
iface <eth if> inet static
      address xxx.x.x.x
      network xx.xx.xx.x
      netmask 255.255.255.0
      gateway xxx.x.x.x
```

```
# DHCP CONFIG:
auto <eth if>
# allow-hotplug eth0
iface <eth if> inet dhcp
```

```
#NETWORK CONFIG COMMANDS
ifconfig <ifce> <GW> netmask <Mask>
route add default gw <GW>

<edit /etc/resolv.conf>
# Bring ifce up
ip link set dev <eth iface> up
ifup <ifce> // ifdown <ifce>
```

GESTIÓ D'USUARIS

Bases de dades del sistema:

`/etc/passwd:` Base de dades de usuaris. Totes les dades excepte el password. Estructura:
`username:password:UID:GID:UserInfo:HomeDir:Shell`

`/etc/group:` Estructura → ***`groupname:password:GID:GroupUserList`*** → *Separated by comma, password not used generally.*

`/etc/shadow:` Conté el password encriptat. Estructura → ***`usrname:passwrd:lastchanged:min:max:warn:inactive:expire`***

- Min: number of days required between password changes
- Max: maximum number of days the password is valid. After that, user is forced to change.
- Warn: number of days before password is to expire that user is warned that his/her password must be changed.
- Inactive: The number of days after password expires that account is disabled.
- Expire: days since Jan 1, 1970 that account is disabled: when the login may no longer be used.

`/etc/aliases:` Mail aliases.

Comandes útils: ***`useradd | userdel | passwd | newusers | vipw | groupadd | groupdel | gpasswd | newgrp | vigr | chmod | chown | chgrp`***

Els comptes d'usuari permeten al sistema diferenciar les dades i processos de cada usuari I permeten als usuaris protegir la seva informació. Al kernel els usuaris s'identifiquen amb un nombre enter conegut com l'identificador d'usuari.

Creació d'un nou usuari: assignació d'un UID i modificació de la base de dades d'usuaris per assignar els paràmetres propis de l'usuari, associar almenys un grup a l'usuari i copiar els fitxers de configuració i personalització al directori home de cada usuari.

Afegir usuari a sudoers:

1. Editar arxiu `/etc/sudoers` amb **`visudo`** o directament editant-lo:
   `[%grup / user] HOSTS:(TARGET USERS) [NOPASSWD]:ALLOWED_COMMANDS`
   `usuari ALL:(ALL) ALL` → Això dona permís d'executar qualsevol comanda en qualsevol host I com a qualsevol usuari.

2. **`sudo adduser <username> sudo`** : això afegeix l'usuari al grup «sudo».

PROFILE I ENTORN D'USUARI

En el cas de **bash e**l fitxer `/etc/profile` permet a l'administrador del sistema definir un entorn comú per a tots els usuaris definint la variable PATH. Per altra banda `.bash_profile` permet a cada usuari definir el seu propi entorn adequant el PATH, prompt, etc.

Quan es crea el directori home d'un usuari s'han de copiar els fitxers del directori `/etc/skel`. L'administrador del sistema pot posar fitxers a `/etc/skel` que donin un entorn inicial pels usuaris. Aquest directori és el directori esquelet per a la creació del home de nous usuaris.

Comprovar que al PATH de tots els usuaris hi sigui el directori `/usr/local/bin` I feu que el `.bash_profile` modifiqui el PATH per incloure un directori bin situat enel directori home de cada usuari (`$HOME/bin`):

```
# PATH = PATH:HOME/bin
```

Per a **canviar el prompt** s'utilitza la variable d'entorn **`$PS1`** on \u = username, \w = working dir, \h = hostname.

CREACIÓ MANUAL D'USUARIS

1. Selecció de paràmetres:

- UID
- Username
- Home dir
- Shell
- Groups

2. Editar la base de dades d'usuaris per afegir-los. Per a fer-ho utilitzem **`vipw`** aquesta comanda ens fa un lock del fitxer passwd, per tal de que només el poguem editar nosaltres com a root user. Així no hi ha conflicte amb un possible usuari canviant el seu password al mateix moment:

```
test:x:1001:1001:Test user aso:/home/test:/usr/bin/zsh
```

3. Editar base de dades de grups de la mateixa manera, utilitzant **`vigr`** .

