Examen Laboratori ASO

0. ASO SFTP

1. `# cd /The/Directory/Where/You/Want/To/Download/Somthing`
2. `# sftp aso@asoserver.pc.ac.upc.edu`
    - Password: **AsORoCkSHaRd!**
3. `# get /what/you/want/to/download`
4. `# exit`
5. `tar xzf package-downloaded.tar.gz`
6. `rm clean-the-tar-pls.tar.gz`

1. Installation of the OS

1.1 Hardware Identification

1.1.1 Seen in class

`# lspci`

> Package name: **pciutils**
>
> **-m:** This option "simplifies" and makes the output a little more human

`# lsusb`

> *Not very useful...*
>
> Package name: **usbutils**
>
> **-D:** Lets you specify the device you want info of

`# dmesg`

> That's a lot of info there.... I mean, don't use it without a grep or similar
>
> Package name: **util-linux**
>
> **-H:** This is useful? Well... Just avoid using this comand if you can

1.1.2 My recomendation

```
# hwinfo
```

> The most human friendly and very complete scan
>
> Package name: **hwinfo**
>
> **--short:** Don't use it without this option pls XD, it just simplifies it

1.2 Disk Partition

Before start this section is **strongly recomended** using:

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

1.2.1 Seen in class

```
# gdisk
```

> Package name: **gdisk**
>
> Great partition tool :D
>
> **Micro Tutorial:**
>
> 1. Execute gdisk as:
>
> `# gdisk /dev/sdX`
>
> *Where sdX is the name of the disk you want to partition*
>
> 2. Select the partition table, normally gpt
> 3. ? for info, Normally you will only use:
>
> - **p:** Prints the partitions info of sdX
> - **l:** List the known partition types (Useful for filesystem creation)
> - **n:** Create a new partiton specifing it's start point and end point
> - **w:** Writes all the changes and exits
> - **q:** Useful when you fuck it up (Exits without saving)

1.2.2 My recomendation (only if gdisk does not work)

`# fdisk`

> The good old fdisk never fails :D
>
> Very similar to gdisk but with less options

1.3 Filesystem creation

Example "minimum" filesystem:

Device	Code	Size	Mount-point	Comments
/dev/sdb1	EFI (EF00)	~512MB	/boot/efi	**Formated in vfat**
/dev/sdb2	Linux (8304)	~30GB+	/	Main partition **the first one to mount!!**
/dev/sdb3	Linux (8300)	~5GB	/usr/local	*Optional, not that common*
/dev/sdb4	Linux (8302)	~100GB+	/home	*Optional* and can be more than one
/dev/sdb5	Swap (8200)	~2x'RAM'		Check 'RAM' with `# free`, normally 8GB or 16GB

1.3.1 Commands

`# mkswap /dev/sdb5`

> Change /dev/sdb5 with the device name of your swap partition

`# mkfs -t fstype device`

> Change fstype with the type used, normally: ext4 or vfat

1.4 Mounting

`# mount partition directory`

> Without options it displays the mounted filesystems
>
> You will get errors if:
>
> - The partition is already mounted
>   - Sol: `# umount partition`
> - The directory does not exist
>   - Sol: `# mkdir directory`
> - A sub-partition is already mounted:
>   - Ex: If you want to mount `/boot/efi` you have to have `/` mounted

1.4.1 Mounting during boot

To mount during boot you have to modify the `/etc/fstab` fiel

It always follows a fixed format:

Device	Mountpoint	fstype	Options	Dump	pass num
/dev/sdb5	none	swap	defaults	0	0
/dev/sdb2	/	ext4	defaults	0	1
/dev/sdbX	mount/point	fstype	defaults	0	2

> **Always follow the order in which you want them to be mounted**
>
> **Device:** The device name
>
> **Mountpoint:** The mount-point
>
> **fstype:** Normally {swap, ext4, vfat}
>
> **Options:** The options when mounting, most common:
>
> - **defaults**: Its the same as: rw, suid, dev, exec, auto, nouser, async
> - **ro**: mounts it as read only
> - **wo**: mounts it as write only
> - **auto**: it will be mounted at boot or with mount -a
> - **noauto**: It won't be mounted at boot or with mount -a
> - **user**: Permits any user to mount he filesystem
> - **nouser**: Only root can mount the filesystem
> - **exec/noexec**: Oermit/Prevent the execution of binaries from the fs
> - **suid/nosuid**: Permit/Block the operation of suid and sgid bits
>
> **Dump:** If 1 the *dump* backup utility will back it up, if 0 it wont
>
> **pass num**: Always put 0 swap, 1 to root partition, and 2 for the rest

