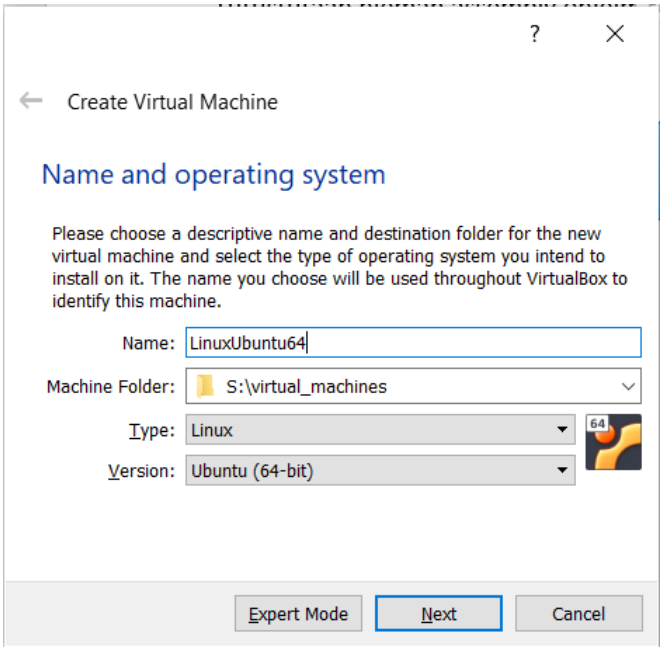
If you face problems with this task, participate to question time Zoom on Monday (week 10) or send e-mail to jani.heinikoski@student.lut.fi.

Installation of a virtual machine:

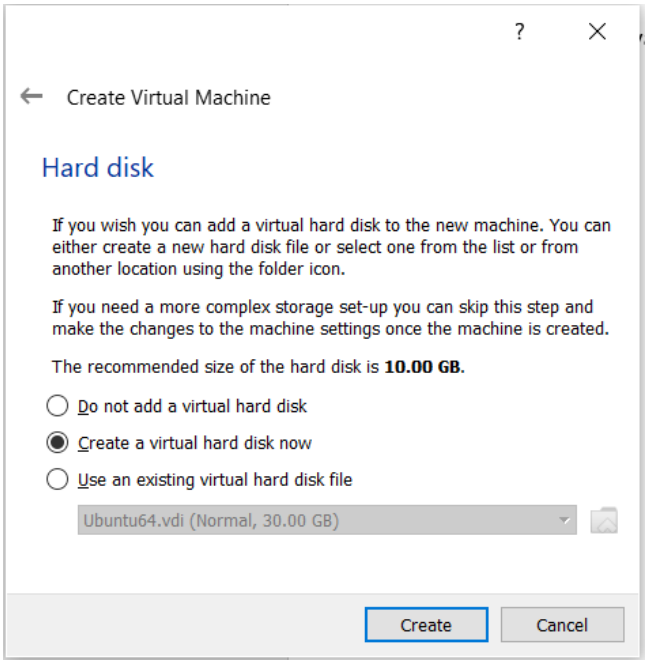
In order to install a virtual machine, a virtualization tool is needed. You can use whichever you like (there are several alternatives – f.ex. Hyper-V) but one very light virtual environment is Oracle's Virtualbox, which can be downloaded for free from here: https://www.virtualbox.org/

When you have installed the program, download an .iso file of 64-bit Ubuntu to your computer from here: https://ubuntu.com/download/desktop

After this, open Oracle Virtualbox and create a new virtual machine by pressing "New" button:

