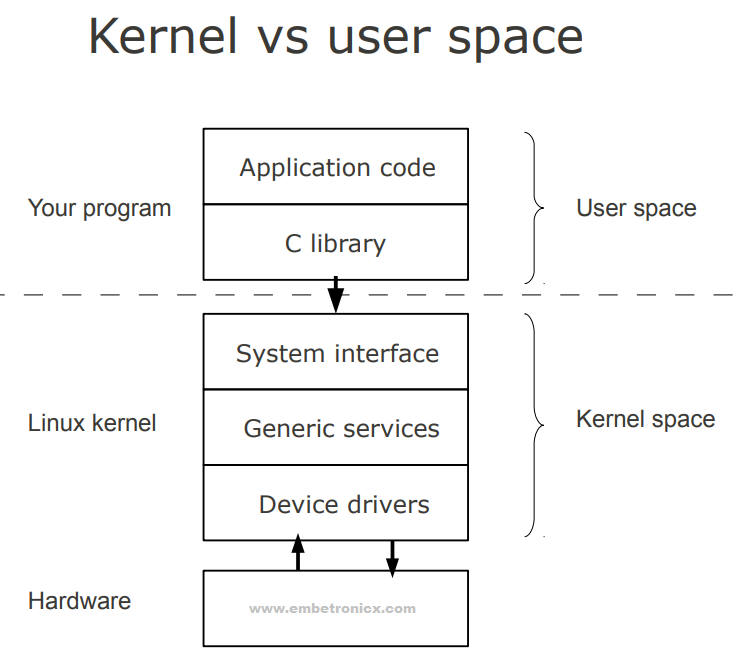
**Linux Architecture:-**

Linux is primarily divided into User Space & Kernel Space. These two components interact through a System Call Interface – which is a predefined and matured interface to Linux Kernel for Userspace applications.

**Kernel Space:-**

Kernel space is where the kernel (i.e., the core of the operating system) executes (i.e., runs) and provides its services.

**User Space: -** User Space is where the user applications are executed.



**Linux Kernel Modules:-**

Kernel modules are pieces of code that can be loaded and unloaded into the kernel upon demand. They extend the functionality of the kernel without the need to reboot the system.

Custom codes can be added to Linux kernels via two methods.

* **Static method:-** The basic way is to add the code to the kernel source tree and recompile the kernel.
* **Dynamic method:-** A more efficient way is to do this is by adding code to the kernel while it is running. This process is called loading the module, where the module refers to the code that we want to add to the kernel.

**Device Driver:-**

A device driver is a particular form of software application that is designed to enable interaction with hardware devices. Without the required device driver, the corresponding hardware device fails to work.

A device driver usually communicates with the hardware by means of the communications subsystem or computer bus to which the hardware is connected. Device drivers are operating system-specific and hardware-dependent. A device driver acts as a translator between the hardware device and the programs or operating systems that use it.

**Types:-** In the traditional classification, there are three kinds of the device:

* Character device
* Block device
* Network device

**Module Information:-**

**License:-**

GPL, or the GNU General Public License, is an open-source license meant for software. If your software is licensed under the terms of the GPL, it is free.

We can give the License for our Linux device driver (module) like below. For this, you need to include the Linux/module.h header file.

MODULE\_LICENSE("GPL");

MODULE\_LICENSE("GPL v2");

MODULE\_LICENSE("Dual BSD/GPL");

**Author:-**

Using this Macro we can mention that who is writing this Linux device driver or module. We can give the Author’s name for our driver (module) like below.

MODULE\_AUTHOR("Author");

**Module Description:-**

Using this Macro we can give a description of the module or Linux device driver. We can give the description for our driver (module) like below.

MODULE\_DESCRIPTION("A sample driver");

**Module Version:-**

Using this Macro we can give the version of the module or driver.

MODULE\_VERSION("2:1.0");

**Simple Kernel Module Programming:-**

In C Language, the starting point would be the main function and runs till end of the program. So similarlly here we have two separate functions used for that starting and ending.

**Init function:-**

This is the function that will execute first when the Linux device driver is loaded into the kernel. For example, when we load the driver using insmod, this function will execute.

**syntax:-**

static int \_\_init hello\_world\_init(void)

{

return 0;

}

module\_init(hello\_world\_init);

**Exit function:-**

This is the function that will execute last when the Linux device driver is unloaded from the kernel. For example, when we unload the driver using rmmod, this function will execute.

**syntax:-**

void \_\_exit hello\_world\_exit(void)

{

}

module\_exit(hello\_world\_exit);

**Simple Linux Device Driver:-**

#include<linux/kernel.h>

#include<linux/init.h>

#include<linux/module.h>

static int \_\_init hello\_world\_init(void)

{

printk(KERN\_INFO "Hello world\n");

printk(KERN\_INFO "This is the Simple Module\n");

printk(KERN\_INFO "Kernel Module Inserted Successfully...\n");

return 0;

}

static void \_\_exit hello\_world\_exit(void)

{

printk(KERN\_INFO "Kernel Module Removed Successfully...\n");

}

module\_init(hello\_world\_init);

module\_exit(hello\_world\_exit);

MODULE\_LICENSE("GPL");

MODULE\_AUTHOR("chandan");

MODULE\_DESCRIPTION("A simple hello world driver");

MODULE\_VERSION("2:1.0");

**Loading and Unloading the Linux Device driver:-**

**Loading:-**

To load a Kernel Module, use the “insmod” command with root privileges.

**Syntax:-**

sudo insmod kernel\_module\_name.ko

**Unloading:-**

To un-load, a Kernel module, use the “rmmod” command with root privileges.

**Syntax:-**

sudo rmmod kernel\_module\_name.ko

**Listing the Modules:-**

In order to see the list of currently loaded modules, use the “lsmod” command.

**Getting Module Details:-**

In order to get information about a Module (author, supported options), we may use the modinfo command.

For example: modinfo kernel\_module\_name.ko