ECEN 749 Microprocessor System Design

HW 4 - Linux kernel built in device drivers

Due: 4/28/2020 @ 11:59pm

Presetup: Installing and starting a Ubuntu Linux system in Virtualbox

1. Download VirtualBox from www.virtualbox.org and install it on your machine.
2. Download "ECEN449-Student.zip" development platform from following source :

   Dropbox Link : https://www.dropbox.com/s/oz2m1zvlr1zma37/ECEN449-Student.zip?dl=1

   You can also build up your own Ubuntu Virtual Machine using steps outlined in Appendix at the end of the document.

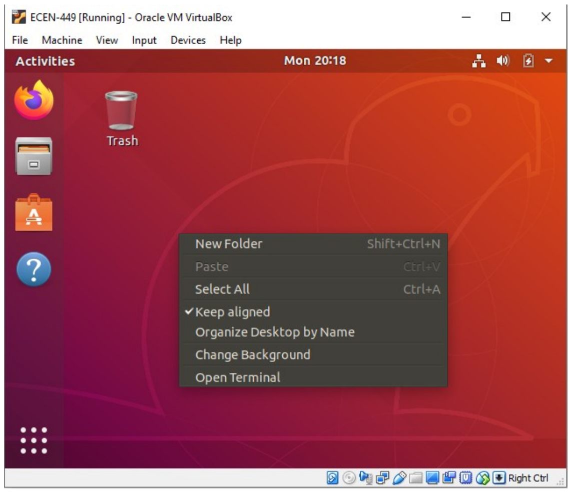
3. Uncompress the folder and click on ECEN449-Student.vbox (VirtualBox Machine Definition).
4. Click on the machine on the left and press start on the top of the screen. This should take you to a fully set up Ubuntu based Virtual Machine development platform.

   For sudo permissions, following is the login details for the given Ubuntu system.

   Username : student
   Password : 123456

PART 1 : Compiling and Installing Linux-4.20.17 in Ubuntu based Virtual Machine

1. Boot the Ubuntu system as described above and when the gui starts, right click anywhere on desktop screen to select and open terminal -



2. Run following command to identify the version of your Linux kernel currently running

```
$ uname -r
```

3. Extract the tarball archive present in your home directory for the Linux that we are going to compile and install ourselves

```
$ tar xf linux-4.20.17.tar.gz
```

4. Go To Linux directory

```
$ cd linux-4.20.17
```

5. Run the following command, to select and remove a few bigger kernel modules which we don't require.

```
$ make ARCH=x86 menuconfig
```

   Scroll down to Device Drivers and press enter. Remove multimedia, soundcard, from drivers by pressing the "SPACEBAR" twice (or until the symbol to the left of the driver name looks like < >). After making changes, save and exit back to the terminal.

6. Compile and Install the Linux-4.20.17.

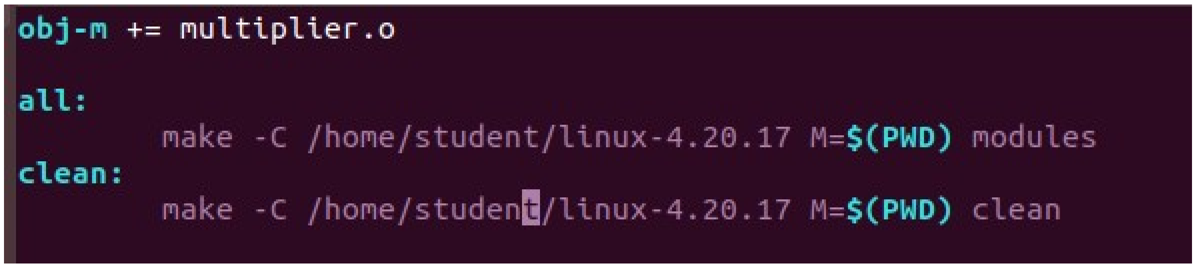
```
$ sudo make ARCH=x86 -j 4
$ sudo make ARCH=x86 modules_install -j 4
$ sudo make ARCH=x86 install -j 4
```

   Note the size of your Linux Image, and take a screenshot of terminal feed after successful compilation for your report.
   (Number "4" represents the number of allocated cores, adjust according to your system configuration)
   Now, Similar to what you did in Lab 5 and 6, you shall write a kernel module in C/C++ in PART 2, compile it, and then insert as Loadable Kernel Module dynamically to a running Linux-4.20.17 (achieved in PART 3) in PART 4. In PART 5, this kernel module would be built-in Linux-4.20.17 kernel and would be automatically inserted as a part of Linux installation process.