1.5 Configurations and utils

1.5.1 Changing root directory

```
# chroot /'directory'
```

> It changes the `/` to `/'directory'`

1.5.2 Keyboard configuration

Best Way

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

Fast Way

`# loadkeys es`

1.5.2 Grub

You MUST be chrooted into the mounted device, for example /linux

`# grub-install --target=x86_64-efi /dev/usb`

`# update-grub`

1.5.3 Passwords

`# passwd user`

> user is the name of the user that you want to change the password

1.5.4 System login messages

`/etc/issue`

> Normaly contains the distro name, Ex: Debian GNU, Arch linux..

`/etc/motd`

> Is the first message printed once logged in

1.5.5 Network Configuration

We flush de current network configuration

```
# ip link show // We show all the interfaces and pick the right one
# ip link set dev <ethernet IF> down
# ip link set dev <ethernet IF> up
```

Permanent Configuration

1. Open the `/etc/network/interfaces` with an editor, and add:
2. `auto <ethernet IF>`

3. `iface <ethernet IF> inet static`
4. Input the following parameters, changing them as necessary:
    - `address 10.10.41.???`
    - `network 10.10.41.0`
    - `netmask 255.255.255.0`
    - `gateway 10.10.41.1`
5. You should reboot... In order to skip this you can run:
    - `# ifup <ethernet IF>`
    - `# ifdown <ethernet IF>`
6. You can use dhcp with
    - `# echo "iface <ethernet IF> inet dhcp" >> /etc/network/interfaces`
7. Put up the net
    - `#ifup <ethernet IF>`

1.5.6 Sudo Configuration

```
# apt update
# apt install sudo
# usermod -a -G sudo <user> // We put user in sudo group
```

2. Application Management

2.1 General Commands

2.1.1 Tar

tar.gz / tar.bz2 / tar.xz

``-z / -j / -J`

Listing contents tar

`$ tar -tvf file.tar`

Searching Files

`$ tar -tvf file.tar.bz2 'hola.txt'`

Extracting files

`-x`

2.1.2 Links

Hard link: Creates an complete copy of a file in another location

- `$ ln sourceFile linkFile`

Soft link: Creates a "shortcut" to a file in another location

- `$ ln -s source link`

Verify:

- `ls -l source link`

2.2 Manual Installation

Install

`$ dpkg -i <file.deb>`

Uninstall

`$ dpkg -r <package>`

Purge

`$ dpkg -P <package>`

2.3 Package manager

2.3.1 Configure Software Repositories

1. Open `/etc/apt` and modify it with:
2. `deb http://ftp.es.debian.org/debian/ stable main non-free contrib`
3. Run: `# apt update`

> To search info about a package you can use: `$ apt info <package-name>`

2.3.2 Installing a desktop

1. `sudo apt install x-window-system`
2. ``sudo apt list --all-versions | grep ^task-`
    - We can also use `sudo apt search <package>`
3. Select the desired desktop

> If tou have problems with the configuration of a package you can use
>
> `# dpkg-reconfigure <package-name>`

2.4 Building the package

1. Uncompress the .tar.gz
2. Read README
3. Run configure
    - If does not work is probably because of dependencies, install them and re-run
4. `$ make`
5. `# make install``

3. Scripts

`#!/bin/bash` : at the begining of the script

`.sh` : Extension of the script

`# chmod +x script.sh` : To give execution permises

`$#` : Number of parameters introuced

`$1, $2...` : First parameter, Second parameter

Switch syntax

```
case first_case in
    "first_possibility")
        # code if first_case == "first_possiblity"
        ;;
    "second_possibility")
        # code if first_case == "second_possiblity"
        ;;
    *)
        # code if default
        ;;
esac
```