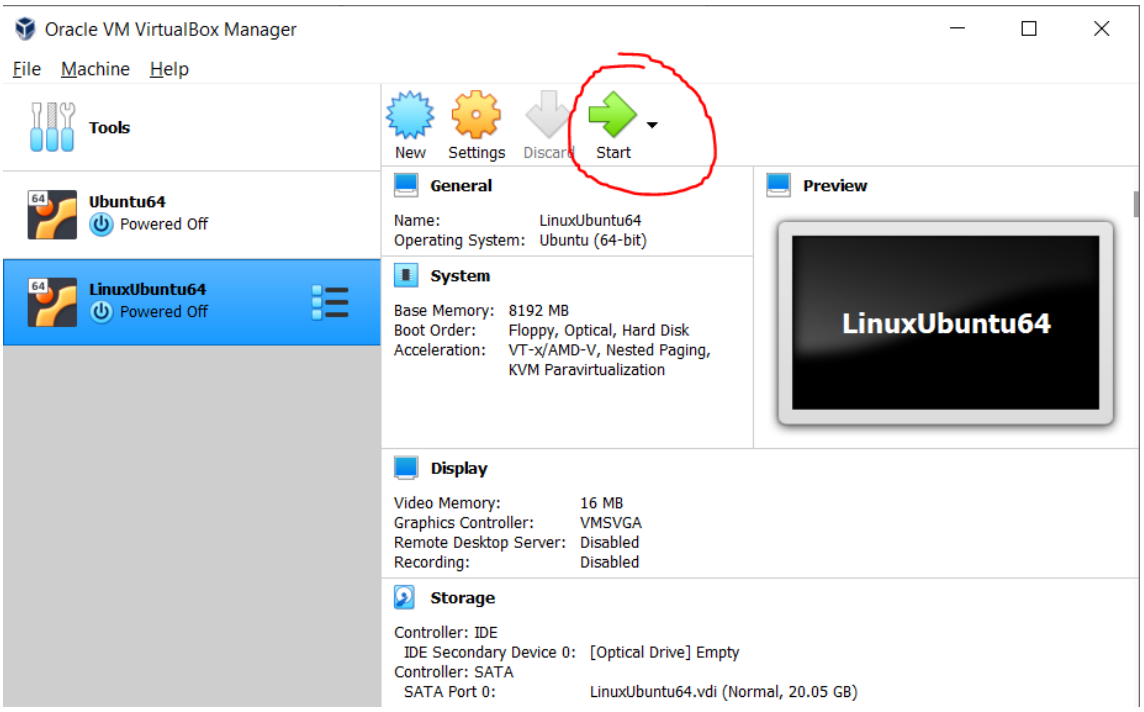


Make sure that all fields match the picture above (sans Name). Click "Next" and allocate an arbitrary amount of RAM for your virtual machine (recommended at least 2048 MB). Click "Next" and create a virtual hard disk:

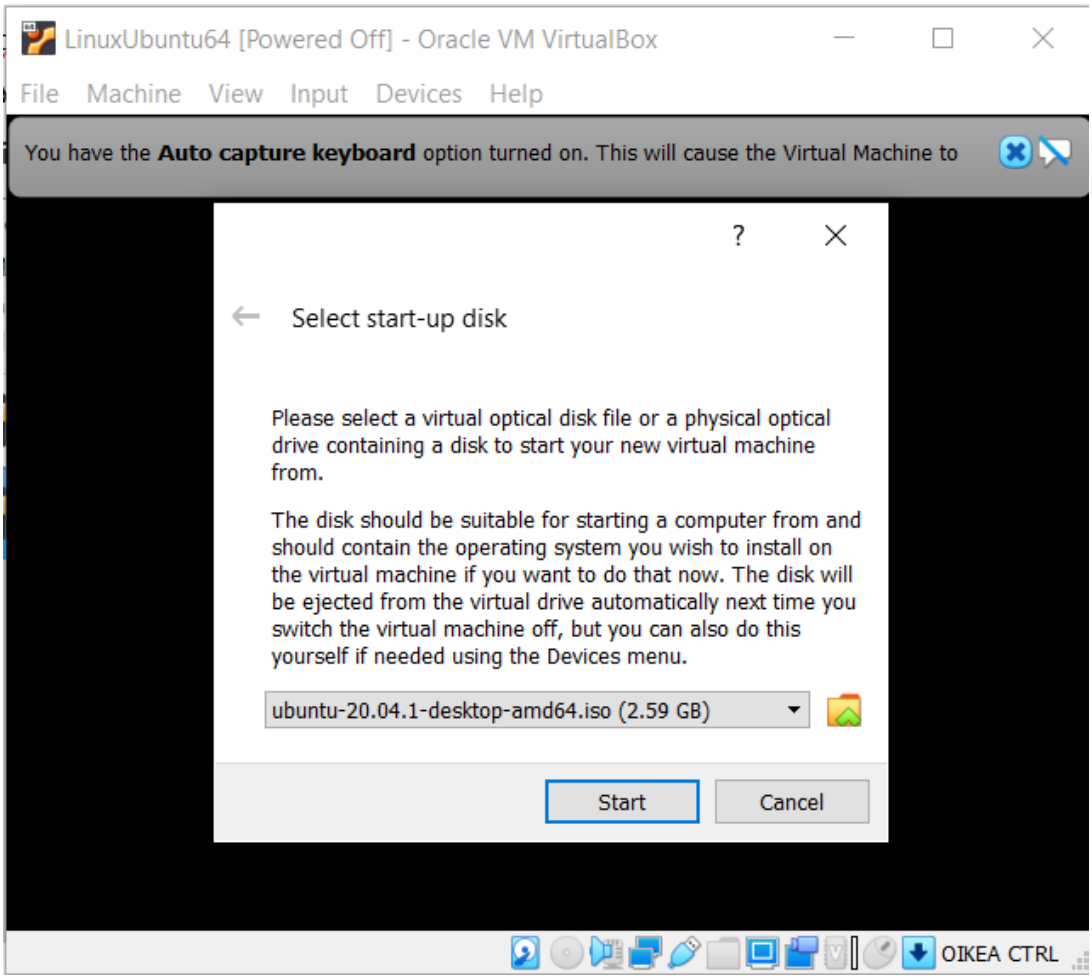


Click ”Create” → ”Next” → choose ”Fixed size” → ”Next” → allocate at least 10 GB of space for the virtual hard disk and click “Create”.

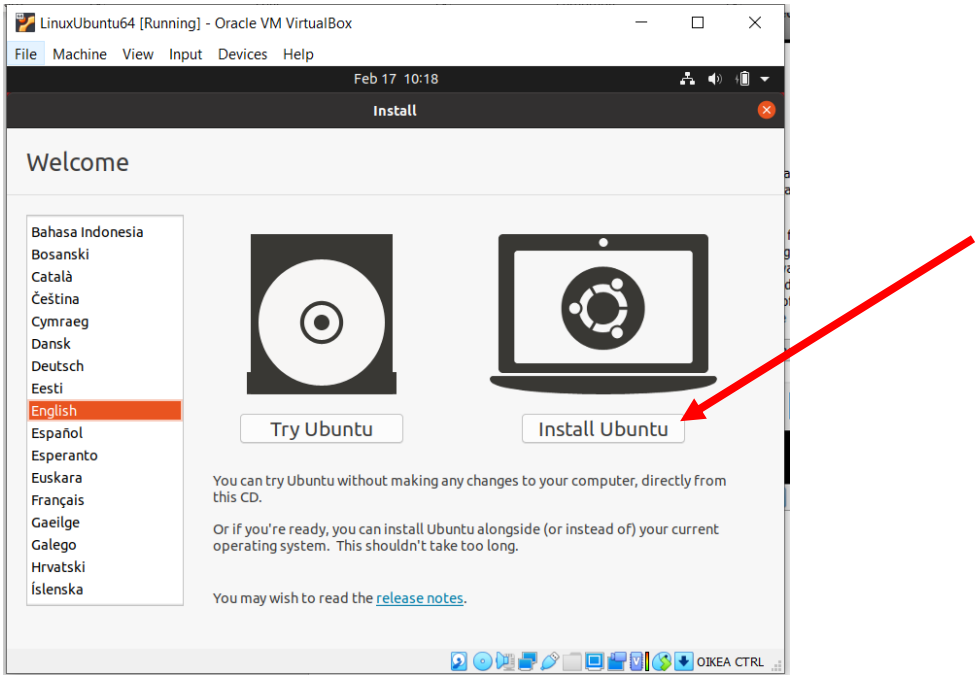
Your virtual machine should now be ready for operation. Start it by pressing “Start” button from the top:



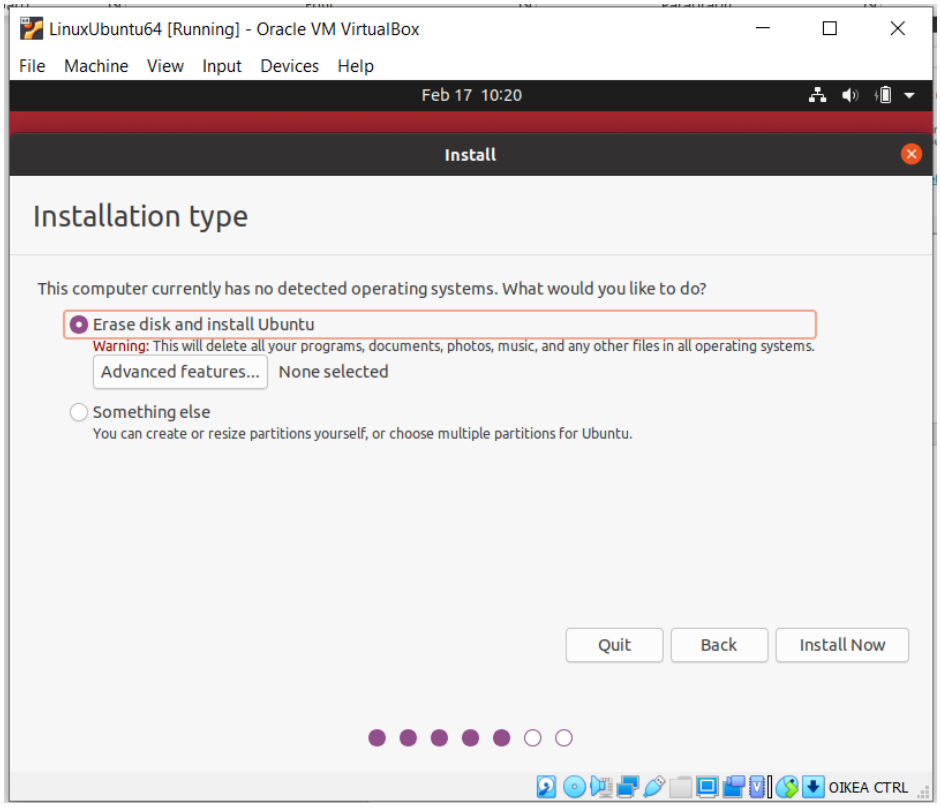
When booted for the first time, the machine asks for a start-up disk. Select the Ubuntu .iso file you downloaded and click ”Start”.



During the installation, select “Install Ubuntu”:



Follow the installation guide until you reach the following stage:



Select ”Erase disk and install Ubuntu”. (NOTE: Because we’re installing this on a virtual machine, this only formats the previously configured virtual hard disk; there’s no danger that you’d lose the files located on your actual hard disks.)

About Assembly:

Let’s clarify the terminology a bit first.

- Assembly = some lines of code/executable program, which is on a language that the machine understands and hence is able to run it (for example, open some .exe program using a text editor like Notepad in order to see what this looks like)
- Assembly language = symbolic programming language, which can be translated to machine language
- Assembler = Translator which assembles the program written on assembly language to an assembly

In this case gcc uses GNU Assembler (known as “GAS” for short) in the background for the translation process.

Program written on assembly language has been divided to two parts: .text and .data. Data part is where all the variables, their types and initial values are stored in. (In this case .quad GAS-directive means a 64-bit integer and .ascii a string of characters. Text part is where all assembly commands and program blocks (their names) go. The program is executed row

after row starting from up to down and from left to right, if there are no jumps, but the order of data and text parts doesn't matter.

Commands can have 0 to 3 parameters; for example, command ”movq $10, %rax” has two parameters, but command ”decq $10” has only one. The ”q” added in the end of the command means, that the operands must be 64-bit. Registers that start with letter ”r” are 64-bit registers. In assembly language, variables are simply memory addresses that have been given names. Variables must be presented in data part before they can be used. Variables are only referred to by their names.

Practically, programs can't change (at least in currently used OS's) things from anywhere else except virtual memory spaces that are shared with them. Programs can't therefore ask the system or device drivers to execute commands. What they can do is ask for the OS kernel to execute certain tasks via system calls. The OS checks whether the task requested by the program is allowed, and if it is, asks the corresponding device driver to perform the task. In this assembly language we can perform system calls using the command “syscall”. System calls to Linux 64-bit x86-64 ISA have been listed for example in the link below:

https://chromium.googlesource.com/chromiumos/docs/+/master/constants/syscalls.md#x86_64-64_bit

Linux Ubuntu's terminal is bash (Bourne-again shell) -based. When performing the task in exercise 7, the following commands may be needed:

- cd (change directory; for example: cd /home/your_username/scripts)
- mkdir (make new directory; for example: mkdir new_folder)
- touch (used for creating a new file; for example: touch hello_world.s)
- gedit (opens a file in gedit text editor; for example: gedit hello_world.s)
- man (opens Linux manual; here you can find help for commands, if you don't know how to use them – for example: man mkdir)

Linux manual can be also found in web form here:

https://man7.org/linux/man-pages/man1/man.1.html

Dollar sign is used for referring to the address of the variable (not the value itself!). More information can be easily found by googling “x86-64 GNU Assembly”. One good link for introduction here:

https://software.intel.com/content/www/us/en/develop/articles/introduction-to-x64-assembly.html