PART 2 : Creating a Kernel Module for Multiplication and User Application

1. Create a directory outside the linux source directory named "multiplier" where you will place your device driver source files.

2. The Makefile is identical to the one used in Lab 5. Remember to include the correct path to the kernel directory.



```
obj-m += multiplier.o

all:
	make -C /home/student/linux-4.20.17 M=$(PWD) modules
clean:
	make -C /home/student/linux-4.20.17 M=$(PWD) clean
```

3. As we don't have the access to a hardware Multiplier IP, we will go with a software implementation.

4. Write a kernel module – multiplier.c and multiplier.h to implement the software multiplication by following the character device driver example given in Lab 6. Modify your code to remove all references to a hardware IP multiplier, virtual and physical memory mapping/unmapping, and associated functions like iowrite/ioread/xparameters.

5. You should modify your "device_write" to do following :

   a) Read in two integers, A & B (8 bytes in total, length would be passed by the application) from "user buffer" and place it in a local buffer "msg_buffer" (Bytes 0-7).

   b) Multiply A & B to get "mult_result" and put it in msg_buffer starting at byte location 8.

      *" mult_result = A * B"*

   c) Return number of bytes read from "device driver file" (It should be 8).

6. Modify the "device_read function to do the following :

   a) Read bytes of data from the "msg_buffer" and put it in "user buffer". User application will pass how many bytes to be read and "user buffer file handle" (It should be 12, for reading 3 integers - A, B and mult_result).

   b) Return total number of bytes read (It should be 12).

7. To compile the kernel module, execute the following command in the multiplier directory.

   ```
   $ make
   ```

   If successful, you should be able to see a multiplier.ko file in the same folder.

8. Write a devtest application similar to interact with the multiplier kernel module similar to lab 6 and use following system calls

   a) open() - creates file handle, fd

   b) write() - write two integers, 8 bytes to user device driver file (user buffer)

   c) read() - read already written 2 integers and resultant multiplication output (12 bytes).

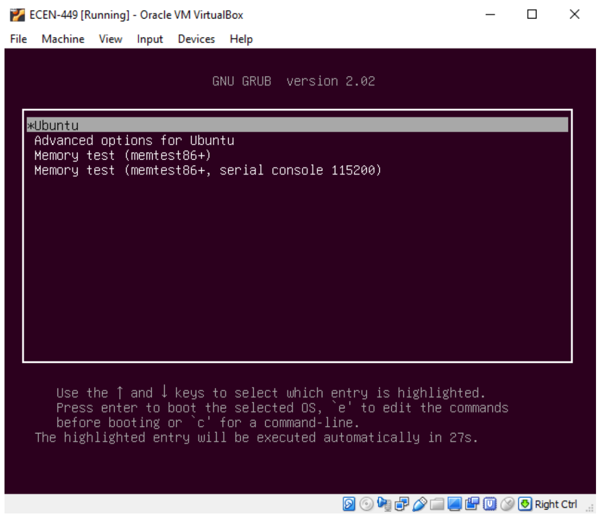
d) close() - closes the device file

Use two for loops to multiply every possible combination of two numbers in set {0- 16}. Compile devtest using:

```
$ gcc devtest.c -o devtest
```

PART 3: Booting up the new Linux-4.20.17

1. Reboot your UBUNTU VM and hold “SHIFT” key to access the GRUB bootloader menu (If not able to, press F12 on bootup followed by “b”), which will take you to a screen like this –

