Linux Device Drivers

Have a look

- Before getting into **Linux Device Drivers,** please take a look at
  - LK_Bird's Eye View session
    S9 Device Driver,Model,Frameworks

Linux Device Drivers

- Linux device drivers classes

  - Character
    Deal with the devices that allow data to be read and written character-by-character, like (input, sound, graphics, serial,...).

  - Block
    Deal with the devices that allow data to be read or written only in multiples of block units, like (Hard disk, CDROM,...).

  - Network
    Communication between the kernel and a network device driver is completely different from that used with char and block drivers.

    Instead of read and write, the kernel calls functions related to packet transmission and sockets.

HDD

Linux Device Drivers

- Elements of Device Model

  - **device**: Each HW should contain "device" structure
    that represent it to the system as a new device.0
    it appears under ***/sys/devices/***.

  - **device_driver**
    Each HW or more than one HW should be handled with a **"devie_driver"**.

  - **bus_type**
    Each HW attached to certain bus "bus_type" (USB, I2c,...).
    it appears under ***/sys/bus/***.

  - **class**
    Each HW belongs to a **"class"**, this class has
    a group of attributes to be implemented like ("leds" class supports the blinking,
    flashing, and brightness control features of physical LED's, "disk" class
    supports block sizes and I/O, removable).

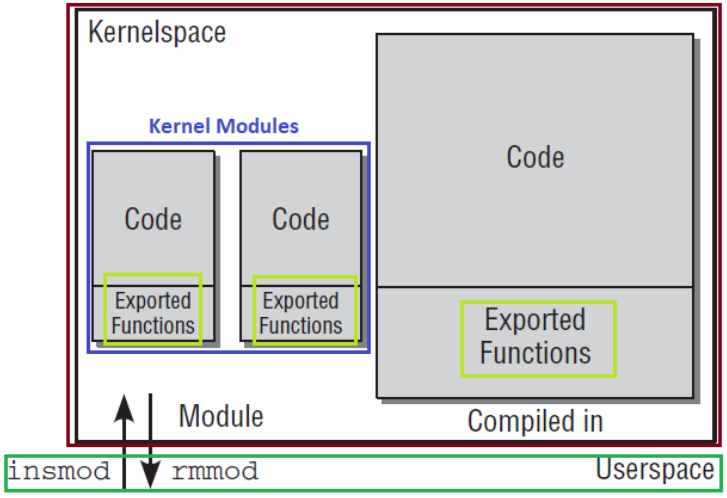
    it appears under ***/sys/class/***.

Kernel Modules

Kernel Modules

- Each piece of code that can be added dynamically to the kernel at run-time is called a ***module*** as each module is made up of object code ***.ko***
- The module code exports functions that can be used by other kernel modules (and also by code permanently compiled into the kernel).
- Add/Remove Modules
  - ***$ file kernelnewbies.ko***

    The kernel module file is **relocatable** which has no function references to absolute addresses but point only to **relative** addresses within the code and can therefore be loaded by the kernel at any offsets in memory.




```
karimeshapa@karimeshapa-Inspiron-5537:~/ldd/kernelnewbies$ file kernelnewbies.ko
kernelnewbies.ko: ELF 64-bit LSB relocatable, x86-64, version 1 (SYSV), BuildID[sha1]=23c9
karimeshapa@karimeshapa-Inspiron-5537:~/ldd/kernelnewbies$
karimeshapa@karimeshapa-Inspiron-5537:~/ldd/kernelnewbies$
karimeshapa@karimeshapa-Inspiron-5537:~/ldd/kernelnewbies$
```

Kernel Modules

- Add/Remove Modules, *Cont'd*

  - ***isnmod*** : uses ***finit_module()*** syscall to copy the module code into kernel memory followed by relocation and the resolution of as yet undefined references in the module.

  - ***rmmod*** : uses ***delete_module()*** syscall to remove a module from the kernel.

  - Kernel Symbol Table
    $ nm kernelnewbies.ko
    a list of all external functions in a module.
    U/u : unresolved reference.
    T/t : symbol is located in the text segment.
    B/b : BSS segment.
    In order to resolve the references,kernel holds symbol table which contains the addresses of global kernel items-functions and variables-that are needed to implement modularized drivers.

