

Operating Systems [2020-2021]

Tutorial – Installing Linux and the required tools

Note: this tutorial will not be done during the practical classes, although questions will be clarified by the teacher.

Introduction

The goal of this Tutorial is to guide the student through the process of installing the Linux operating system, together with all the tools required for the programming classes and assignments, throughout the semester. In order to preserve the student's current machine, the installation will be made using VirtualBox virtualization software.

Installing VirtualBox

Our first goal is to download and install **VirtualBox**, the Virtual Machine manager, available from <https://www.virtualbox.org/wiki/Downloads>. You should download the installation package most appropriate to your host system and begin the installation process, as illustrated in Figure 1.

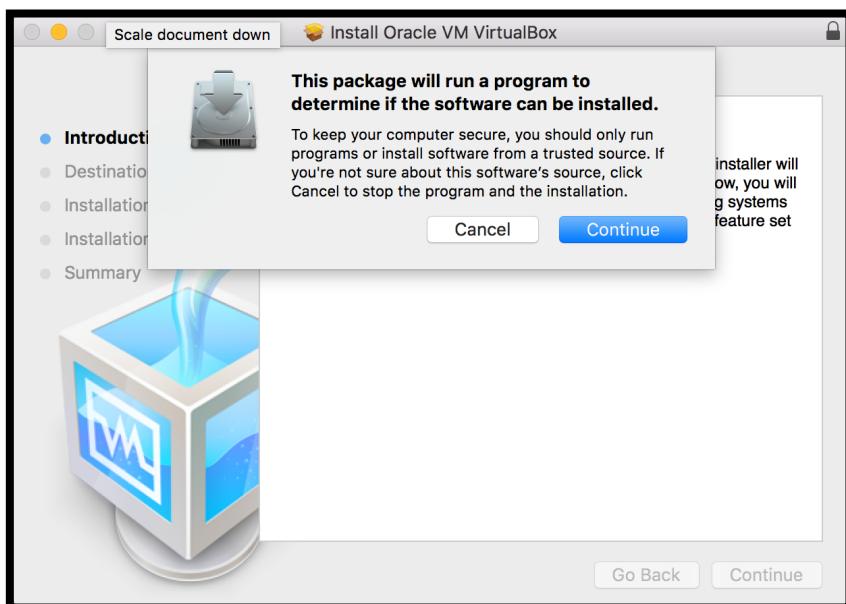


Figure 1 – VirtualBox installation

After installing VirtualBox you should also install the **VirtualBox Extension Pack**, available from <https://www.virtualbox.org/wiki/Downloads>. The Extension Pack is required to improve integration of the Virtual Machine environment with the host operating system.

Downloading and installing Linux

We next are going to download and install Linux in a new Virtual Machine, using the VirtualBox manager. For this purpose, we begin by downloading an ISO image of a recent version of the Linux Ubuntu operating system (17.04 at time of writing of the assignment), available from:

<ftp://ftp.dei.uc.pt/pub/linux/ubuntu/releases/17.04/ubuntu-17.04-desktop-i386.iso>

Alternatively, Ubuntu can also be downloaded from the home page of the project, at: <https://www.ubuntu.com/download>. After having downloaded the ISO image to your computer, create a new Virtual Machine. This process starts by asking information about what type of operating system (and release) we are going to install, and is illustrated in Figure 2.

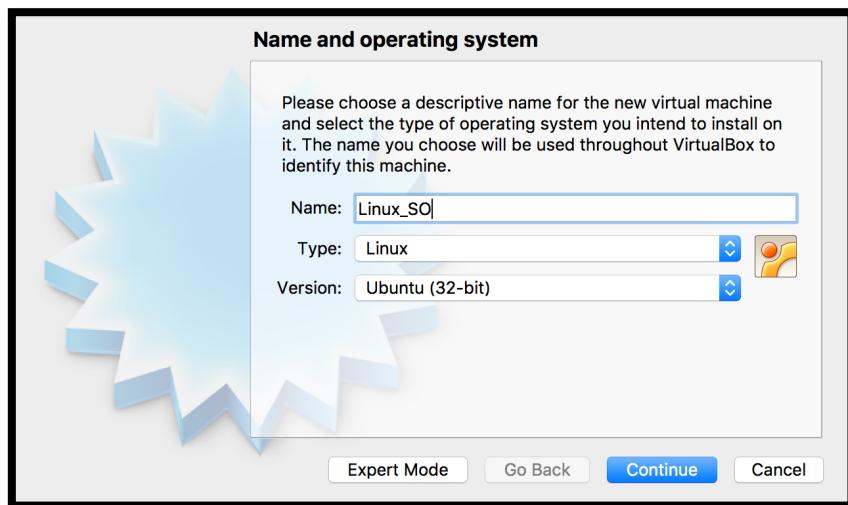


Figure 2 – Creating a new Virtual Machine for Linux

You should allocate the required resources (memory and disk space) to the new Virtual Machine, depending on the availability on your system. This process is illustrated in Figures 3 to 7, and in this example we allocate 2Gb of RAM and 10Gb of disk space to our new Virtual Machine.

Please note: you are advised to allocate at least 7Gb of disk space to the new Virtual Machine and at least 1Gb of RAM (or more, if available).