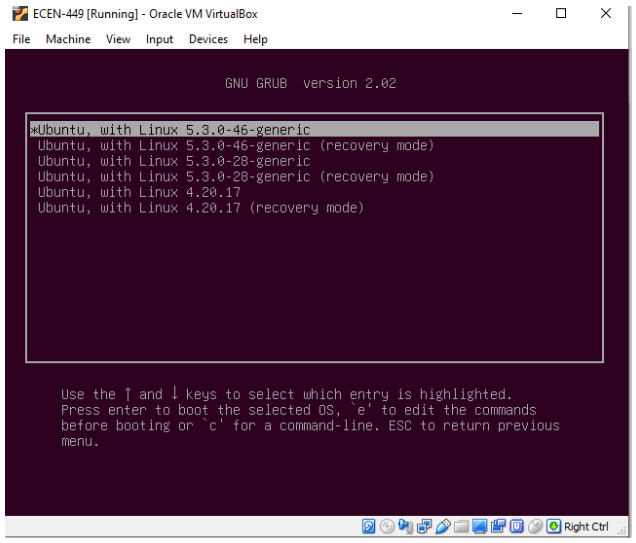

ECEN-449 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Right Ctrl

```
GNU GRUB  version 2.02

*Ubuntu
 Advanced options for Ubuntu
 Memory test (memtest86+)
 Memory test (memtest86+, serial console 115200)

   Use the ↑ and ↓ keys to select which entry is highlighted.
   Press enter to boot the selected OS, `e' to edit the commands
   before booting or `c' for a command-line.
The highlighted entry will be executed automatically in 27s.
```

2. Choose “Advanced options for Ubuntu”, and you will be directed to a menu with the list of kernels available (by default, as well as custom ones you have installed).


ECEN-449 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Right Ctrl

```
GNU GRUB  version 2.02

*Ubuntu, with Linux 5.3.0-46-generic
 Ubuntu, with Linux 5.3.0-46-generic (recovery mode)
 Ubuntu, with Linux 5.3.0-28-generic
 Ubuntu, with Linux 5.3.0-28-generic (recovery mode)
 Ubuntu, with Linux 4.20.17
 Ubuntu, with Linux 4.20.17 (recovery mode)

   Use the ↑ and ↓ keys to select which entry is highlighted.
   Press enter to boot the selected OS, `e' to edit the commands
   before booting or `c' for a command-line. ESC to return previous
   menu.
```

3. You can go ahead and choose Ubuntu, with Linux 4.20.17 which will boot you into our desired kernel version. All your previous file hierarchy will be intact and .ko and devtest application would be visible.

4. Run following to check your current Linux build

   ```
   $ uname -r
   ```

PART 4: Inserting a loadable kernel module (LKM) post Linux installation and boot-up

In this exercise, we will insert a pre compiled kernel module into already installed and running Linux-4.20.17 OS. This is an example of a loadable kernel module which can be inserted and removed as per the need.

1. Go to the “multiplier” directory created earlier and insert LKM using “insmod” command

   ```
   $ sudo insmod multiplier.ko
   ```

2. “lsmod” command will confirm that multiplier.ko has been installed

   ```
   $ lsmod | grep multiplier
   ```

3. Check the Linux messages to see the major number and other information from the bootup using

   ```
   $ dmesg | tail
   ```

4. Take a screenshot of step 3. Then, create the device driver file - “sudo mknod /dev/multiplier c %major# %minor#”

   ```
   $ sudo mknod /dev/multiplier c 240 0
   ```

5. To make it possible for a user application to read from or write to the device, do the following.

   ```
   $ sudo chmod 666 /dev/multiplier
   ```

6. Run the devtest to check the intended function

   ```
   $ ./devtest
   ```

7. Take a screenshot of the correct devtest output. Then, remove the multiplier module

   ```
   $ sudo rm /dev/multiplier
   $ sudo rmmod multiplier
   ```

8. Check the removal by lsmod command again

   ```
   $ lsmod | grep multiplier
   ```

9. If you need to update and re-compile your multiplier module or application, it can be done from Linux-4.20.17 boot up as well. First remove your module, change, compile and insert it again.

