**Practical No:10**

Learn Linux Kernel with respect to:

1. What is Linux Kernel ?
2. Operating Model
3. Licensing Model
4. How development works ?

****

**Linux Kernel**

1. **What is Linux Kernel ?:**

* The Linux kernel is a monolithic Unix-like computer operating system kernel. The [Linux](https://en.wikipedia.org/wiki/Linux) family of operating systems is based on this kernel and deployed on both traditional computer systems such as [personal computers](https://en.wikipedia.org/wiki/Personal_computer) and [servers](https://en.wikipedia.org/wiki/Server_(computing)), usually in the form of [Linux distributions](https://en.wikipedia.org/wiki/Linux_distribution)

1. **Operating Model:**

* Kernel is the core part of Linux. It is responsible for all major activities of this operating system. It consists of various modules and it interacts directly with the underlying hardware. Kernel provides the required abstraction to hide low level hardware details to system or application programs.

1. **Licensing Model:**

* Initially, Torvalds released Linux under a license which forbade any commercial use.This was changed in version 0.12 by a switch to the [GNU General Public License](https://en.wikipedia.org/wiki/GNU_General_Public_License) (GPL).
* The Linux kernel is licensed explicitly only under version 2 of the GPL,without offering the licensee the option to choose "any later version", which is a common GPL extension. There was considerable debate about how easily the license could be changed to use later GPL versions (including version 3), and whether this change is even desirable.

1. **How development works ?:**

* Linux kernel development process currently consists of a few different main kernel “branches” and lots of different subsystem-specific kernel branches. These different branches are:

main 4.x kernel tree

4.x.y -stable kernel tree

4.x -git kernel patches

subsystem specific kernel trees and patches

the 4.x -next kernel tree for integration tests

* The kernel is written mostly in C, with some architecture-dependent parts written in assembly. A good understanding of C is required for kernel development. Assembly (any architecture) is not required unless you plan to do low-level development for that architecture. Though they are not a good substitute for a solid C education and/or years of experience, the following books are good for, if anything, reference:

1. “The C Programming Language” by Kernighan and Ritchie [Prentice Hall]
2. “Practical C Programming” by Steve Oualline [O’Reilly]
3. “C: A Reference Manual” by Harbison and Steele [Prentice Hall]