Common Comparison Operators

```
$var -eq 0  # var is equal to 0
$var -ne 0  # var is not equal to 0
$var -gt 0  # var is greater than 0
$var -ge 0  # var is greater or equal to 0
$var -lt 0  # var is less than 0
$var -le 0  # var is less or equal to 0
```

While syntax

```
while [ condition ]; do
    # code
done
```

If / elif / else

```
if [ condition1 ]; then
    # code
elif [ condition2 ]; then
    # code
else
    # code
fi
```

For

```
for item in $item_list; do
    # code
    # You can acces to the "current" item with:
    $item
done
```

Variable declaration

```
caca = somthing
# Now caca is a variable, you can acces it with
$caca
```

Using linux commands

```
res = $(cat .. | grep .. | cut .. | sort | uniq)
for item in $(chunky_command); do
done
```

Writing in console

```
name = "My name"
echo "I wanna write this :D"
echo "Call me $name" # With this we can write variables
```

4. User Management

4.1 PATH

Temporary

`$ export PATH=$PATH:/path/to/add`

Permanent

`$ echo "export PATH=$PATH:/path/to/add" >> ~/.bash_profile`

4.2 User creation

4.2.1 By hand

`# vipw`

> Allows safe edition of the passwd file, there you can change UID's, usernames, home drectories, shells...
>
> **-s:** Allows modifing the shadow file

`# vigr`

> Allows safe edition of the group file, there you can add users to groups and create groups

Home config

1. `# cp /etc/skel/* /home/of/the/user/`
2. `# chown -R user /home/of/the/user/`
3. `# chgrp -R userGroup /home/of/the/user`
4. `# chmod -R chmodbits /home/of/the/user`

> **First number**:
>
> - 0: Nothing
> - 2: Sgid
> - 4: Suid
>
> **Second number**:
>
> - Owner permissions, rwx
>
> **Third number**:
>
> - Group permissions, rwx
>
> **Fourth number**:
>
> - Others permissions, rwx

4.2.2 Automatic

`# useradd`

> -**d**: Home dir
>
> -**g**: Main group
>
> -**G**: Other groups
>
> - GROUP1[,GROUP2[,GROUPN]
>
> -**r**: Creates a system account
>
> -**s**: Name of the shell
>
> -**u**: Numerical value of the user's ID
>
> -**U**: Create a group with the same name as the user

4.3 Removing and disabling users

Before removing files of a user, first create a backup

Ways of disabiling a user:

1. `# usermod -L user`
   - This command puts an '!' in the corresponding line in the passwd file
2. `# chage -E0 user`
   - This command will expire the account with name user, is better.
3. `# usermod -s /sbin/nologin user`
   - This command will change the shell to a "useless" one
4. To rest at ease the best method is doing the 3 methods :D

5. Backups

5.1 Backup creation

Full Backup

```
tar -cvpf "name_of_file-$(date '+%Y...').tar" /root/
```

> If we add:
>
> - **-j**: The backup will be compressed with .tar.bz2
> - **-z**: The backup will be compressed with .tar.gz
> - **-J**: The backup will be compressed with .tar.xz

Incremental Backups

`tar -cvpf --newer ="date"`

`tar -cvpf --newer =./file`

> it will add to the tar only the files that are newer than date/file

Checking Backup Integrity

1. `$ sha512sum file`

> Creates a sha checksum and adds it into the file

2. `$ sha512sum -c file`

> Checks if the checksum is the same of the checksum of the file.

Restoring a Backup

We always start restoring in inverse order of creation

So we always start with level 0, then 1...

`$ tar -xvf file.tar -g /dev/null`

> If its compressed we have to add the apropiate flags already seen before
>
> **-C:** to restore to an especific location

6. Task Scheduling

6.1 One-Time

The command will be executed with the same privilegies of at, so **it may need sudo**

`$ at almost_any_date/time_format`

`at > comand_to_execute`

`at > <EOT>` # For this you have to press Ctrl + D on a "new" line

> at may not be installed... It need to have "atd" running
>
> Package: at

6.2 Periodically

It is necessary to modify the **crontab** archive.

The structure of this archive is the following:

```
# min hour day month weekday user comand_to_be_executed
* * * * Sunday aso /home/aso/script.sh # Will be executed every sunday
10 19 3 12 * aso /bin/local/script.sh # Executed every 3/12 at 19:10
...
```