PART 5: Built in module

1. Boot in the original Ubuntu 5.30 mode again.

2. Create a directory called “multiplier_driver” under linux-4.20.17/drivers/ and copy multiplier kernel module source files - multiplier.c and multiplier.h into it

3. Use a text editor to create a ‘Makefile’ in the ‘multiplier_driver’ directory and add the following line to it.

```
obj-$(CONFIG_MULTIPLIER_DRIVER) += multiplier.o
```

4. Use a text editor to create a ‘Kconfig’ file in the ‘multiplier_driver’ directory and add the following lines to it.

```
config MULTIPLIER_DRIVER
tristate “multiplier_driver”
depends on X86
default y if X86
help
 refer to ECEN449@TAMU
```

5. In the Kconfig file, every line starts with a key word and can be followed by multiple arguments. ‘config’ starts a new config entry. The following lines define attributes for this config option. Attributes can be the type of config option, input prompt, dependencies, help text, and default values. A config option can be defined multiple times with the same name, but every definition can have only a single input prompt, and the type must not conflict.

6. In the above Kconfig entry, the first line defines what configuration option this entry represents. Note that the CONFIG prefix is assumed and not written.

7. The second line states that this option is a tristate, meaning that it can be built into the kernel (Y), built as a module (M), or not built at all (N). The quoted text following the directive provides the name of this option in the various configuration utilities.

8. The third line specifies the dependency is X86 architecture, as we have developed this driver for an X86 device.

9. The fourth line specifies the default for this option, which is ‘y’ if the architecture is x86.

10. The help directive signifies that the rest of the text, indented as it is, is the help text for this entry

11. The multiplier driver should be listed under the Device Drivers menu entry. In the Device Drivers directory under the Linux source file, find the Makefile and add the following line

to it.

```
# ECEN 449
obj-$(CONFIG_MULTIPLIER_DRIVER) += multiplier_driver/
```

12. Open the Kconfig file under Device Drivers and add the following lines before the ‘endmenu’.

    ```
    source “drivers/multiplier_driver/Kconfig”
    ```

13. Navigate to the Linux source directory and run the following command.

    ```
    $ make ARCH=x86 menuconfig
    ```

    You should be able to see an entry for the multiplier driver under Device Drivers in menuconfig. Select it to be compiled as built-in. (Press SPACEBAR to make [*]) Take a screenshot of this.

14. Compile and Install the Linux-4.20.17

    ```
    $ sudo make clean

    $ sudo make ARCH=x86 -j 4 && sudo make ARCH=x86 modules_install -
    j 4 && sudo make ARCH=x86 install -j 4
    ```

15. Reboot your VM and select the Linux-4.20.17 distribution as shown in PART 3. This time we already have a multiplier module as part of the Linux kernel and there is no need for an explicit insertion. Check using the “dmesg” command and then search for “multiplier”. You can search on the terminal(Shift+Ctrl+f). Take a screenshot of the compilation messages showing major number.

    There you will find the <Major Number> with which you need to register -

    ```
    $ sudo mknod /dev/multiplier c <Major Number> 0
    $ sudo chmod 666 /dev/multiplier
    ```

16. Run the devtest to check the functionality of multiplier. Use “`sudo ./devtest`”. Take a screenshot of correct output.

17. At this point, you should have a Linux kernel with built-in support for the multiplier driver. Note that the size of the uImage increases as you have added more built-in drivers. We have included device drivers to support our applications ‘multiply’ and ‘ir demod’. Similarly, we can exclude features which we do not require, which will reduce the kernel size. We should only exclude features that are not critical to the Linux boot, otherwise the Linux kernel may not function properly or even cause a boot failure.

Deliverables:

1. Assignment Report PDF detailing the Objective, Process, Result and Conclusion.

2. Following screenshots should be part of the report as a proof that

a) Linux 4.20.17 compiled successfully on Ubuntu VM. (At Least One Screenshot taken at the end of compilation message).

b) Insertion of Loadable Kernel Module was achieved ("lsmod" command output) and the user application runs with intended multiplication output. (Two screenshots)

c) Multiplier added to the Kernel build - menuconfig screen which shows "multiplier_driver" is included. (One screenshot)

d) Linux compiles successfully with the multiplier_driver built-in and the user application runs and produces the output. (Two Screenshots)

3. Source Code should be attached separately - "multiplier.c , multiplier.h, devtest.c"

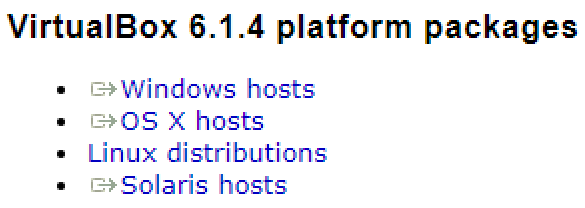
Answer the following questions and include at the end of your report.

