

# Lab 1

## Installation of the OS

### 1.1 Objectives

The goal of this first session is to install a Debian/Linux operating system from scratch on a Intel x86-based computer. The installation will be made on a removable USB disk. More information about installing Linux can be found in [1, 2]. A detailed description of the Linux boot process can be found in [3]

At the end of installation you should be able to use the installed operating system, i.e., doing a correct boot and login process.

### 1.2 Before you start

- Review UNIX basic commands: `cd`, `ls`, ...
- Basic usage of the `vi` editor

The rest of the session assumes the user connected as superuser, to become superuser we run the following command:

```
$ sudo su
```

### 1.3 Installation

#### 1.3.1 Hardware Identification

There are many graphical tools that allow to identify, list, and provide details about the hardware present on a system. However, during this first session we will provide some low level tools and commands that may be of assistance when no graphical interface is present on the system. We will mostly focus on the identification and understanding of the hard drive topology.

First we will identify the different hardware components present on the system. Hence we run:

```
# lspci
...
# lsusb
```

These commands allow to identify the devices directly connected to PCI and USB buses. Look at the manpage for each command to see the accepted parameters. Now try to fill as many entries as you can in Table 1.1.

Now, we will try to map the found hardware parts with the actual devices, plus try to find the devices not found using the previous commands. To do so we run:

```
# dmesg
```

Now you should be able to finish filling up the table.

Part	Hardware found	Device Name
<i>Network Card</i>		
<i>Internal Hard Drive</i>		
<i>USB Hard Drive</i>		

Table 1.1: Hardware table

### 1.3.2 Disk configuration: partitions

The first step to install the system is to partition the disk on the USB drive. First step is to find which is the device corresponding to the usb drive, it is in the form: `/dev/usb1`.

An important point before continuing, Ubuntu automatically mounts all external drives, this may lead to issues later during the lab, as the disk already has some partitions. As a consequence, it is **strongly recommended** to unmount all the external drives before continuing. To do so run:

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

Now, to partition the disk use the command `fdisk` on the device `/dev/usb`. With this command you can:

- Determine which is the geometry of your disk, and its size.
- Create the partitions indicated in Table 1.2 (select the appropriate size for each one). Since we will be using an 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.
- Write the contents of the partition table to disk (remember to do so before leaving from the "fdisk" command).

Device	Type	File system	Size	Mount-point	Comments
<code>/dev/usb1</code>	Primary	ext4		/	At least 10Gb
<code>/dev/usb3</code>	Primary	-swap-			
<code>/dev/usb5</code>	Logical	ext4		<code>/usr/local</code>	
<code>/dev/usb6</code>	Logical	ext4		<code>/home</code>	
<code>/dev/usb4</code>	Primary	free	20GB		Reserved for future use

Table 1.2: Partition table

Check that the device files corresponding to the new partitions appear in the `/dev` directory. Write here the full names of such new *files*:

For the system to be able to boot later, the `/` partition must have the bootable flag enabled, this can be accomplished using `fdisk`, indicate which option you used to do so:

<sup>1</sup>Be careful, *usb* actually is not the device name, find the proper one

Don't forget to save the changes before exiting `fdisk`.

### 1.3.3 Disk configuration: Creating the file systems

Once you have created the necessary partitions, you must initialize the file system on those partitions that will contain your files, and prepare the swap area for use (this step is not actually mandatory but recommended on embedded systems).

To format the swap area should use the command:

```
# mkswap device
```

Later, you can activate the swap area with:

```
# swapon device
```

To create a Linux file system in the other partitions, we use the command:

```
# mkfs.fstype device
```

For each partition where you initialize a filesystem, where `fstype` can be: `ext4`, `btrfs`, `vfat`, `reiserfs`, ... as the type of filesystem you want to create on the partition.

Depending on the type of file system that you want to use, the options to create the file system can be different. See the different options which can be used on each command help.

Now give format to the partitions you made earlier given the indications you have on the table.

### 1.3.4 Mounting the filesystems to install the system

Now we have to mount the filesystems in a temporary directory to be able to install the software. Let's create the mount points for the installation, using a new directory:

```
# mkdir /linux
```

and now we will mount all filesystems inside such a mount point (`/linux`), creating the appropriate directories inside, at the same time.

```
# mount partition directory
```

Table 1.2 shows in which mount point (directory) each partition has to be mounted (remember, always inside `/linux`; for instance, `/` partition into `/linux`, `/home` partition into `/linux/home`, etc.). It is **important** to create the directories after mounting `/linux` but **before** mounting the rest.

