Linux Device Driver

(Kmod & Advanced Modularization)

Amir Hossein Payberah

payberah@yahoo.com

Contents

- Loading Module on Demand
- Intermodule Communication

Loading Module on Demand

- To make it easier for users to load and unload modules, Linux offers support for automatic loading and unloading of modules.
  - To avoid wasting kernel memory.
  - To allow the creation of ‘‘generic’’ kernels that can support a wide variety of hardware.

kmod

- Whenever the kernel tries to access certain types of resources and finds them unavailable, it makes a special kernel call to the kmod subsystem instead of simply returning an error.
- kmod was initially implemented as a separate, standalone kernel process that handled module loading requests.

Request modules in kernel

- Any kernel-space code can request the loading of a module when needed.
  - By invoking a facility known as kmod.
- int request_module(const char *module_name);
- It is defined in <linux/kmod.h>.

Request_module

- request_module is synchronous.
- The return value indicates that request_module was successful in running modprobe, but does not reflect the success status of modprobe itself.

The user space side

- The actual task of loading a module requires help from user space.
- When the kernel code calls request_module, a new ‘‘kernel thread’’ process is created, which runs a helper program in the user context.
  - □ This program is called modprobe.

modprobe

- It just calls insmod with the name of a module as passed to request_module.
- It can also handle module dependencies.
  - If a requested module requires yet another module to function, modprobe will load both.
  - Assuming that depmod -a was run after the modules have been installed.
- The modprobe utility is configured by the file /etc/modules.conf.

/etc/modules.conf

- path[misc]=directory
  - Miscellaneous modules can be found in the misc subdirectory under the given directory.
- Keep
  - By placing a keep before any path directives, you can cause to add new paths to the list instead of replacing it.
- alias alias_name real_name

/etc/modules.conf

- options [-k] *module opts*
  - Provides a set of options *(opts)* for the given *module* when it is loaded.
- pre-install *module command*
- post-install *module command*
  - Specify a *command* to be run either before or after the given *module* is installed.
- pre-remove *module command*
- post-remove *module command*
  - The command before or after module removal.

Sample

```
alias scsi_hostadapter aic7xxx
alias eth0 eepro100
pre-install pcmcia_core /etc/rc.d/init.d/pcmcia start
options short irq=1
alias sound es1370
```

Contents

- Loading Module on Demand
- Intermodule Communication

Intermodule communication

- The intermodule scheme allows modules to register strings pointing to data of interest, which can be retrieved by other modules.

Intermodule communication

- Sender side functions:
- void inter_module_register(const char *string, struct module *module, const void *data);
- void inter_module_unregister(const char *string);

Intermodule communication

- Receiver side functions:
- const void *inter_module_get(const char *string);
- const void inter_module_get_request(const char *string, const char *module);
- void inter_module_put(const char *string);

Sender sample

```
static char *string = "inter says ’Hello World’";
void ime_function(const char *who)
{
	printk(KERN_INFO "inter: ime_function called by %s\n", who);
}

int ime_init(void)
{
	inter_module_register("ime_string", THIS_MODULE, string);
	inter_module_register("ime_function", THIS_MODULE, ime_function);
	return 0;
}
void ime_cleanup(void)
{
	inter_module_unregister("ime_string");
	inter_module_unregister("ime_function");
}
```

Receiver sample

```
static const char *ime_string = NULL;
static void master_test_inter();
void master_test_inter()
{
     void (*ime_func)();
     ime_string = inter_module_get_request("ime_string", "inter");
     printk(KERN_INFO "master: got ime_string ’%s’\n", ime_string);
     ime_func = inter_module_get("ime_function");
      (*ime_func)("master");
     inter_module_put("ime_function");
}

void master_cleanup_module(void)
{
     inter_module_put("ime_string");
}
```

Question?