1. What are the advantages and disadvantages of both the loadable kernel modules and built-in module approach?
2. Describe both one good thing and one thing that can be improved about your experience in the labs this semester.

APPENDIX : Setting Up the Virtual Box with Ubuntu.

1. Download Virtual Box from the website - https://www.virtualbox.org/wiki/Downloads

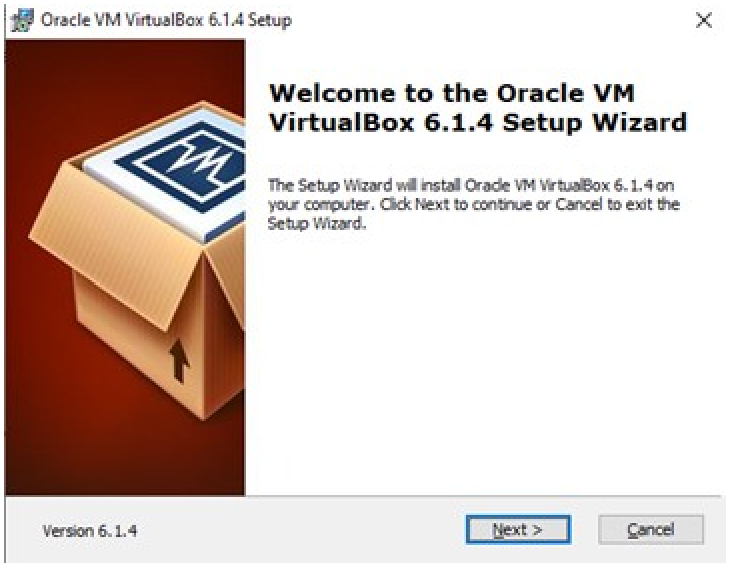
Select “Windows hosts” if you have a Windows PC or “OS X hosts” if you have a Mac.



VirtualBox 6.1.4 platform packages

- ⇨Windows hosts
- ⇨OS X hosts
- Linux distributions
- ⇨Solaris hosts

2. This will download an installer. Click on the installer which should open a dialog box like this:



Oracle VM VirtualBox 6.1.4 Setup

Welcome to the Oracle VM VirtualBox 6.1.4 Setup Wizard

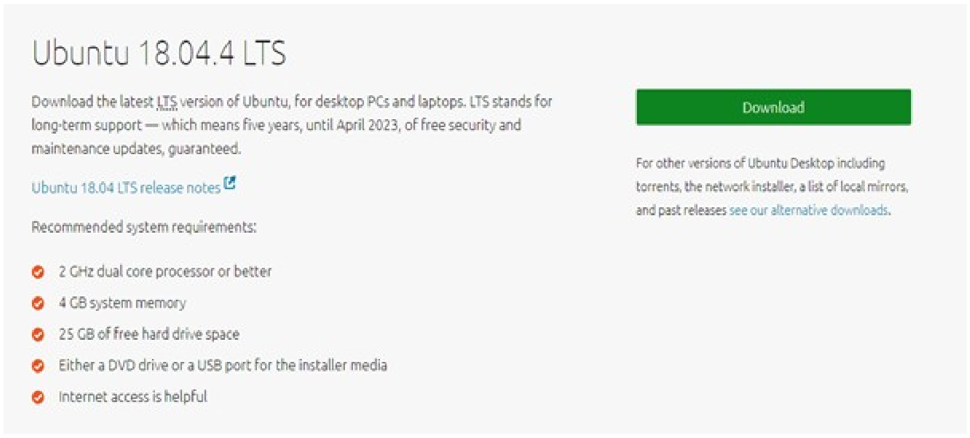
The Setup Wizard will install Oracle VM VirtualBox 6.1.4 on your computer. Click Next to continue or Cancel to exit the Setup Wizard.

