3. KERNEL COMPILATION

In this chapter we are going to explain about the compilation process that is to be done.

3.1 DEVELOPING A (.config) FILE:

The .config file is developed after defining which drivers are to enabled as modules during the process of compilation. The configuration file is used to configure the parameters and initialise the settings of device drivers for the kernel. To generate a .config file we have to follow the following commands.We can generate a .config file using several make <option>config commands.The most commonly used commands are:

make menuconfig -

This allows the user to configure in the text editor mode with low GUI. After giving this command a dialog box will be displayed which shows us to enable or disable the drivers. We can enable the drivers as two types, one by giving it as an inbuilt thing and the second is by giving it as a module thing. To enable it as a module we have to give **[M]**, to enable it as an inbuilt feature we have to give **[*]**, if we leave it blank **[ ]** then it represents the driver is not initialised.

make xconfig -

This allows the user to configure by selecting the boxes which is provided with more GUI. By selecting the required box it enables the user to enable the feature.

make defconfig -

This allows the user to generate a configuration file with default configuration.

3.2 COMPILING THE KEREL USING "make":

make is a build animation tool that automatically builds executable programs and libraries from source code by reading files called Makefiles which specify how to derive the target program. Make can also be used to manage any project where some files must be updated automatically from others whenever the others change. After running the command **make menuconfig** and generating a **.config** file we have to run the command **make** so that the **'.c'** files are compiled to **'.o'** to produce a build executable program.

3.3 INSTALLING THE MODULES :

After running the **make** command we have to run the command **sudo make modules_install** so that the modules of necessary drivers that are enabled are installed. After this we have to give the command **sudo make install** which generates a kernel version that is compiled. **make install** invokes make, make finds the target install in Makefile and files the directions to install the program.

3.4 GENERATING THE INITRDIMG:

To generate the initrdimg we have to give the following command **sudo mkinitramfs -o /boot/initrdimg- (kernel version). mkinitramfs** is a low-level tool for generating an initramfs image, it is used to mount the root file system. **initrdimg** is an initial root file system that is the mounted prior to when the real file system is available. **initrd** is bound to kernel and loaded as a part of kernel boot procedure. It produces the capability to lead the RAM disk by the boot loader.

3.5 UPDATING THE GRUB AND RUNNING THE KERNEL:

After generating the initrdimg we have to update the grub by using the command **sudo update-grub** so that the generated new version is loaded at the process of starting in the grub boot loader. GNU GRUB (GNU GRand Unified Bootloader) is a boot loader is the reference implementation of the Free Software Foundation's Multiboot Specification, which provides a user the choice to boot one of multiple operating systems installed on a computer or select a specific kernel configuration available on a particular operating system's partitions. We can view the booted kernel in the terminal by using the command **"uname -a".**