As usual, this is also a group project.

3 people per group.

Email me (rahman@cs.fsu.edu) the groups no later than Sept 30.

For this project, we are using linux kernel version 5.3.0.

Download it from linux kernel archive (https://www.kernel.org/)

Here is a direct link to the tarball: https://cdn.kernel.org/pub/linux/kernel/v5.x/linux-5.3.tar.xz

You need to do it in your own laptop.

If you already have a linux OS installed (say, Ubuntu), you can install the kernel directly in it.

This may give you better performance in terms of speed.

However, we are going to modify the kernel in this project. And it’s very easy to corrupt the OS in the process, and you may not be able to boot.

A safer option is using a Virtual Machine.

Using a VM allows you to quickly recover from kernel errors by using a snapshot (Important, do take system snapshots at regular interval).

VirtualBox is a good one, you can use something else if you prefer.

Download a guest OS image (This is a good source: https://www.osboxes.org/virtualbox-images/)

Create a new VM using the downloaded image as the hard-drive.

The more memory and CPU you can allocate to the VM, the faster the kernel will compile.

You may consider giving it half of the available memory, and half of the processor cores (if you have multiple cores).

Ok, now let’s compile a kernel.

First, install some necessary packages:

```
$sudo apt-get update

$sudo apt-get install build-
essential libncurses-dev
bison flex libssl-dev libelf-
dev
```

For this project, we are using linux kernel version 5.3.0.

Download it from linux kernel archive (https://www.kernel.org/)

Here is a direct link to the tarball: https://cdn.kernel.org/pub/linux/kernel/v5.x/linux-5.3.tar.xzDownload the linux kernel

Extract it in your home directory
Rename the directory to test_kernel
(Optional, you can choose to keep it as linux-5.3)

Add a symbolic link to your kernel directory from /usr/src.

This will make our job easier later when we add our own kernel modules.

```
$sudo ln -s ~/test_kernel /usr/src
```

Now, let's compile the kernel.

Before that, check your own kernel version using `uname -r`

First, you need to configure which modules are to be included in the kernel and which not.

```
$make menuconfig
```

This will bring a graphical window and you can decide which modules to include.

You may save a lot of compiling time by removing unnecessary device drivers and file systems.

But, this is time-consuming, and potentially risky, given that you may also remove something important and end up with a broken kernel.

A safer alternative can be to use your existing old, working kernel.

```
$ cd /usr/src/test_kernel
$ cp .config .config_old
$ cp /boot/config-5.0.0-23-generic .config
$ make oldconfig
```

Here, `config-5.0.0-23-generic` is my existing config. Yours can be named different. Check in your /boot directory.

Accept the changes that may be prompted by make oldconfig. This sets up an working configuration for you.

```
$ make menuconfig
```

Graphical configuration setup. Save and exit.

```
$ make
```

Compiles the kernel. Depending on your system and configuration, can take HOURS!

If you have multiple cores in your machine / VM, parallelize it by

```
$ make -j $(nproc)
$ sudo make modules_install
```

Install module binaries into modules

```
$ sudo make install
```

Installs final binary into /boot. (Phew!)

In Debian/Ubuntu, the following commands are optional as make install does everything for you. But if there was an error in updating the boot-loader, then you may have to do it manually.

```
$ sudo update-initramfs -c -k 5.3.0
$ sudo update-grub
```

Finally, reboot your system.

```
$ shutdown -r 0
```

Once it boots up, you should be in the new kernel. Check it yourself.

```
$ uname -r
```

At “make install”, if you see an error message like this: “gzip: stdout: No space left on device”, it means your /boot partition is full.

In that case, you need to delete some existing images from the /boot partition and then try again.

You should be able to list your installed kernels with

```
$ aptitude search ~ilinux-image
```

You can remove the old kernels, one by one, like this:

```
$sudo apt-get autoremove linux-image-3.2.0-23-generic
```

Make sure you substitute "3.2.0-23-generic" with the actual kernel version you want to remove.