```
test:x:1001:test
```

4. Desactivem el compte de l'usuari, fins acabar el procés d'alta, canviant el seu shell en l'arxiu **`passwd`** a **`...:/bin/nologin`**. Una altra opció que tenim és invalidar el password, això ho podem fer substituïnt el password (x) de l'usuari a l'arxiu `/etc/passwd` per un **`!`** o utilitzant la comanda **`passwd -l <username>`**

5. Crear el directori home de cada usuari i modificar-ne els permisos i propietaris.

```
# mkdir /home/usuari
# cp /etc/skel /home/usuari
# chmod 700 -R /home/usuari
# chown test:test /home/usuari
```

6. Assignar un password als nous usuaris. Per fer això s'ha de modificar l'arxiu `/etc/shadow`. Això només ho pot fer el super usuari, i es fa través de la comanda: **`# passwd <username>`** altres comandes per editar el fitxer shadow son:

`chage:` change user password expiry information
`pwck:` verify integrity of password files

`chfn:` change your finger information
`chsh:` change your login shell

CREACIÓ AUTOMÀTICA D'USUARIS

Tot el procés anterior es pot fer de manera més senzilla a través de comandes com **`useradd`** i **`adduser.`** Son el mateix, però adduser et guia a través de preguntes a la mateixa cli.

*Els valors per defecte de **`useradd`** estàn en l'arxiu de configuració que es troba a `/etc/default/useradd.` Aquests valors per defecte també es poden canviar a través de flags en la mateixa comanda:

```
useradd [-d <Home_Dir> -c <UserInfo> -e <ExpireDate> -g <GID> -p <password> -s <Shell> -u <UID>] <uname>
```

Per aconseguir diferents nivells d'accés sense modificar els permisos dels directoris de home cada usuari, el millor és modificar les carpetes compartides de cada usuari.

*És també important conéixer la comanda **`visudo`** per editar l'arxiu `/etc/sudoers`

ELIMINACIÓ I DESACTIVACIÓ D'USUARIS

Per a donar de baixa un usuari s'han d'eliminar tots els seus fitxers, busties, impressió, cron, at I referències a l'usuari. Llavors s'esborren les línies associades al fitxer `passwd` i `groups`. Pot ser que l'usuari tingui fitxers fora del directori home.

Buscar per tots els directoris fitxers de l'usuari i eliminar-los:

```
# find / -user <user> -exec rm -fR {}\
```

Backup de tots els fitxers de l'usuari amb **`xargs`**.

```
# find / -user <user> | xargs tar -czf
```

Si volem que **`xargs`** no tingui problemes amb els noms que contenen espais, podem utilitzar el flag **`-d '\n'`**. Això marca a **`xargs`** que el delimitador entre arxius es '\n' i no '\'

*Què es **xargs?** És una comanda utilitzada per a crear i exectuar comandes a través de l'entrada standard (stdin). Converteix l'input del stdin en arguments per la comanda següent.

És recomenable desactivar el compte abans de començar la baixa. A part de les maneres de desactivar el compte ja comentades abans, també podem utilitzar un **tail script**. I canviar el *shell* de l'usuari per aquest tail, que mostrarà un missatge amb informació sobre perquè s'està desactivant el seu compte. Podem canviar el shell de l'usuari amb la comanda **chsh.** L'script pot ser guardat a un directori com `/usr/local/lib/no-login.` L'script podria ser:

```
#!/usr/bin/tail -n 2
This account has been closed due to a security problem. Please
contact the system administrator.
```

Exemple d'script que donat un username fa un backup del seu directori home, esborra tots els fitxers que l'usuari té al sistema i canvia el shell per un tail script.