### 1.3.5 Installation of the base system

Once you have prepared the partitions, our next step is to install the base operating system. This process may vary depending on the system. Usually system software is organized into packages, and the software installer decompresses them into the destination directory, and then automatically configures them (maybe with some hints from the user).

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
```

And now get the image:

```
# sftp aso@asoserver.pc.ac.upc.edu
```

Use the following password: `AsORoCkSHaRd!`

From there you can download the file `/packages/aso-install.tar.gz`. To finally untar the file:

```
# tar xzf aso-install.tar.gz
```

Now look at the contents of the `/linux` directory. You should see that it has been populated with the basic components of your future system.

Optionally we can erase the downloaded file by running:

```
# rm aso-install.tar.gz
```

### 1.3.6 Finish mounting auxiliary systems

We also have to **bind-mount** the directories `/dev`, `/sys`, and `/proc` inside `/linux`, to temporarily expose the current existing devices to the new system. To do that, use the following commands:

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

Use the `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.

What is the purpose of the flag `-o bind` in the `mount` command?

## 1.4 Basic system configuration

Before rebooting, you should perform some more steps: configure system mountpoints through the `/etc/fstab` file and install a boot loader.

The configuration files in operating systems based on Unix/Linux are on the default directory `/etc` and almost always in text format. The Linux environments have many tools for text processing from command line (`grep`, `sed`, `tail`, `cut` ...) and editors (`vi`, `nano`, `emacs`, ...). During the installation process, only the `'vi'` editor will be available in most cases.

### 1.4.1 Configuring the file systems table (`/etc/fstab`)

In order for the file systems to be mounted correctly at system power-on, you need to generate a correct `/etc/fstab` file. To do this edit the file and make sure that the following parts exist:

- 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

Why `/proc` and `/sys` do not have any device attached?

### 1.4.2 Changing root directory

At this point, 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. To change the root of your system, use:

```
# chroot /linux
```

From this point on, you can use the system we have installed, and access for example manual pages with the command `man`.

### 1.4.3 Configuring the keyboard

In Debian, the keyboard layout is configured by running:

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

In the `locales` configuration make sure you select a suitable locale for your machine, by searching for it and pressing `<Spacebar>` to select it.

### 1.4.4 Configuring the boot process

Formerly, the operating system was installed on a partition that was marked as bootable in the partition table. The BIOS was searching for it, and then booting the system. This meant that we could have only one system working on a PC, and that if we wanted to boot from another partition, we should change the partition table and reboot. To address this limitation, second-level boot managers (bootstrap loaders) help in booting several operating systems. A boot loader is a set of programs residents in the disk drive, that allow the user to load other operating systems (including from other disk drives). They do the same as BIOS does with them: loading the OS into memory and transferring control. Among the most used we find: *LILO* (Linux Loader), *GRUB* and *NTLDR* (used in systems from Microsoft).

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.

To configure correctly the boot of the machine using *GRUB*, we need to execute the script provided by Linux (remember to keep your root (*/*) directory set to */linux*):

```
# 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 (you do not need to repeat them) are:

- Creates the directory */boot/grub*.
- Copy the files needed for *GRUB* to */boot*.
- Installs the bootloader in the MBR of the USB disk.

Besides the boot configuration, the system must inform to *GRUB* which kernel must be used to properly boot, this can be accomplished by editing the file */boot/grub/grub.cfg*. Since this may be an involved process, 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 it is necessary to run the script:

```
# update-grub
```

To have an example of the output of this command, give a look to the */boot/grub/grub.cfg* file, which has been autogenerated. In *GRUB*, and other system tools, e.g, */etc/fstab*, rather than using partition names, we tend to use the partition UUID.

What do you think is the UUID?



**Note:** it is possible to get the UUID of a particular harddisk partition using the command `blkid`.

A useful way to access to grub configuration is by pressing the 'e' key when grub is loaded at boot time, this allows us to edit boot options (without saving them) to boot when there is an error in the file *grub.cfg*.

### 1.4.5 Setting up the passwords

In the current system, the existing user passwords are set to defaults, which is not what we want. 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`.

Now change the passwords for the user `aso` and the user `root`:



You can now exit the `chroot` shell.

Unmount all filesystems by running the relevant `umount` commands, and reboot using the `shutdown` command.

How have you executed the `shutdown` command?

Could we use other system commands to reboot? Which ones?

## 1.5 Post-configuration

Boot the newly installed system. Now you have to perform a series of system configuration tasks on this new system. The system has two user accounts: `root` and `aso`. Login as `aso`. Use the password you set previously. In general you should use always an unprivileged user to minimize the possibility of damaging your system by mistake. When you need to have user privileges (i.e. become root), use the command `su`:

```
$ su
# privileged-command
# exit
```

or

```
# su -c "privileged-comand"
```

### 1.5.1 Configuring filesystems

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

What other parameters can be adjusted with the `tune2fs` command?