    When a module is loaded, any symbol exported by the module becomes part of the kernel symbol table.
    $ sudo cat /proc/kallsyms | grep cdev_add

/kernel/module/main.c

```
SYSCALL_DEFINE2(delete_module, const char __user *, name_user,
        unsigned int, flags)
{
    ...
}

SYSCALL_DEFINE3(finit_module, int, fd, const char __user *, uargs,
int, flags)
{
    ...
}
```

```
karimeshapa@karimeshapa-Inspiron-5537:~/ldd/kernelnewbies$ nm kernelnewbies.ko
                 U alloc_chrdev_region
                 U cdev_add
                 U cdev_del
                 U cdev_init
                 U __class_create
                 U class_destroy
0000000000000000 T cleanup_module
                 U device_create
                 U device_destroy
0000000000000090 b dev_nr
0000000000000019 t driver_open_fifo
0000000000000070 t driver_read_fifo
0000000000000000 t driver_release_fifo
0000000000000000 t driver_write_fifo
0000000000000032 t driver_write_fifo.cold
                 U __fentry__
0000000000000000 d fops
0000000000000000 T init_module
0000000000000000 t kernelnewbies_exit
0000000000000000 t kernelnewbies_init
```

```
karimeshapa@karimeshapa-Inspiron-5537:~/ldd/kernelnewbies$ sudo cat /proc/kallsyms | grep cdev_add
[sudo] password for karimeshapa:
ffffffffabf06dc0 T cdev_add
```

Kernel Modules

- Add/Remove Modules, *Cont'd*

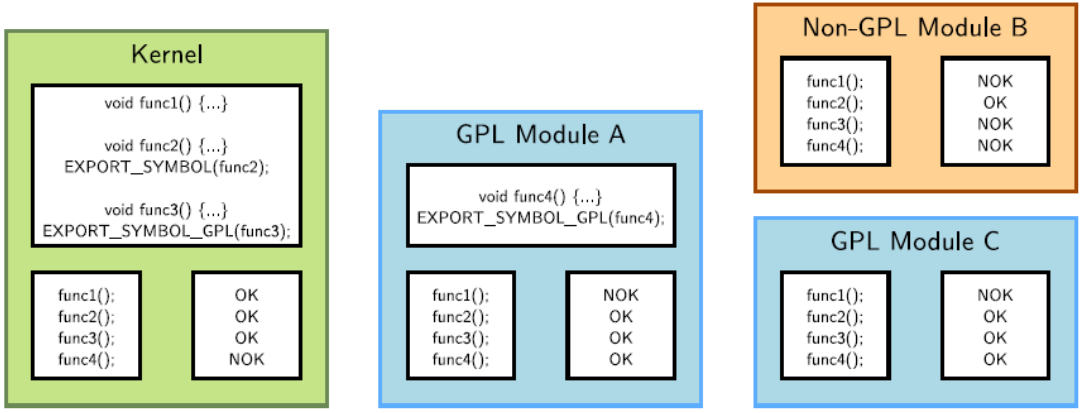
  - Kernel Symbol Table, *Cont'd*
    module that needs to export symbols for other modules

    EXPORT_SYMBOL(name) : exports a function or variable to all modules.
    EXPORT_SYMBOL_GPL(name) : which exports a function or variable only to GPL.

  - Initialization and Shutdown
    module_init(kernelnewbies_init);
    module_exit(kernelnewbies_exit);
    __init, __initdata, __exit, __exitdata
    Markers for functions (__init and __exit) and data (__initdata and __exitdata) that are only used at module

    initialization or cleanup time. Items marked for initialization may be discarded once initialization completes; the exit items may be discarded if module unloading has not been configured into the kernel.

    These markers work by causing the relevant objects to be placed in a special ELF section in the executable file.