```
#!/bin/bash
p=0
usage="Usage: backndel.sh [username]"
# detecció opcións entrada.
if [ $# -eq 0 ]; then
     echo $usage; exit 1
  fi

user=$1
filename="backup$user.tar.gz"

# Backup. El "-T" equival a "--files-from=FILE", el "-" al final indicia
# que ho volem agafar del stdin.
copy=`find / -user $user | tar -czf /home/copies/$filename -T - `
delete= 'find / -user $user -exec rm -fR {} \; '
chshell=`chsh -s /usr/local/lib/no-login.sh $user`
```

Per a desactivar l'usuari root o per eliminar-ne el password, podem utilitzar les comandes:

`# passwd -l root`

`# passwd -d root`

Per a reactivar-lo:

`# sudo passwd root`

BACKUPS

Hard link: Los enlaces duros, asocian dos o más ficheros compartiendo el mismo inodo. Esto hace que cada enlace duro sea una copia exacta del resto de ficheros asociados, tanto de datos como de permisos, propietario. Al modificar los datos apuntados por cualquiera de ellos, se cambian los datos reales almacenados en disco, quedando modificados para todos por igual. El proceso de eliminación de un enlace, desvincula un nombre de los datos reales. Los datos todavía estarán accesibles mientras quede algún enlace. Cuando se elimina el último enlace duro, el espacio que ocupaban los datos se considera disponible.

Soft link: Indica un acceso a un directorio o fichero que se encuentra en un lugar distinto dentro de la estructura de directorios. Una modificación realizada utilizando este enlace se reflejará en el original; pero, por el contrario, si se elimina el enlace, no se eliminará el auténtico.

Per a fer aquestes còpies utilitzarem **`tar`** i **`rsync`.**

PARTICIÓ PER EMMAGATZEMAR LES CÒPIES DESEGURETAT

1. Crear (com a usuari root) un nou directori `/backup` on hi muntarem l'espai de disc.
2. Crear una nova partició a l'espai lliure del disc (ext3 o ext4).

    a. Si ja havies creat una partició amb espai buit:

    ```
    # mkfs.ext4 /dev/usb4
    ```

    b. Si havies deixat l'espai en blanc (sense partició):

    ```
    # fdisk /dev/usb
    Fdisk> n \ p \ 4 \ \ +20G
    Fdisk> t \ 4 \ 83
    Fdisk> w
    ```

3. Muntar la nova partició: **`mount /dev/sdb4 /backup`**

4. Concedir només accés al directori a l'usuari root:

    ```
    # chmod go-rx /backup
    ```

    + Seguretat:

        Muntar en mode Read-Only sempre i només muntar-lo per escriptura quan anem a fer backups:

        ```
        # mount -o remount,[ro / rw] /dev/usb4 /backup
        ```

5. Modificar "/etc/fstab" per tal de que aquesta partció es monti sola durant el boot.

    ```
    # vim /etc/fstab
    Vim fstab>>
        /dev/sdb4   /backup   ext4   ro   0   2
    ```

CÒPIES AMB TAR

CÒPIES COMPLETES

Còpia completa del directori `/etc`: **`tar -cf /backup/backup-etc-nivell0-`date +%y%m%d%h%s`.tar /etc`**
La compressió no es recomanable ja que augmenta el temps de backup. Per això no posem la opció "z" en la comanda tar. Si volem comprimir-lo, hem de activar l'opció "z".

Per tal d'excloure fitxers d'una còpia completa, podem crear un fitxer amb una llista de fitxers que es volen excloure: **`tar -cf /backup/backup-etc-nivell0-`date +%y%m%d%h%s`.tar -x <excludes> /etc`**

Un mecanisme de signatura digital, com son els hashs MD5, permeten verificar la integritat d'un fitxer I comprovar-la:

```
# md5sum <file> > <file>.asc
# md5sum -c <file>.asc <file>
```

CÒPIES INCREMENTALS

Les còpies incrementals només fan backup dels fitxers canviats. Per tal de poder experimentar, s'han de fer canvis a alguns fitxers. **`touch`** ens permet modificar la data de modificació d'alguns fitxers.

`tar` té la opció «`--newer`» que només inclourà els fitxers modificats des d'una data determinada. La data la podem especificar manualment **`tar --newer='2007-11-28 12:10'`** o podem agafar una data d'un fitxer: **`tar -newer='<file>'`** . *Recordar que és possible que el directori de backup estigui en Read-Only.

RESTAURACIÓ DE CÒPIES

Primer es restaura la còpia total, llavors per ordre les incrementals. S'extreu l'arxiu de backup en el directori on ha d'anar.

RESTAURACIÓ DE FRAGMENTS

`tar` ens permet extreure fitxers o rutes concretes. Per a fer-ho:

```
# tar xvf file.tar <filetoextract>
# tar xvf file.tar [path/to/dir || path/to/file}
```

* Pots utilitzar el flag -C per indicar a continuació el directori on vols que sigui descomprimit l'arxiu: **`tar...-C /target/dir`**

RSYNC

`rsync` permet copiar un directori (o un conjunt de fitxers) a un altre directori a través de una connexió de xarxa. Usa un algorisme de checksum eficient per transmetre només les diferències entre els dos directoris i al mateix temps comprimeix els fitxers per fer més ràpida la transmissió de dades. **`rsync`** permet copiar enllaços, dispositius, i preservar permisos,grups i propietaris. També suporta llistats de exclusió i connexió remota ambshell segur (ssh) entre altres possibilitats.

BACKUPS A TRAVÉS DE XARXA

`rsync -avz /root -e ssh root@localhost:/backup/backup-rsync/`

*és necessari activar el compte del root i posar-li una contrasenya vàlida & tenir ssh instalat.

Si ara creem un nou arxiu al directori root, tornem a fer la comanda **`rsync`** d'abans I esborrem el fitxer nou veurem que el fitxer encara existeix en el directori de backup.

CÒPIES INCREMENTALS INVERSES

El directori en què tenim el mirror queda exactament igual que el directori d'origen. Això és un problema perquè no tenim control dels canvis realitzats. Per solucionar aquest problema podem utilitzar l'opció «`--backup`» i «`--backup-dir`». Els backups generats amb les opcions es diuen inversos perquè la còpia completa és la més recent i no la mésantiga com amb **`tar`**. Amb aquesta opció la còpia completa correspon a laúltima data en que s'ha fet el backup i les incrementals a les dels diesanteriors.

```
#!/bin/bash
SOURCE_DIR=
DEST_DIR=
# excludes file: list of files to exclude
EXCLUDES=
# the name of the backup machine
BSERVER=
# the name of the incremental backups directory
# put a date command for: year month day hour minute second
BACKUP_DATE=
# options for rsync
OPTS="--ignore-errors --delete-excluded --exclude-from=$EXCLUDES \ --delete --backup
--backup-dir=$DEST_DIR/$BACKUP_DATE -av"
# now the actual transfer
rsync $OPTS $SOURCE_DIR root@$BSERVER:$DEST_DIR/complet
```

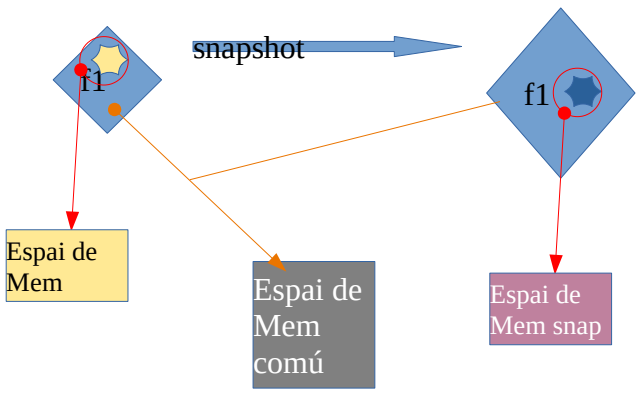
SNAPSHOTS

Amb els enllaços durs, és possible fer que les còpies incrementals semblin còpies completes. El nom d'un fitxer no representa el fitxer mateix, per al sistema és només un enllaç dur al inode. Un fitxer (inode) pot tenir més d'un enllaç dur. Si un inode té dos links (fitxers), un canvi en un d'ells també provoca un canvi en l'altre (dates de modificació, permisos, etc.), però si el borrem llavors el fitxer tan sols perd un link I no s'esborra. **`cp -l`** ens permet crear un nou hard link en comptes de copiar l'arxiu. **`-a`** que fa una còpia recursiva i preserva els permisos de accés, els temps i els propietaris delsfitxers.

Poden combinar **`rsync`** i **`cp -al`** per crear backups que semblin múltiples còpies completes d'un sistema de fitxers sense que sigui necessari gastar tot l'espai en disc requerit per totes les còpies:

```
rm -rf backup.3
mv backup.2 backup.3
mv backup.1 backup.2
cp -al backup.0 backup.1
rsync -a --delete source_directory/  backup.0/
```

S'elimina el backup més antic, I llavors els altres es desplacen una posició (2 → 3). Es copia el backup0 com a backup1 en forma de hard-link. Això el que fa és que ens permet tenir una "instantània" del backup total del sistema en backup1. La última comanda sincronitzarà els arxius nous i modificats amb el nostre backup0. El backup0 és la còpia més recent del nostre directori font, la qual serà la que sempre es sincronitzarà amb aquest. Backup1 és un nou hard link al fitxer backup0.



Un snapshot bàsicament és la creació d'un nou fitxer que apunta al mateix inode que l'altre. Aquests comparteixen el mateix espai de memòria i ho tenen tot en comú en un principi. Quan un dels dos pateix un **canvi** aquest s'apunta en un nou espai de memòria (només el canvi). De manera que el que tenen en comú sempre estarà en un espai compartit, i els canvis s'anotaràn en espais diferents.

INSTAL·LACIÓ D'APPS MANUAL

Si es un arxiu .deb:

```
sudo dpkg -i program.deb
```

A partir de codi font (**.tar.bz2 normalment**):

1. Descompirmir en directori propi.
2. Llegir docu.
3. Instalar dependències (normalment a la docu i amb **apt**).
4. Localització de llibreries i fitxers:

```
autotools:./configure
 cmake:cmake -DCMAKE_INSTALL_PREFIX=..
```

5. Compilar

```
make
```

6. Instal·lar

```
make install
```

SCRIPTS

1. **Usuaris innecessaris.** Es demana fer un script que determini quins usuaris del fitxer /etc/passwordsón invàlids. Un usuari invàlid és aquell que existeix en el fitxer de passwd peròque en canvi no té cap presencia en el sistema (és a dir, que no té cap fitxer).També, hi ha usuaris que no tenen cap fitxer, però que serveixen per executardaemons del sistema. Afegiu una opció per declarar vàlids als usuaris quetenen algun procés en execució (flag -p).

```
#!/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 -u $user | wc -l`

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

2. **ocupacio.sh**. S'ha de fer un script que calculi l'espai en disc utilitzat per cada usuari del sistema. Si sobrepassa un espai determinat que es passa com paràmetre, s'haurà d'escriure un missatge al .profile del usuari en qüestió per informar-li que ha d'esborrar/comprimir alguns dels seus fitxers.

```
#!/bin/bash
usage="Usage: ocupacio.sh "

if [ $# -ne 0 ]; then
    if [ $# -eq 1 ]; then
        t=$1
    else
        echo $usage; exit 1
    fi
else
    echo $usage; exit 1
fi

p=`echo $t | cut -c ${#t}`
t=`echo ${t::-1}`

if [ "$p" = "M" ]; then
   t=$((t*1024))
fi


for user in `cat /etc/passwd | cut -d: -f1`; do
    home=`cat /etc/passwd | grep "^$user\>" | cut -d: -f6`

    if [ `echo $home | cut -d/ -f2` = "home" ]; then
       if [ `echo $home | cut -d/ -f3` = "festival" ]; then
            echo "festival"
       else
        u=`du -sh $home | cut  -f1`
        echo $user $u

            p=`echo $u | cut -c ${#u}`
            u=`echo ${u::-1}`

            if [ "$p" = "M" ]; then
                u=$((u*1024))
            fi


            if [ $u -ge $t ]; then
                if [ `cat $home/.profile | grep "You use too much space" | wc -l` -eq 0 ]; then
                    echo "echo \"You use too much space!\"" >> $home/.profile
                    echo "> .profile updated"
                fi
            fi
        fi
    fi

done
```