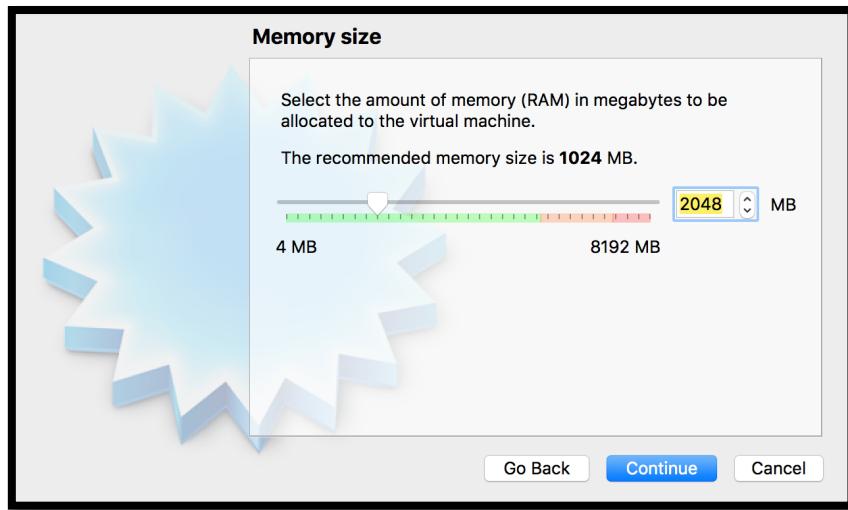


Figure 3 – Configuring RAM for the Linux VM

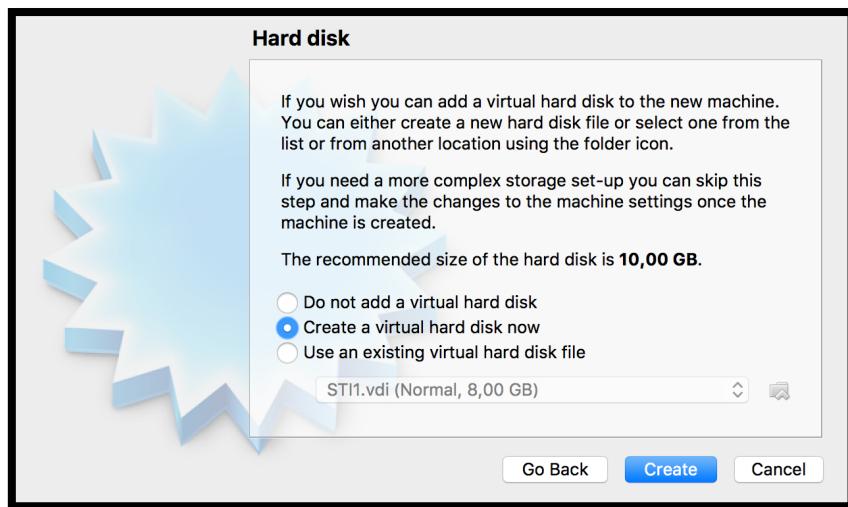


Figure 4 – Configuring the type of Hard Disk

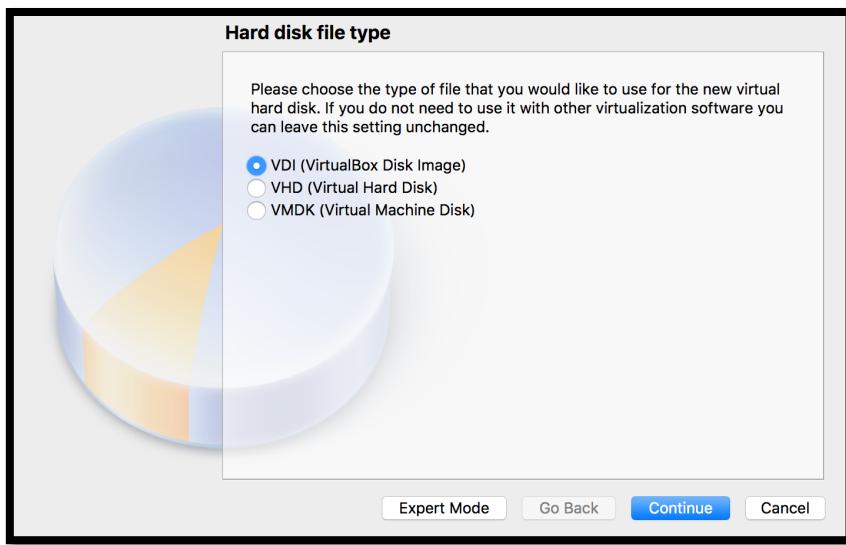


Figure 5 - Configuring the type of Hard Disk (cont.)