Version 6.1.4

3. This is a very simple installation. Keep clicking “Next” and then “Install”. After this the installation will finish.

4. Now, we need to download a Linux distribution. We are going to use Ubuntu 18.04 for this. Go to the link - https://ubuntu.com/download/desktop

And then download the mentioned version:



Ubuntu 18.04.4 LTS

Download the latest LTS version of Ubuntu, for desktop PCs and laptops. LTS stands for long-term support — which means five years, until April 2023, of free security and maintenance updates, guaranteed.

Ubuntu 18.04 LTS release notes

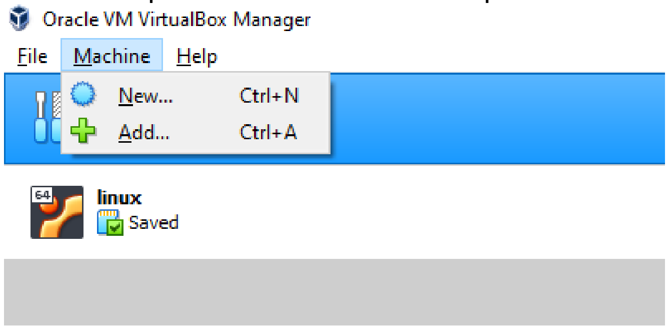
Recommended system requirements:

- 2 GHz dual core processor or better
- 4 GB system memory
- 25 GB of free hard drive space
- Either a DVD drive or a USB port for the installer media
- Internet access is helpful

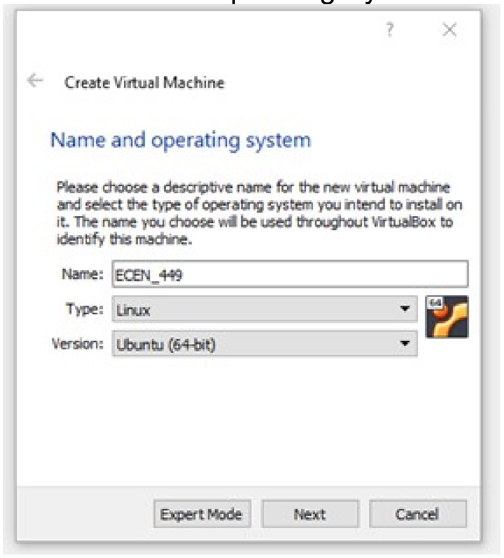
For other versions of Ubuntu Desktop including torrents, the network installer, a list of local mirrors, and past releases see our alternative downloads.

This will download a disk image file. It is around 2GB in size and hence may take some time.

5. Now open Virtual Box. On the top left click on "Machine"->"New"



6. Name and Operating System



Create Virtual Machine

Name and operating system

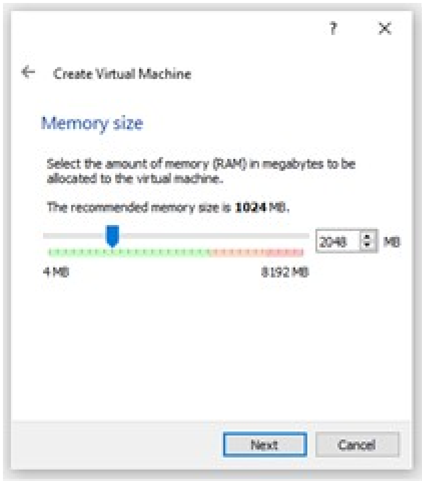
Please choose a descriptive name for the new virtual machine and select the type of operating system you intend to install on it. The name you choose will be used throughout VirtualBox to identify this machine.

Name: ECEN_449

Type: Linux

Version: Ubuntu (64-bit)

7. Select Memory Size, Virtual Hard Disk, Hard Disk type


Create Virtual Machine

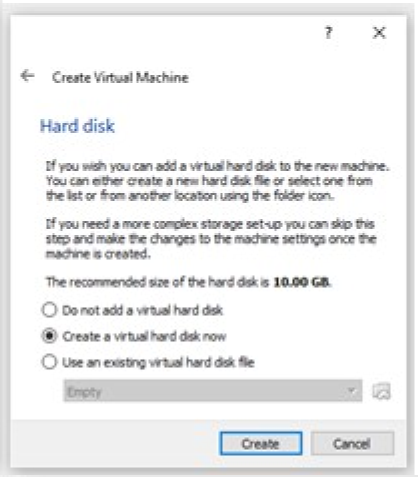
Memory size

Select the amount of memory (RAM) in megabytes to be allocated to the virtual machine.

