**Ex 2 KERNEL CONFIGURATION AND COMPILATION**

**Date: 25.08.20**

**Aim:**

To study and implement the kernel configuration and compilation.

**Description:**

**The Linux Kernel :**

The main purpose of a computer is to run a predefined sequence of instructions,

known as a program. A program under execution is often referred to as a process.

Now, most special purpose computers are meant to run a single process, but in a

sophisticated system such a general purpose computer, are intended to run many

processes simultaneously. Any kind of process requires hardware resources such are

Memory, Processor time, Storage space, etc.

In a General Purpose Computer running many processes simultaneously, we need a

middle layer to manage the distribution of the hardware resources of the computer

efficiently and fairly among all the various processes running on the computer. This

the middle layer is referred to as the kernel. Basically the kernel virtualizes the common

hardware resources of the computer to provide each process with its own virtual

resources. This makes the process seem as it is the sole process running on the

machine. The kernel is also responsible for preventing and mitigating conflicts

between different processes.

The Core Subsystems of the Linux Kernel are as follows:

1. The Process Scheduler

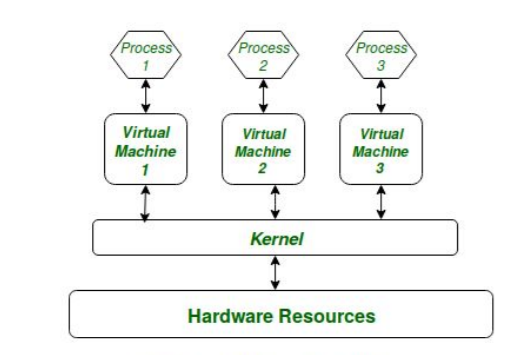
2. The Memory Management Unit (MMU)

3. The Virtual File System (VFS)

4. The Networking Unit

5. Inter-Process Communication Unit

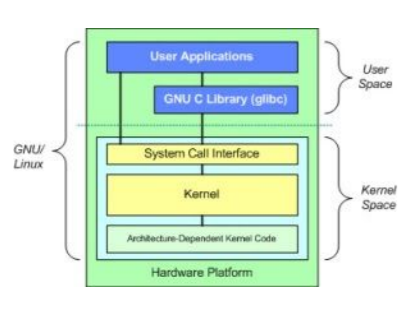
This schematically represented below:



**Architecture of system kernel :**

We can think of Linux Kernel architecture to be divided into two levels – User Space

and Kernel Space.



At the top is the user space. Below the user space is the kernel space. Here, the Linux kernel exists.

User Space: This is where the user applications are executed. There is also the GNU C Library (glibc). This provides the system call interface that connects to the kernel and provides the mechanism to transition between the user-space application and the kernel.

**Exercise**

**Configure and compilation the kernel**

**Step 1 :** Download the latest kernel source from www.kernel.org or from a repository.

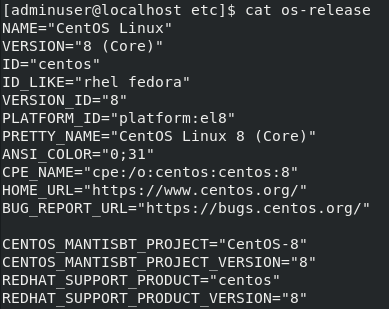
**Step 2 :** Check the current kernel version and name of the kernel.

**Step 3 :** Move the module from downloads to /usr/src and uzip the file.

**Step 4 :** Make a systemlink to the existing kernel and clean the existing kernel.

**Output:**

Kernel Version



Kernel Name



**Results:**

The kernel configuration and compilation is studied and executed.

**Video :**