/include/linux/init.h

```
#define __init      __section(".init.text") __cold  __latent_entropy __noinitretpoline __nocfi
#define __initdata  __section(".init.data")
#define __initconst __section(".init.rodata")
#define __exitdata  __section(".exit.data")
#define __exit_call __used __section(".exitcall.exit")
```

Kernel Modules

- Module Dependencies
  - ***modprobe***, like ***insmod,*** but it looks at the dependencies of the module to load them first.
  - The order in which modules are added into the kernel is important if there are inter-dependencies between the modules.
  - Module dependencies listed at, ***/lib/modules/version/modules.dep.*** and all symbols exported by the kernel are written into ***/boot/System.map***

```
karimeshapa@karimeshapa-Inspiron-5537:~/ldd/kernelnewbies$ cat /lib/modules/5.8.0-43-generic/modules.dep | grep cramfs
kernel/fs/cramfs/cramfs.ko: kernel/drivers/mtd/mtd.ko
karimeshapa@karimeshapa-Inspiron-5537:~/ldd/kernelnewbies$
karimeshapa@karimeshapa-Inspiron-5537:~/ldd/kernelnewbies$
```

```
karimeshapa@karimeshapa-Inspiron-5537:~/ldd/kernelnewbies$ sudo cat /boot/System.map-5.8.0-43-generic | less
0000000000000000 D __per_cpu_start
0000000000000000 D fixed_percpu_data
00000000000001d9 A kexec_control_code_size
0000000000001000 D cpu_debug_store
0000000000002000 D irq_stack_backing_store
0000000000006000 D cpu_tss_rw
000000000000b000 D gdt_page
000000000000c000 d exception_stacks
0000000000010000 d entry_stack_storage
0000000000011000 D espfix_waddr
0000000000011008 D espfix_stack
0000000000011010 D cpu_llc_id
0000000000011020 d mce_banks_array
0000000000011220 D mce_num_banks
0000000000011240 D cpu_llc_shared_map
```

Kernel Modules

- Module Info
  - ***lsmod***
    Displays the list of loaded modules.
    memory size, instances
  - ***/proc/modules***
    name, memory size, instances,
    dependencies, state, kernel memory offset
  - ***Modinfo***
    Get information about a module
    parameters, license, description and dependencies.
- Module Parameters
  - read/write module parameters
    *module_param(fsize, int, **S_IWUSR|S_IRUGO**);*

```
root@karimeshapa-Inspiron-5537:/home/karimeshapa/ldd/kernelnewbies# lsmod | grep kernelnewbies
kernelnewbies          16384  0
root@karimeshapa-Inspiron-5537:/home/karimeshapa/ldd/kernelnewbies#
```

```
root@karimeshapa-Inspiron-5537:/home/karimeshapa/ldd/kernelnewbies# cat /proc/modules | grep kernelnewbies
kernelnewbies 16384 0 - Live 0xffffffffc1055000 (OE)
root@karimeshapa-Inspiron-5537:/home/karimeshapa/ldd/kernelnewbies#
```

```
root@karimeshapa-Inspiron-5537:/home/karimeshapa/ldd/kernelnewbies# modinfo kernelnewbies.ko
filename:       /home/karimeshapa/ldd/kernelnewbies/kernelnewbies.ko
license:        GPL
description:    kernelnewbies ldd
author:         Linux Community
srcversion:     12E008B8262395689A48866
depends:
retpoline:      Y
name:           kernelnewbies
vermagic:       5.8.0-43-generic SMP mod_unload
parm:           fsize:int
root@karimeshapa-Inspiron-5537:/home/karimeshapa/ldd/kernelnewbies#
```

```
root@karimeshapa-Inspiron-5537:/home/karimeshapa/ldd/kernelnewbies# cat /sys/module/kernelnewbies/parameters/fsize
16
root@karimeshapa-Inspiron-5537:/home/karimeshapa/ldd/kernelnewbies#
```

Kernel Modules

- Doing It in User Space

  - Advantages
    1) The full C library can be linked in.

    2) If a user-space driver hangs, you can simply kill it.

  - Drawbacks
    1) Interrupts are not available in user space.

    2) Direct access to memory is possible only by
    mmapping /dev/mem, and only a privileged user can do that.

    3) Response time is slower, because a context switch is required
    to transfer information or actions between the client and the hardware.

    4) The most important devices can’t be handled in user space, including,
    but not limited to, network interfaces and block devices.