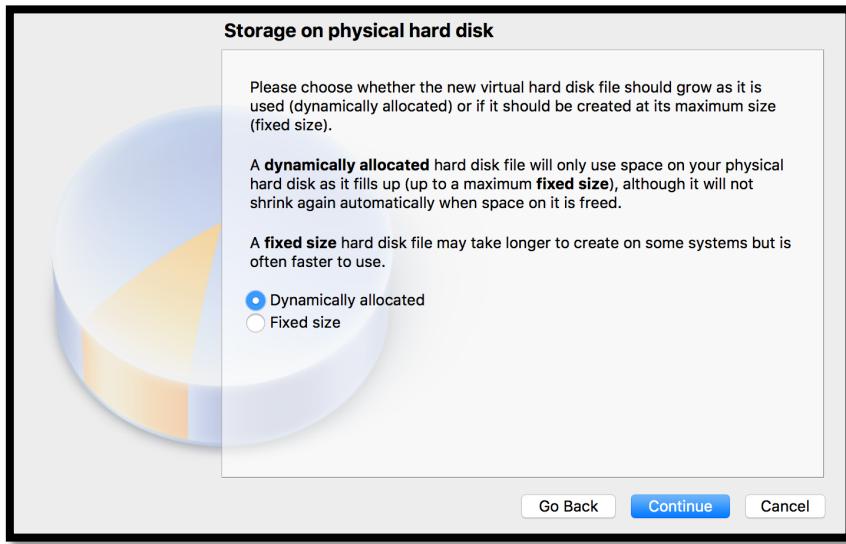


Figure 6 – Configuring storage

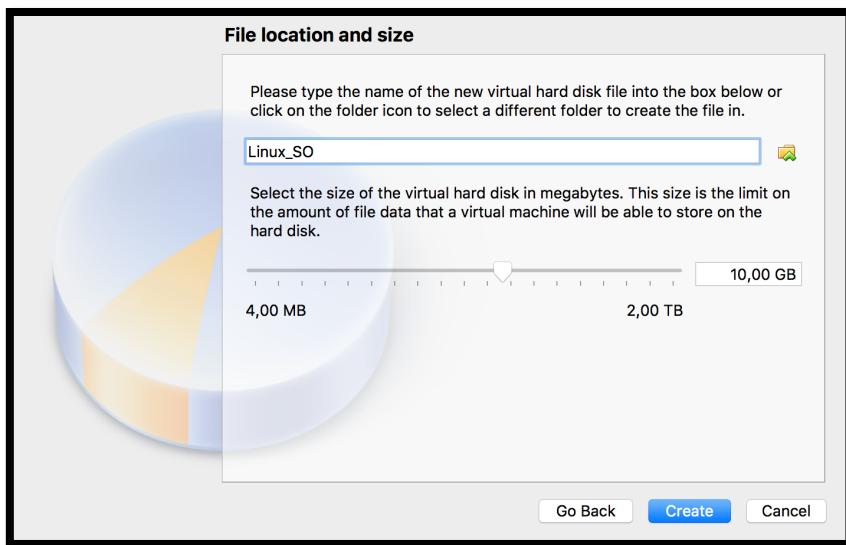


Figure 7 - Configuring disk space

We are now ready to start installing Linux Ubuntu on our just created Virtual Machine in VirtualBox. For this purpose, just boot up the Virtual Machine and locate and select the file containing the ISO image previously downloaded, as Figure 8 illustrates.

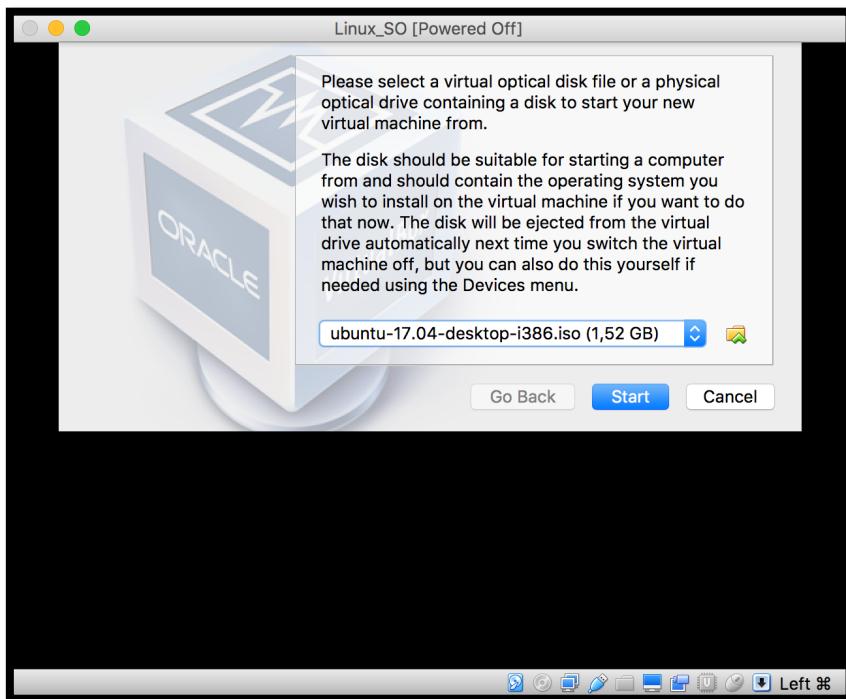


Figure 8 – Selecting the ISO image for the Linux installation

During the installation process, will also be asked for the required information, such as the Location, type of Keyboard and user information. We will also be creating a personal account that you'll use during the semester, as illustrated in Figures 9 to 12.

