Loadable Kernel Module

Kathan V. Adhyaru (121022)

**What is LKM?**

To add a new code to a Linux kernel, it is necessary to add some source files to kernel source tree and recompile the kernel. But you can also add code to the Linux kernel while it is running. A chunk of code added in such way is called a loadable kernel module

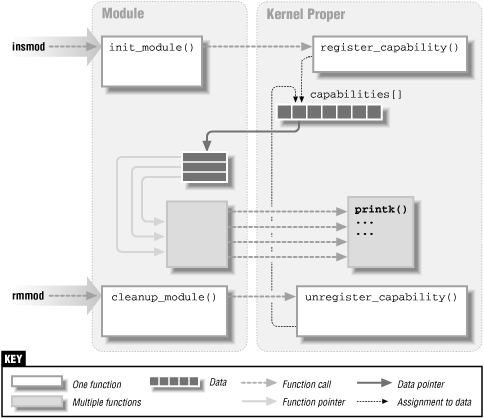
Types of modules:

(i)device drivers, (ii)file system drivers, (iii)system calls

Build working base kernel, that include anything that is necessary to get the system up, everything else can be built as modules .Modules are stored in the file system as ELF object files.The kernel keeps track of the use of modules, so that no modules is unloaded while another module or kernel is using it (/proc/modules).

**Linking and Unlinking the module**

* **Programs for linking and unlinking**

1.insmod

-Reads from the name of the module to be linked. Locates the file containing the module's object code

-Computes the size of the memory area needed to store the module code, its name, and the module object

-Invokes the create\_module( ) system call & Invokes the query\_module( ) system call. Using the kernel symbol table, the module symbol tables, and the address returned by

the create\_module( ) system call, relocates the object code included in the module's file.

-Allocates a memory area in the User Mode address space and loads with a copy of the module object. Invokes the init\_module( ) system call, passing to it the address of the User Mode memory area .Releases the User Mode memory area and terminates

2.ismod

-reads /proc/modules

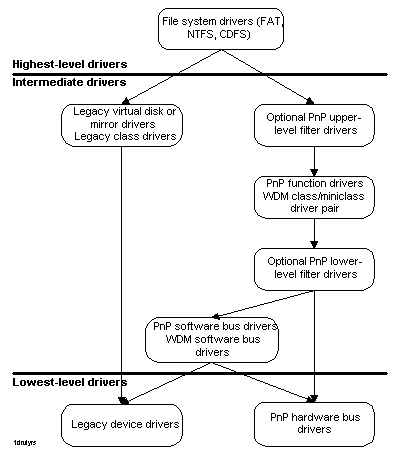
3.rmmod

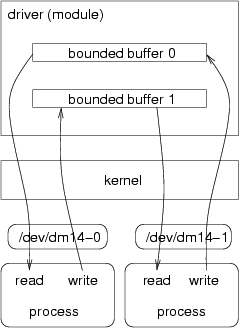
-From reads the name of the module to be unlinked.

-Invokes the query\_module( ) & Invokes the delete\_module( ) system call, with the QM\_REFS subcommand several times, to retrieve dependency information on the linked modules.

3.modprobe

-takes care of possible complications due to module dependencies, uses depmod program and /etc/modules.conf file

**Device Drivers**

-There are two major ways for a kernel module to talk to processes:

* To use the proc file system (**/proc** directory)
  + Through device files (**/dev** directory)

-Device driver sits between some hardware and the kernel I/O subsystem. Its purpose is to give the kernel a consistent interface to the type of hardware it "drives".

**Advantages**

-There is no necessity to rebuild the kernel, when a new kernel option is added

-Modules help find system problems (if system problem caused a module just don't load it)

-Modules save memory

-Modules are much faster to maintain and debug

-Modules once loaded are inasmuch fast as kernel

**Limitations with Current Models**

-Generic driver model (WDM) is too complex

# Focuses on very advanced drivers which punishes simple ones

-Many drivers must be written in kernel mode

* + Even though much functionality could be user mode

-Developers spend too much time driving our software

* + Cannot concentrate on driving their hardware
  + Driver quality suffers as a result

-Do not allow extension and future growth

**References**

* [**http://unix.stackexchange.com/questions/1605/benefit-of-kernel-module-compiled-inside-kernel**](http://unix.stackexchange.com/questions/1605/benefit-of-kernel-module-compiled-inside-kernel)
* [**http://en.wikipedia.org/wiki/Loadable\_kernel\_module**](http://en.wikipedia.org/wiki/Loadable_kernel_module)
* [**http://www.tldp.org/LDP/lkmpg/2.6/lkmpg.pdf**](http://www.tldp.org/LDP/lkmpg/2.6/lkmpg.pdf)
* [**http://h21007.www2.hp.com/portal/download/files/unprot/drivers/docs/refs/ddguide/Chap10.pdf**](http://h21007.www2.hp.com/portal/download/files/unprot/drivers/docs/refs/ddguide/Chap10.pdf)
* [**http://www.webalice.it/roberto.farina/teaching/embsys/Modules.pdf**](http://www.webalice.it/roberto.farina/teaching/embsys/Modules.pdf)