3. **Informació d'usuaris.** Volem fer un script que donat un nom d'usuari ens doni la següent informaciórelacionada amb ell: Home, Tamany total del directori home (tot incloent subdirectoris), Directoris fora del directori home on l'usuari te fitxers propis, Nombre de processos actius de l'usuari

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

# afegiu una comanda per llegir el fitxer de password i només agafar el camp de # nom de l'usuari
exists=`cat /etc/passwd | grep "^$p\>" | wc -l`

if [ $exists -eq 0 ]; then
    echo "Invalid username"
else
    home=`cat /etc/passwd | grep "^$p\>" | cut -d: -f6`
    echo "Home: $home"
    echo "Home size: `du -sh $home | cut  -f1`"
    echo "Other dirs: "
#    for archivo in `find / -type f -user $p`; do
 #       if [ "`echo $archivo | cut -d/ -f2`" != "proc" ]; then
  #          if [ "`echo $archivo | cut -d/ -f2`" != "home" ]; then
   #             if [ "`echo $archivo | cut -d/ -f3`" != "$p" ]; then
    #                dirname $archivo
     #           fi
      #      fi
       # fi
    #done | sort | uniq
    echo "Active processes: `ps -u $p | wc -l`"

fi
```

4. netout.sh

```
#!/bin/bash

usage="Usage: net-out.sh [time]"
if [ $# -ne 0 ]; then
    if [ $# -eq 1 ]; then
        t=$1
    else
        echo $usage; exit 1
    fi
else
    t=5
fi

while [ 1 ]; do
    for int in `netstat -i | cut -d\  -f1`; do
        if [ $int != "Kernel" ]; then
            if [ $int != "Iface" ]; then
                echo "$int: `cat /sys/class/net/$int/statistics/tx_packets`"
            fi
        fi


    done
    sleep $t
done
```

5. inactiveuser.sh

```
#!/bin/bash
t=0
usage="Usage: InactiveUser.sh -t X{d/m}"
# detecció de opcions d'entrada: només son vàlids: sense paràmetres i -p
if [ $# -ne 0 ]; then
    if [ $# -eq 2 ]; then
        if [ $1 == "-t" ]; then
            t=$2
        else
            echo $usage; exit 1
        fi
    else
        echo $usage; exit 1
    fi
else
    echo $usage; exit 1
fi

p=`echo $t | cut -c ${#t}`
t=`echo ${t::-1}`

if [ "$p" = "m" ]; then
   t=$((t*30))
fi

first=0
for user in `lastlog -b $t | cut -f1 -d\ `; do
    if [ $first -eq 0 ]; then
        first=1
    else
        home=`cat /etc/passwd | grep "^$user\>" | cut -d: -f6`
        if [ -d $home ]; then
            mods=`find $home -user $user -mtime $t`
            num_fich=`find $home -type f -user $user`


        fi
    fi
done
```

6. Delete user

```
#!/bin/bash
usage="Usage: DeleteUser.sh user"

if [ $# -eq 1 ]; then
        user=$1
else
    echo $usage; exit 1
fi

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

find $home -user $user -type f > /tmp/backuperfile.txt
tar -cvf /tmp/backup_$user -T /tmp/backuperfile.txt
rm /tmp/backuperfile.txt

find $home -user $user -print0 | xargs -0 rm -rf

chsh -s /usr/local/lib/no-login $user
```

guide author: Alvaro Moreno - 2019