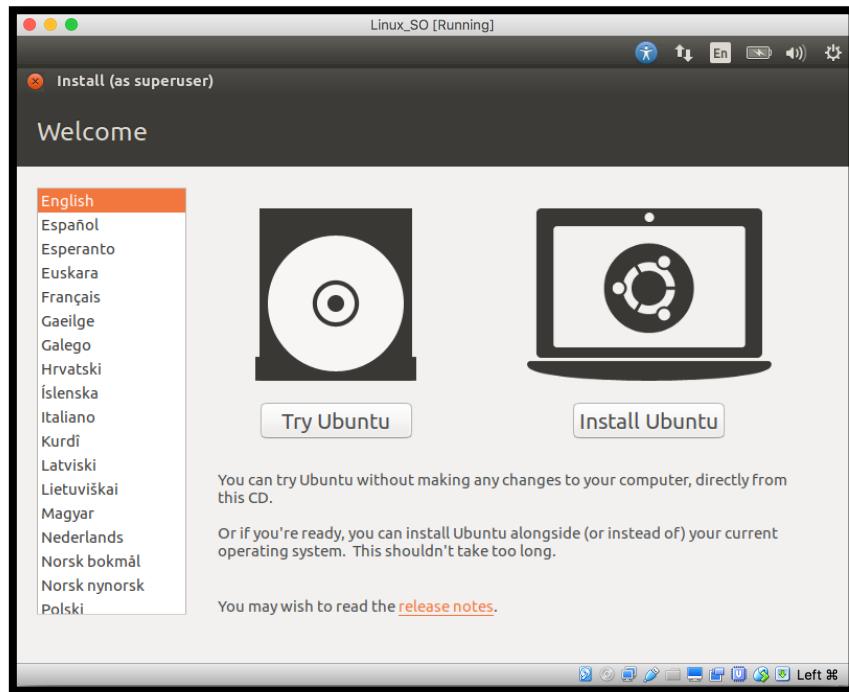


Figure 9 – Start installing Linux Ubuntu

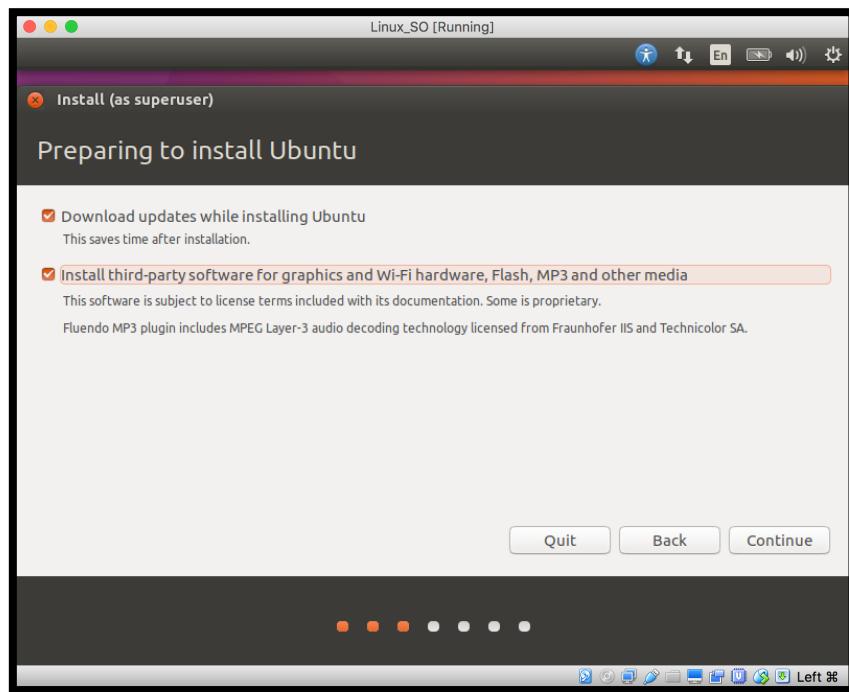


Figure 10 – Download options (during the installation)

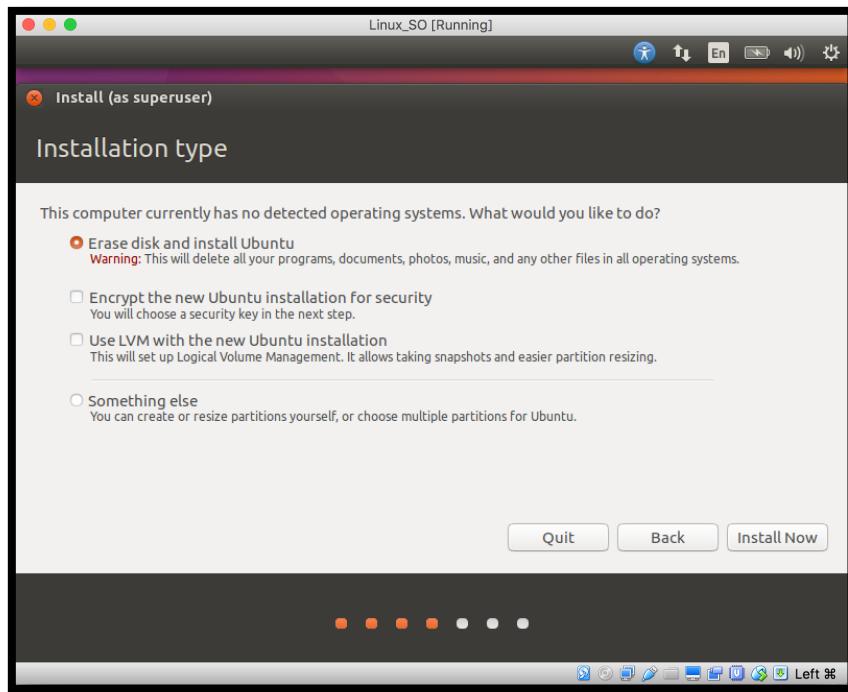


Figure 11 – Selecting the installation type

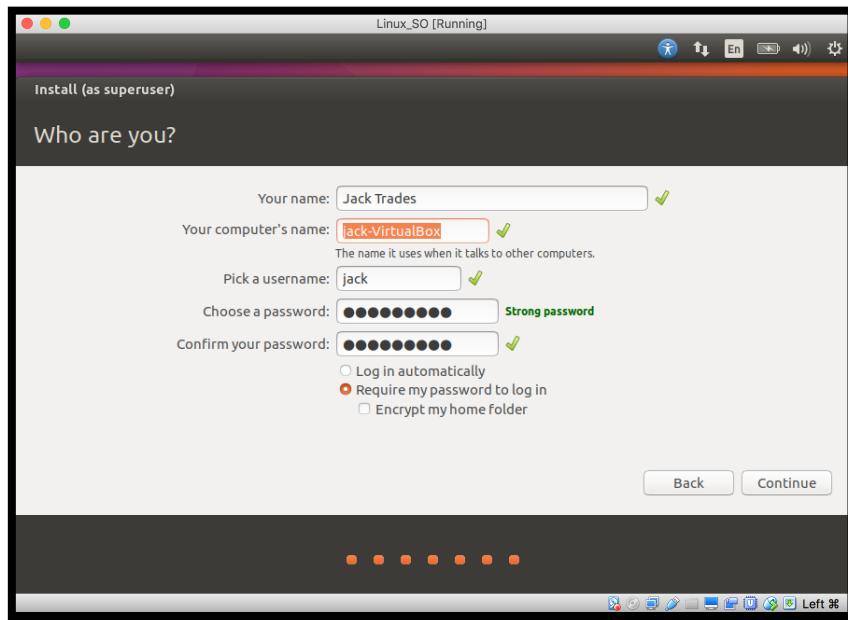


Figure 12 – Creating a (personal) user account on the Linux system

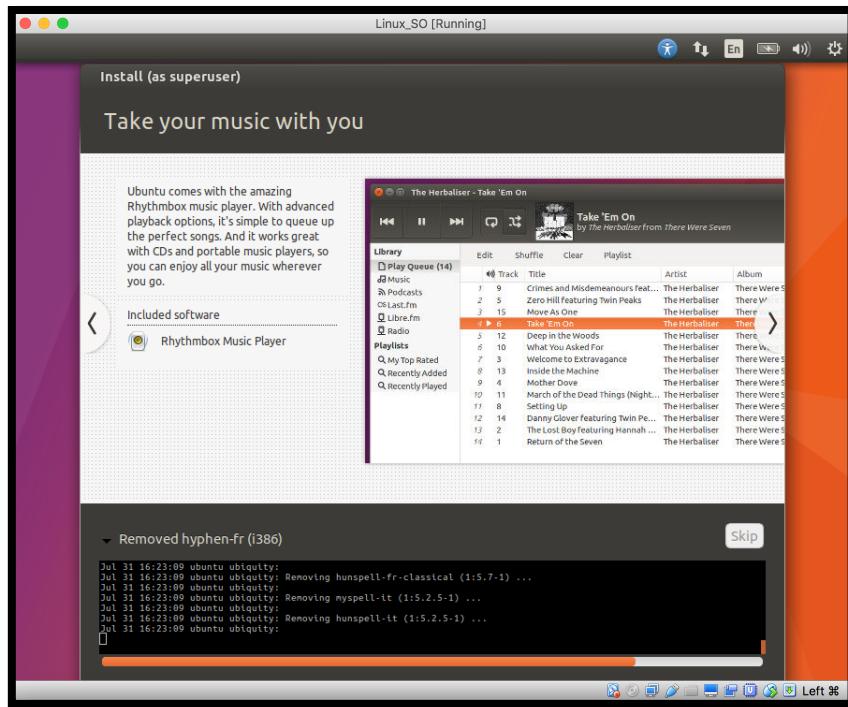


Figure 13 – Installation of the required software

After the installation of the required process (Figure 14), the system will be ready for our first login (using the previously created account), as Figure 15 illustrates.

