### Kernel coding style

scripts/cleanfile - size the tabs and verify the line size

Or use

indent tool

2)

Every function/variable that is not exported should be declared static.

3)

No spaces should be added around (inside) parenthesized expressions

4)

Using typedefs is forbidden.

5)

Always use multi-line comment style instead of single line.

6)

Capitalize macros, functional macros can be in lowercase.

7)

A comment should not replace code that is not illegible.

8)

Dynamic initializers are all macros, they are capitalized.

Eg: INIT\_LIST\_HEAD(),

DECLARE\_TASKLET()

9)

Data structures that represent framework devices are always allocated dynamically, each having its own allocation and deallocation API.

Kernel implements OOP through devices and classes.

Kernel subsystems are abstracted by means of classes. The struct kobject is the centrepiece of this implementation.

Every device that falls into a given subsystem has a pointer to an operations(ops) structure, which exposes the operations that can be performed on this device.

### Device Driver Basics

A device driver exposes the functionality of the hardware to the user programs.

Kernel space - set/range of addresses where kernel is hosted and where it runs. A memory range, protected by access flags. In this mode, CPU can access the whole memory.

User space - set/range of addresses used by applications. It can’t mess with memory or other resources of other programs. Use system calls to get kernel services - read, write, open, close, mmap etc.

**Modules**

CONFIG\_MODULES=y

EXPORT\_SYMBOL - macro using which symbols in a module can be exported.

depmod - a tool that is run during the kernel build process to generate module dependency files, by reading each module in */lib/modules/<kernel\_release>/* to determine what symbols it should export and what symbols it needs. Result is written to the *modules.dep* file. Binary version - *modules.dep.bin*.

##### Module loading

insmod -

modprobe -

/etc/modules-load.d/<filename>.conf

-specify the modules to be loaded at boot time. One module per line. As many .conf files can be created.

Device driver will have the product and vendor IDs. *depmod* processes module files in order to extract and gather that information and generates *modules.alias* file in */lib/modules/<kernel\_version>/*, which maps devices to drivers.

A user-space hot-plug agent (or device manager), udev or mdev, will register with the kernel in order to get notified when a new device appears.

The notification is done by the kernel, sending the device's description (pid, vid, class, device class, device subclass, interface, and all other information that may identify a device) to the hot-plug daemon, which in turn calls *modprobe* with this information. *modprobe* then parses the *modules.alias* file in order to match the driver associated with the device. Before loading the module,*modprobe* will look for its dependencies in *module.dep*. If it finds any, the dependencies will be loaded prior to the associated module loading; otherwise, the module is loaded directly.

##### Module Unloading

CONFIG\_MODULE\_UNLOAD=y

rmmod-

To forcefully unload a module that is unsafe to unload, option MODULE\_FORCE\_UNLOAD=y must be set.

Then while kernel is running, you can do a

$ rmmod -f <module>

$ modprobe -r <module>

-it automatically unloads unused dependencies

$ lsmod

- lists all the modules that are loaded.

## Device Trees