The recommended memory size is **1024** MB.

2048 MB

4 MB 8192 MB


Create Virtual Machine

Hard disk

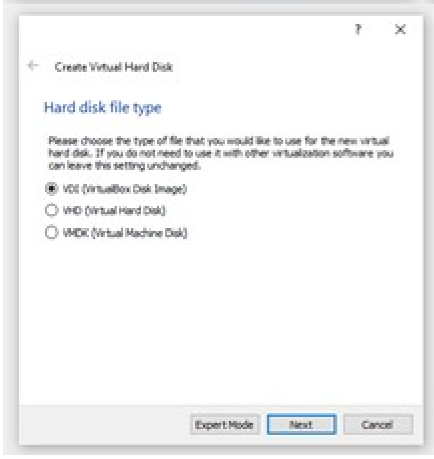
If you wish you can add a virtual hard disk to the new machine. You can either create a new hard disk file or select one from the list or from another location using the folder icon.

If you need a more complex storage set-up you can skip this step and make the changes to the machine settings once the machine is created.

The recommended size of the hard disk is **10.00 GB**.

- ○ Do not add a virtual hard disk
- ● Create a virtual hard disk now
- ○ Use an existing virtual hard disk file

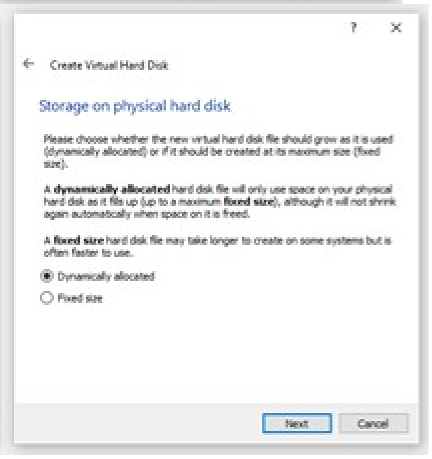
Empty


Create Virtual Hard Disk

Hard disk file type

Please choose the type of file that you would like to use for the new virtual hard disk. If you do not need to use it with other virtualization software you can leave this setting unchanged.

- ● VDI (VirtualBox Disk Image)
- ○ VHD (Virtual Hard Disk)
- ○ VMDK (Virtual Machine Disk)


Create Virtual Hard Disk

Storage on physical hard disk

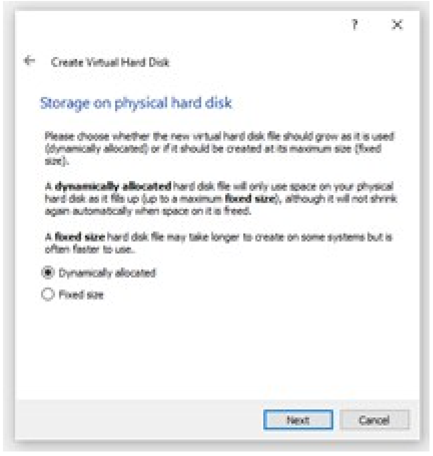
Please choose whether the new virtual hard disk file should grow as it is used (dynamically allocated) or if it should be created at its maximum size (fixed size).

A **dynamically allocated** hard disk file will only use space on your physical hard disk as it fills up (up to a maximum **fixed size**), although it will not shrink again automatically when space on it is freed.

A **fixed size** hard disk file may take longer to create on some systems but is often faster to use.

- ● Dynamically allocated
- ○ Fixed size

8. File location and size – Allocate at least 30 GB


Create Virtual Hard Disk

Storage on physical hard disk

Please choose whether the new virtual hard disk file should grow as it is used (dynamically allocated) or if it should be created at its maximum size (fixed size).

A **dynamically allocated** hard disk file will only use space on your physical hard disk as it fills up (up to a maximum **fixed size**), although it will not shrink again automatically when space on it is freed.