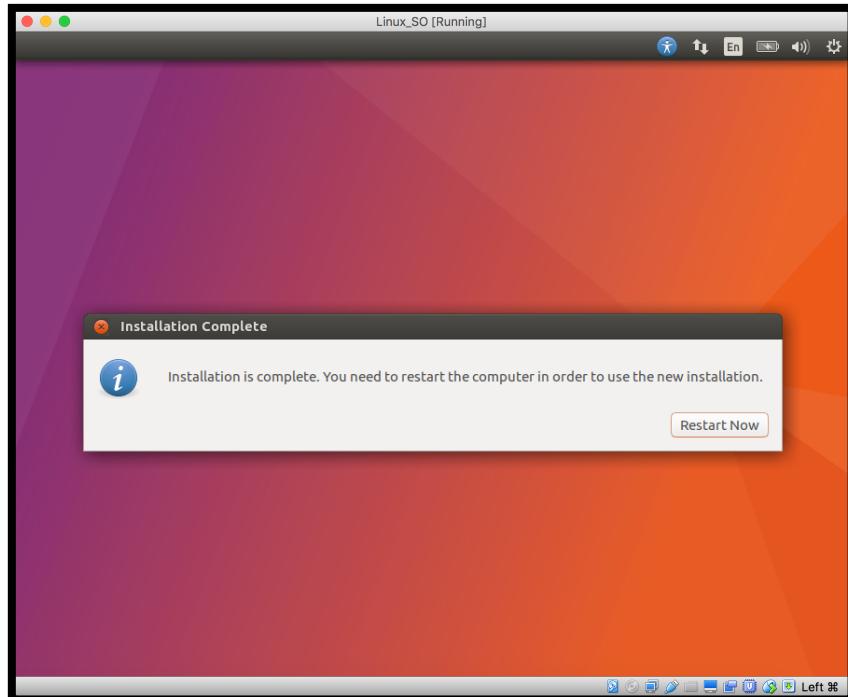


Figure 14 – Conclusion of the installation process

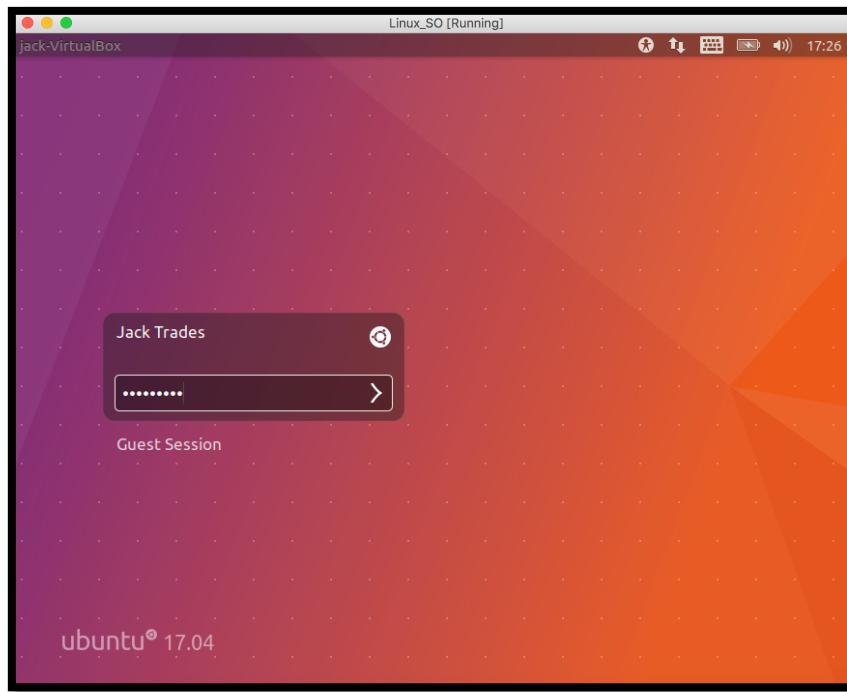
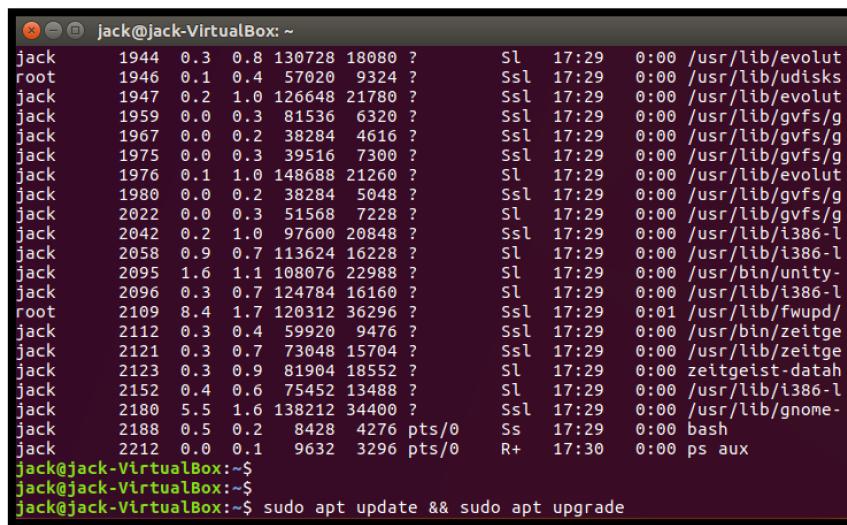


Figure 15 – The Linux system is now available for our first login

As important final installation steps, we should update the system, as illustrated in Figure 16.