### 1.5.2 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.

Where are configuration file commonly located?

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 for something like this (usually a message indicating that normal users activities can be recorded for security reasons):

```
#####
# This system is for the use of authorized users only.          #
# Individuals using this computer system without authority, or in#
# excess of their authority, are subject to having all of their #
# activities on this system monitored and recorded by system    #
# personnel.                                                    #
#                                                                #
# In the course of monitoring individuals improperly using this #
# system, or in the course of system maintenance, the activities#
# of authorized users may also be monitored.                    #
#                                                                #
# Anyone using this system expressly consents to such monitoring #
# and is advised that if such monitoring reveals possible       #
# evidence of criminal activity, system personnel may provide the#
# evidence of such monitoring to law enforcement officials.     #
#####
```

Which file should contain this message? (Hint: search for the file with the original contents that you want to replace) And which command did you use to find the file.



After login, we obtain another message, the motd (or Message-Of-The-Day). Usually, this is used to give last-minute information to the users about the status of the system (for instance, contact information or system news).

Locate such file, and change it to report on how to contact with the system administrators.

Which file have you modified?



### 1.5.3 Network configuration

The next step in the lab session is to configure the network. This means that once this stage is complete, your system should be capable of communicate with other systems via the IP protocol. First, we will do the 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, this may be achieved by running:

```
# ip link set dev <ethernet IF> downa
```

<sup>a</sup>Change <ethernet IF> by the actual name of the interface. Try `ip link show`

#### Manual configuration

The manual network setup usually involves three steps:

1. Configuring the network interface using the `ip address` command
2. Configuring the routing table using `ip route` command
3. Setup name resolution in `/etc/resolv.conf`

Check the *man pages* that correspond to these commands and files.

Which interfaces are configured in your system?

What is the full list of interfaces available? (include the unactive ones)

To configure properly the network, have in mind the data of Table 1.3.

Parameter	Configuration value
IP address	
Mask	
Gateway	
DNS Server	

Table 1.3: Network information

Configure the Ethernet interface corresponding to the Gigabit Ethernet. Which command do you use to bring the network interface up?

Now you have to add the default gateway to the routing table. Which command do you use?

Finally, create the file `/etc/resolv.conf` with the DNS information. The Name Server is 147.83.41.104. How can we check that we have correctly configured our network?

### Permanent configuration

Now we want the network to be configured properly at boot time and do not have to do it manually each time. First, bring the interface down using:

```
# ip link set dev <ethernet IF> down
# ip address del 10.10.41.???.???.???.??/24 dev <ethernet IF>
```

Use `ip address show` to verify that the interface no longer comes to the list of active interfaces.


On Debian (and other systems) the network configuration resides in several files in the `/etc/network` directory.

Since Debian 8 (Jessie), the system uses `systemd` as init system, use the command

```
# systemctl list-units
```

To see all the available services and their status on the system. Which one is related with networking?





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 `interfaces` file to configure your network interface with the parameters you used previously. First add a line to indicate that we want to activate the interface automatically at boot:

```
auto <ethernet IF>
```

After indicating that, we will give all the necessary parameters to configure the interface:

```
iface <ethernet IF> inet static
```

And immediately all the necessary parameters (except the DNS server):

```
address 10.10.41.XX
network 10.10.41.0
netmask 255.255.255.0
gateway 10.10.41.1
```

Now, to check that we have configured it correctly, we could do a reboot. But in fact we do not need that. We can use the commands


```
# ifup <ethernet IF>
```

```
# ifdown <ethernet IF>
```

to tell the system to reconfigure an interface according to the `interfaces` file.

Once you have made it work, now we want to obtain the network settings the system automatically using DHCP. Check the manual `interfaces` file (**man interfaces**) and use DHCP to configure `<ethernet IF>`.

What changes you made to `/etc/network/interfaces`?





# Bibliography

- [1] M. Kalle and M. Welsh, *Running Linux*, 5th ed. Sebastopol, USA: O'Reilly Media, 2005.
- [2] L. Wirzenius, J. Oja, S. Stafford, and A. Weeks, *The Linux System Administrators' Guide, version 0.9*. The Linux Documentation Project. TLDP. [Online]. Available: <http://www.tldp.org/LDP/sag/sag.pdf>
- [3] M. T. Jones, "Inside the linux boot process," IBM Developer Works, 2006. [Online]. Available: <http://www-128.ibm.com/developerworks/linux/library/l-linuxboot/?ca=dgr-lnxw06LinuxBoot>