Linux device driver program:-

This program explain about the basics of Linux device driver, Hello World Program and How to setup the environment,
Linux OS device drivers are also known as kernel module
Here I have mentioned the steps I followed to run this program.

Step 1:-
Setup Environment to Run linux device driver hello world program before
installing Linux or ubuntu os using this link (https://ubuntu.com/download/desktop),

Step 2:-
After successful OS installation give **CTRL+ALT+T** it will show terminal and if you want another tab in same terminal use **CTRL+shift key+T** . Then type this command **(uname -r)** to know what type of kernel version is running in os **(ex: 6.5.0-15-generic)** and type (sudo apt-get install build-essential) it help with basic application that are useful to create basic linux device drivers, Now the laptop would be ready to run program.

Step 3:-
We need to know some important kernel structure for the program

- Header Files
- Global Variables
- Init Functions
- Exit Functions
- Kernel Macros

Header files:-

We should know some header files to run our basic module (device driver) program,

1. **#include <linux/init.h>**
   init.h is one of the kernel's built-in headers, the ones used to actually build kernels from source. It isn't included in the exported headers that go into /usr/include/linux. Many distros provide these kernel headers as a separate package, since people need them for building proprietary driver modules

2. **#include <linux/kernel.h>**
   The Linux kernel is the Linux-basic operating system and interface between hardware and the computer processes. A kernel is a special program responsible for managing the low-level functions of a computer.

3. **#include <linux/module.h>**
   Kernel modules are pieces of code that Linux can load into the kernel to add features like device drivers, file systems, or system calls

Follow the path that will reach the kernel header files.
to check the kernel versions use **uname -r** we will get terminal like this **(ex:6.5.0-15-generic)**
And type the below path.
/lib/modules/(copy and past it your kernel version here)/build/include/linux

Example:-
/lib/modules/6.5.0-15-generic/build/include/linux

Global Variables:-

Global variables is optional to use in program in future program i will give some example program for reference

Init Functions:-

In Linux if device drivers are built as loadable kernel modules, then upon inserting the device driver kernel module, the kernel calls the init function.

Example:-

hello_init is the user-defined name for the init function.

```
int hello_init(void)
        {
                printk("hello kernel space!!!\n"); //we can print message like this
                return 0; // we give return 0
        }
```

Exit Functions:-

We use an exit function that executes last when the Linux device driver is unloaded from the kernel.

Example:-

Here's an example of the syntax for the exit function.

```
 void __exit hello_world_exit(void)
        {
                printk("bye kernel space!!!\n");
        //we can print a message like this it helps to Acknowledge  whether exit function executes or not.


         }
```

Kernel Macros:-

The kernel macros have different types but in this basics program it have mandatory macros are used,
module_init(type user defined init function name here);
module_exit(type user defined exit function name here);

Example:-

module_init(hello_init);
-In this macros hello_init is init function name it may be different for you user define init function name.
module_exit(hello_world_exit);
-In this macros hello_world_exit is exit function name it may be different for you user defined exit function name.
MODULE_LICENSE("GPL");
-They are different licenses are used we choose GPL
GPL-General Purpose OS.

This kernel are not mandatory for device drive program but it is useful to mention some details about derives
MODULE_AUTHOR(name of who create drivers);
MODULE_DESCRIPTION(mention what purpose we created that device driver);

Examples:-

MODULE_AUTHOR("sarbudeen kather");
MODULE_DESCRIPTION("Our first kernel module");

After completing the program.To save filename.c extensions.

Step 4:-

Then create Makefile.Makefile sets, a set of rules to determine which parts of a program need to be recompiled, and issues commands to recompile them. Makefile is a way of automating software building procedures and other complex tasks with dependencies. Makefile contains: dependency rules, macros and suffix(or implicit) rules.

Example program file named in Makefile for your reference.

Step 5:-

How to run the device drive program (kernel module)?

1) After creating Makefile go to the terminal and give this command make.
   give enter key it will so some output like below.

Example :-

```
make -C /lib/modules/6.5.0-15-generic/build
M=/home/sarbudeen/embeddedlinux/Embedded_practices/Vector/Hello modules
make[1]: Entering directory '/usr/src/linux-headers-6.5.0-15-generic'
warning: the compiler differs from the one used to build the kernel
  The kernel was built by: x86_64-linux-gnu-gcc-12 (Ubuntu 12.3.0-1ubuntu1~22.04) 12.3.0
  You are using:           gcc-12 (Ubuntu 12.3.0-1ubuntu1~22.04) 12.3.0
  CC [M]  /home/sarbudeen/embeddedlinux/Embedded_practices/Vector/Hello/Hello_world.o
  MODPOST
/home/sarbudeen/embeddedlinux/Embedded_practices/Vector/Hello/Module.symvers
  CC [M]
/home/sarbudeen/embeddedlinux/Embedded_practices/Vector/Hello/Hello_world.mod.o
  LD [M]  /home/sarbudeen/embeddedlinux/Embedded_practices/Vector/Hello/Hello_world.ko
  BTF [M] /home/sarbudeen/embeddedlinux/Embedded_practices/Vector/Hello/Hello_world.ko
Skipping BTF generation for
/home/sarbudeen/embeddedlinux/Embedded_practices/Vector/Hello/Hello_world.ko due to
unavailability of vmlinux
make[1]: Leaving directory '/usr/src/linux-headers-6.5.0-15-generic'
```

And then give ls command it will show some extension files in present working directory to confirmed the make command works correctly or not

Step 6:-

There are two types of command used to insert and remove compiled file in kernel space

- ❖ insmod
- ❖ rmmod

insmod:-

Now enter the compiled program in kernel space with the help of this command
sudo insmod filename.ko

Or

give sudo su command and enter key
Then give insmod filename.ko

Example :-

sudo insmod Hello_world.ko

rmmod:-

After running the program to remove from kernel space use this command
sudo rmmod filename.ko

Or

give sudo su command and enter key
Then give rmmod filename.ko

Example :-

sudo rmmod Hello_world.ko

This is simple basic linux kernel device driver program to execute in linux operating system

Thank You