A terminal window titled "jack@jack-VirtualBox: ~" showing the output of a "top" command. The terminal also contains commands for updating the system: "sudo apt update && sudo apt upgrade".

```
jack      1944  0.3  0.8 130728 18080 ?          Sl  17:29  0:00 /usr/lib/evolut
root     1946  0.1  0.4 57020  9324 ?          Ssl 17:29  0:00 /usr/lib/udisks
jack      1947  0.2  1.0 126648 21780 ?          Ssl 17:29  0:00 /usr/lib/evolut
jack      1959  0.0  0.3  81536  6320 ?          Ssl 17:29  0:00 /usr/lib/gvfs/g
jack      1967  0.0  0.2  38284  4616 ?          Ssl 17:29  0:00 /usr/lib/gvfs/g
jack      1975  0.0  0.3  39516  7300 ?          Ssl 17:29  0:00 /usr/lib/gvfs/g
jack      1976  0.1  1.0 148688 21260 ?          Sl  17:29  0:00 /usr/lib/evolut
jack      1980  0.0  0.2  38284  5048 ?          Ssl 17:29  0:00 /usr/lib/gvfs/g
jack      2022  0.0  0.3  51568  7228 ?          Sl  17:29  0:00 /usr/lib/gvfs/g
jack      2042  0.2  1.0  97600 20848 ?          Ssl 17:29  0:00 /usr/lib/i386-l
jack      2058  0.9  0.7 113624 16228 ?          Sl  17:29  0:00 /usr/lib/i386-l
jack      2095  1.6  1.1 108076 22988 ?          Sl  17:29  0:00 /usr/bin/unity-
jack      2096  0.3  0.7 124784 16160 ?          Sl  17:29  0:00 /usr/lib/i386-l
root     2109  8.4  1.7 120312 36296 ?          Ssl 17:29  0:01 /usr/lib/fwupd/
jack      2112  0.3  0.4  59920  9476 ?          Ssl 17:29  0:00 /usr/bin/zeitge
jack      2121  0.3  0.7  73048 15704 ?          Ssl 17:29  0:00 /usr/lib/zeitge
jack      2123  0.3  0.9  81904 18552 ?          Sl  17:29  0:00 zeitgeist-datah
jack      2152  0.4  0.6  75452 13488 ?          Sl  17:29  0:00 /usr/lib/i386-l
jack      2180  5.5  1.6 138212 34400 ?          Ssl 17:29  0:00 /usr/lib/gnome-
jack      2188  0.5  0.2   8428  4276 pts/0       Ss  17:29  0:00 bash
jack      2212  0.0  0.1   9632  3296 pts/0       R+ 17:30  0:00 ps aux
jack@jack-VirtualBox:~$ sudo apt update && sudo apt upgrade
jack@jack-VirtualBox:~$
```

Figure 16 – Updating the system

Bootloader and partitions

A bootloader is the program that loads an operating system when the machine boots up and, in the case of Linux Ubuntu, GRUB2 (**G**Rand **U**nified **B**ootloader) is the default boot loader installed. Although the following configurations are not required, they exemplify how the GRUB2 configuration may be changed after the installation of the operating system, as per our previous steps.

All settings related to the configuration of the GRUB2 boot loader are stored in a single configuration file, the **/etc/default/grub** file. As for other configuration aspects in Linux, this file may be edited and changed, in order to fine tune the configuration of the boot loader. Figure 17 illustrates this process, in this case using the *nano* text editor.

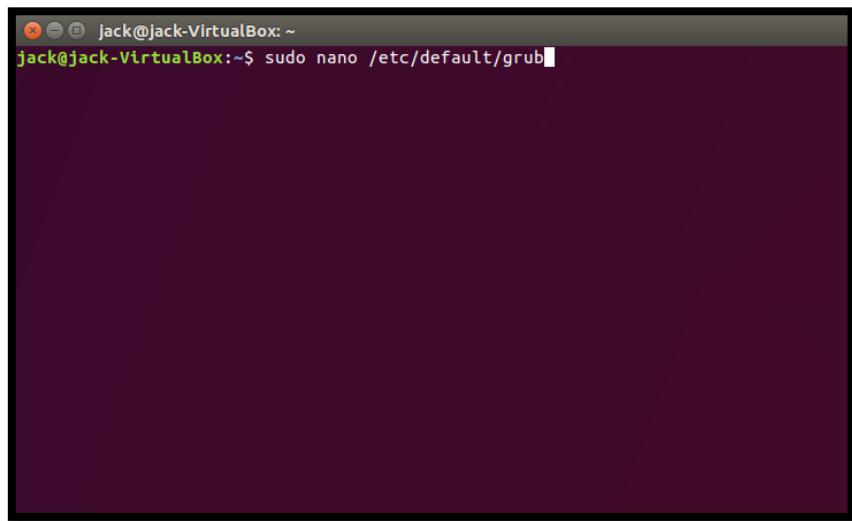
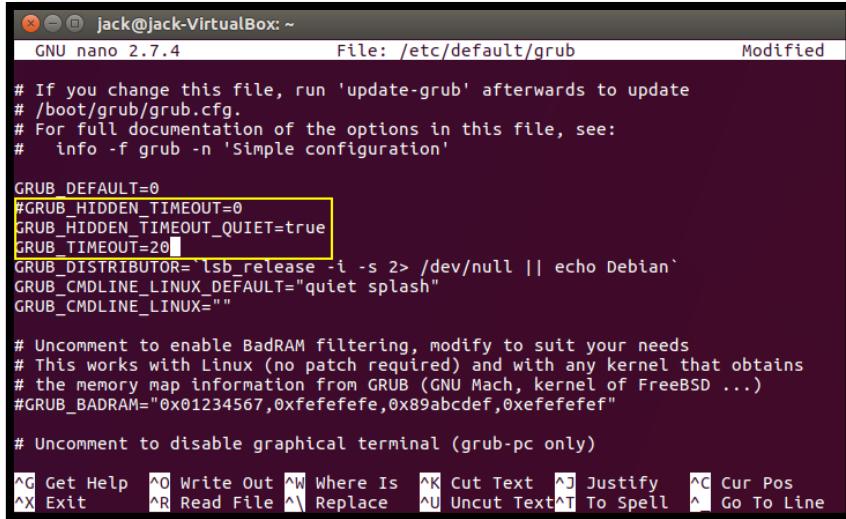