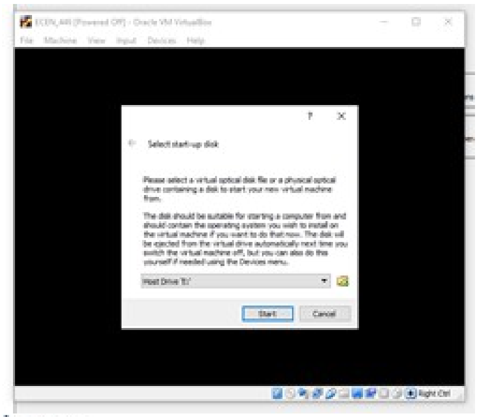
A **fixed size** hard disk file may take longer to create on some systems but is often faster to use.

- ● Dynamically allocated
- ○ Fixed size

9. Right Click on ECEN_449 VM and Press SETTINGS, In system select 4-6 cores and save.

10. Right Click on ECEN_449 and Press “START”.

11. Select start-up disk - select the installed UBUNTU and press “START”

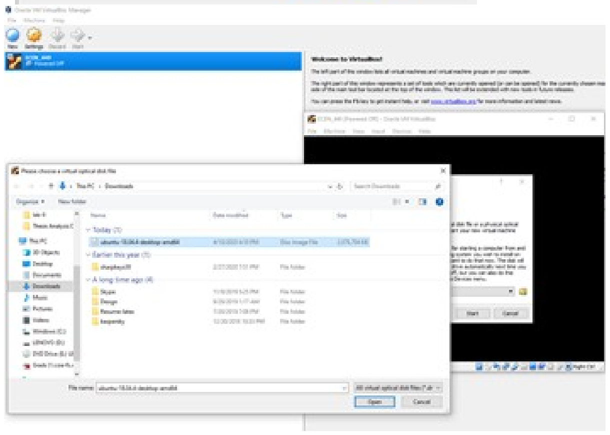


Select start-up disk

Please select a virtual optical disk file or a physical optical drive containing a disk to start your new virtual machine from.

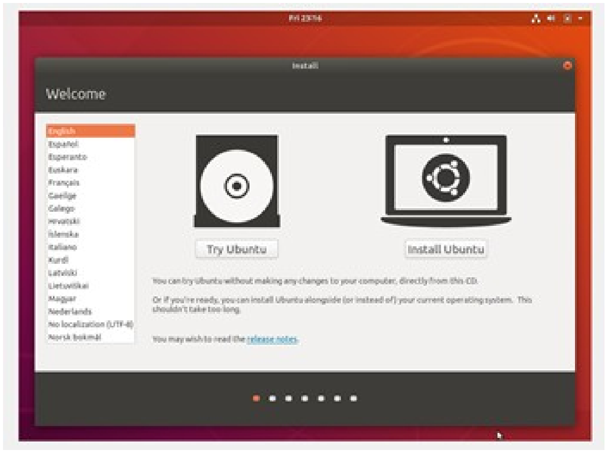
The disk should be suitable for starting a computer from and should contain the operating system you wish to install on the virtual machine if you want to do that now. The disk will be ejected from the virtual drive automatically next time you switch the virtual machine off, but you can also do this yourself if needed using the Devices menu.

Host Drive 'E:'



12. In the View Menu, adjust your screen resolution if unable to see the Virtual machine's UI properly.

13. Proceed and Install Ubuntu with default settings.



You can try Ubuntu without making any changes to your computer, directly from this CD.

Or if you're ready, you can install Ubuntu alongside (or instead of) your current operating system. This shouldn't take too long.

You may wish to read the release notes.

14. After Installation, Restart the system and press ENTER if asked.

15. Install the following packages and install any separately if error occurs

```
$ sudo apt-get update
$ sudo apt-get install git fakeroot build-essential ncurses-dev xz-
utils libssl-dev bc

$ sudo apt-get install vim

$ sudo apt-get install vim-gtk3

$ sudo apt-get install make

$ sudo apt-get install gcc

$ sudo apt-get install bison

$ sudo apt-get install flex

$ sudo apt-get install libelf-dev
```

16. Download linux-4.20.17.tar.gz using following command

```
$wget https://www.kernel.org/pub/linux/kernel/v4.x/linux-
4.20.17.tar.gz
```