Figure 17 – Editing the GRUB2 configuration file

In Figure 18 we illustrate two configuration settings that are changed for GRUB2: the timeout before GRUB2 boots up the selected operating system (in this example we change this timeout to 20s), and forcing the boot menu to show up at the beginning of the boot process.



```

jack@jack-VirtualBox: ~
GNU nano 2.7.4           File: /etc/default/grub           Modified

# If you change this file, run 'update-grub' afterwards to update
# /boot/grub/grub.cfg.
# For full documentation of the options in this file, see:
#   info -f grub -n 'Simple configuration'

GRUB_DEFAULT=0
#GRUB_HIDDEN_TIMEOUT=0
GRUB_HIDDEN_TIMEOUT_QUIET=true
GRUB_TIMEOUT=20
GRUB_DISTRIBUTOR= `lsb_release -i -s 2> /dev/null || echo Debian`
GRUB_CMDLINE_LINUX_DEFAULT="quiet splash"
GRUB_CMDLINE_LINUX=""

# Uncomment to enable BadRAM filtering, modify to suit your needs
# This works with Linux (no patch required) and with any kernel that obtains
# the memory map information from GRUB (GNU Mach, kernel of FreeBSD ...)
#GRUB_BADRAM="0x01234567,0xfefefefe,0x89abcdef,0xefefefef"

# Uncomment to disable graphical terminal (grub-pc only)

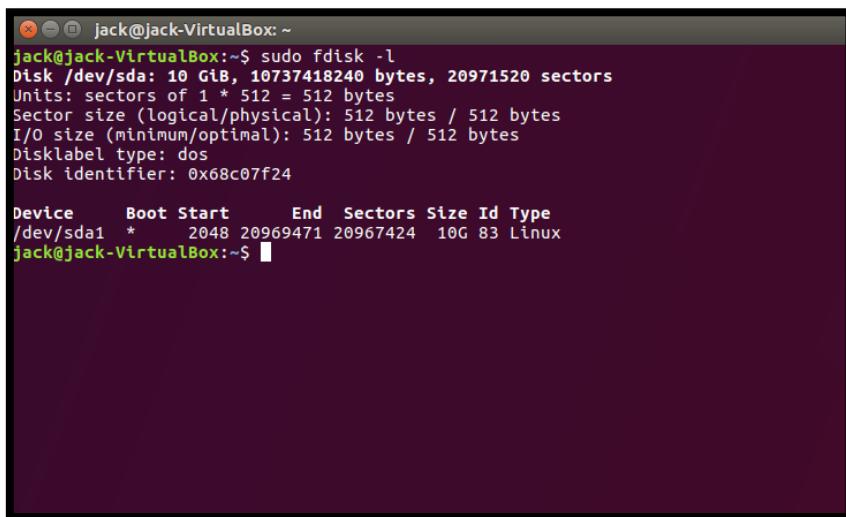
^C Get Help  ^O Write Out  ^W Where Is  ^K Cut Text  ^J Justify  ^C Cur Pos
^X Exit      ^R Read File  ^\ Replace   ^U Uncut Text  ^T To Spell  ^G Go To Line

```

Figure 18 – Some (simple) changes to the GRUB2 configuration

After saving the GRUB2 configuration file, the configuration may be updated by running the command ***sudo update-grub***.

Another fundamental aspect to consider in the configuration of the Linux operating system is that of the disk partitions used by the operating system. In our previous installation of Linux, we have performed an automatic installation and thus the available space (the virtual partition created using VirtualBox) was automatically partitioned and formatted for us. We can analyze the current configuration in terms of the used disk partitions, as illustrated in Figure 19 using the ***fdisk*** command. We may note that, as requested before, the installation has created a Linux partition with 10Gb.



```

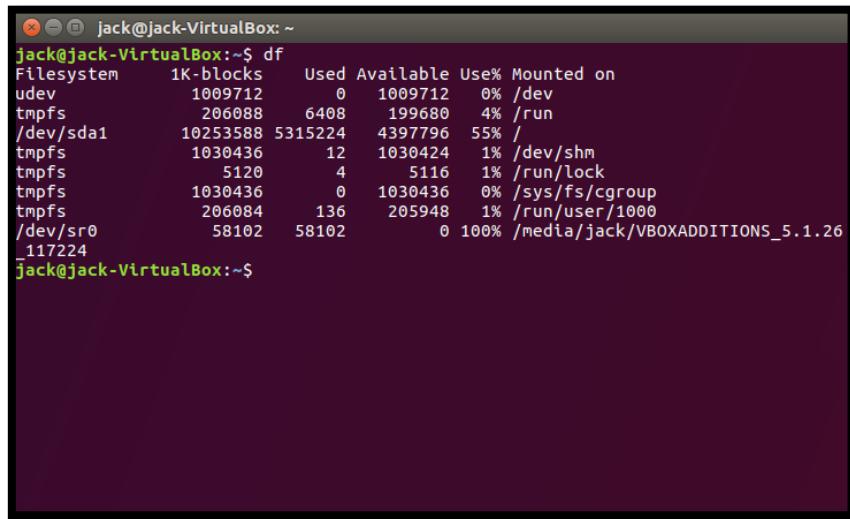
jack@jack-VirtualBox: ~$ sudo fdisk -l
Disk /dev/sda: 10 GiB, 10737418240 bytes, 20971520 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x68c07f24

Device     Boot Start      End  Sectors Size Id Type
/dev/sda1  *     2048 20969471 20967424  10G 83 Linux
jack@jack-VirtualBox: ~$ 

```

Figure 19 – Available partitions

Another useful command in Linux is ***df***, which allows us to visualize the various mounted partitions, as well as the space available in each partition, as illustrated in Figure 20. As we may observe, the main file system used by the Linux operating system (mounted in the '**/**' root directory from the /dev/sda1 partition) currently occupies approximately 55% of the available 10Gb of disk space.



```
jack@jack-VirtualBox:~$ df
Filesystem      1K-blocks    Used Available Use% Mounted on
udev             1009712      0   1009712   0% /dev
tmpfs            206088   6408   199680   4% /run
/dev/sda1        10253588 5315224  4397796  55% /
tmpfs            1030436     12  1030424   1% /dev/shm
tmpfs              5120      4    5116   1% /run/lock
tmpfs            1030436      0  1030436   0% /sys/fs/cgroup
tmpfs            206084    136  205948   1% /run/user/1000
/dev/sr0           58102   58102      0 100% /media/jack/VBOXADDITIONS_5.1.26
117224
jack@jack-VirtualBox:~$
```

Figure 20 – File system disk space usage

Final configurations

As final configurations, we will install the **Virtual Guest Additions** in Linux (in order to enable integration with the host system), as well as the required **tools for development**, during the semester. Starting with the installation of the Virtual Guest Additions, select the “Insert Guest Additions CD Image” option from the VirtualBox Devices menu and follow the installation process, illustrated in Figures 21 and 22.

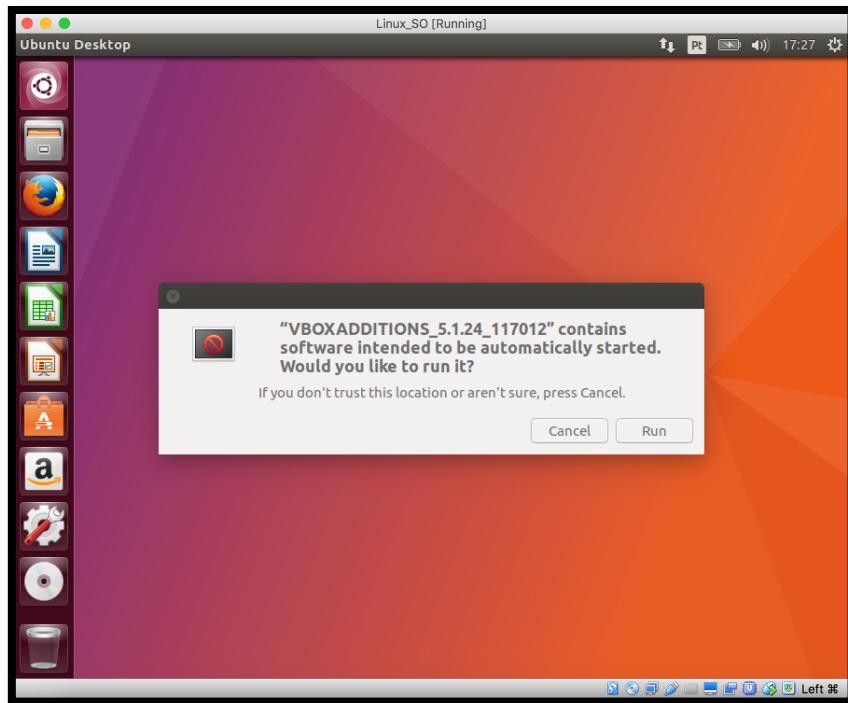


Figure 21 – Installing the VirtualBox Guest Additions

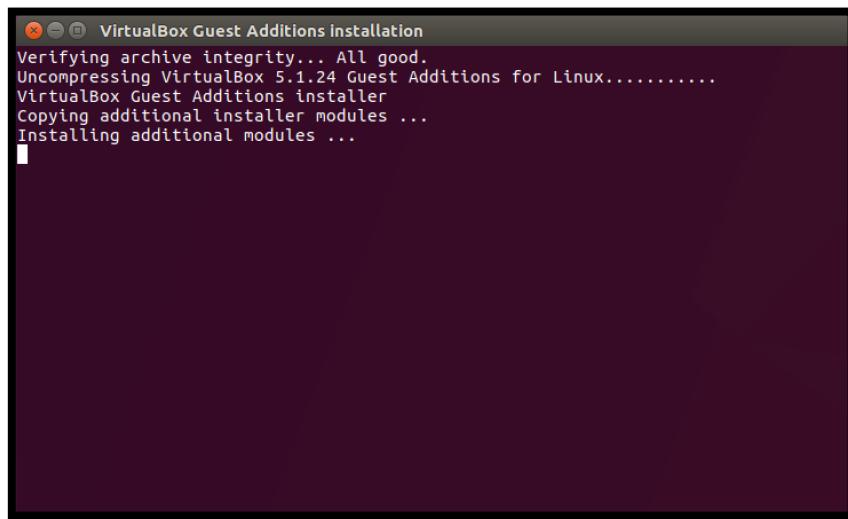
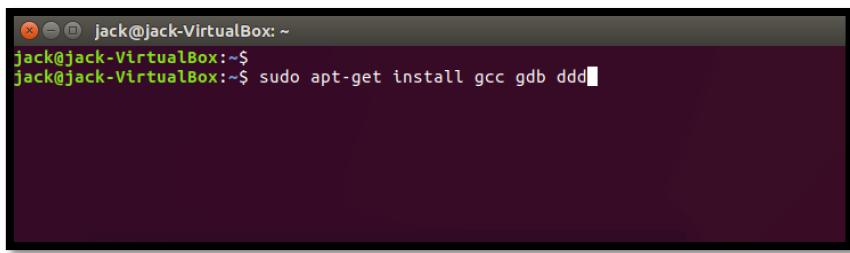


Figure 22 - Installing the VirtualBox Guest Additions (cont.)

Finally, we need to install the required packages for development, during the semester, which are illustrated in the Figure 23.



```
jack@jack-VirtualBox: ~
jack@jack-VirtualBox: ~$ sudo apt-get install gcc gdb ddd
```

Figure 23 - Installing the required development packages

More information about each of the previously installed packages may be obtained using the **sudo dpkg-query -s** command. In short, **gcc** is the GNU C compiler, **gdb** our source-level debugger and **ddd** a graphical debugger interface to the **bdb** debugger. Finally, please note that further configurations may be required in order to prepare your brand new operating system for using during our classes, in particular in what respects the keyboard layout.