Linux Device Driver
Kernel Module Programing

https://community.ruggedboard.com

Kernal Module

What is a Kernel Module?

Kernel Modules are piece of code that can be loaded/inserted
and unloaded/removed from the kernel as per the demand/need.

Other Names

1. Loadable Kernel Modules (LKM)
2. Modules

Extension: .ko (Kernel Object)

Standard Location for Kernel Modules

Modules are installed in the /lib/modules/<kernel version> directory of the rootfs by default.

Configuration

In order to support modules, the kernel must have been built with the following option enabled:

CONFIG_MODULES=y

Kernal Module

Basic Commands

1. List Modules: (lsmod) lsmod gets its information by reading the file /sys/modules.
2. Module Information: (modinfo) : prints the information of the module.

Kernel modules must have at least two functions:

a "start" (initialization) function : which is called when the module is loaded into the kernel
an "end" (cleanup) function called : which is called just before it is removed
This is done with the module_init() and module_exit() macros

Header Files

Every kernel module needs to include linux/module.h.
for macro expansion of module_init and module_exit
linux/kernel.h only for the macro expansion for the printk() log level

Kernal Module

Module should specify which license you are using MODULE_LICENSE() macro

"GPL"	[GNU Public License v2 or later]
"GPL v2"	[GNU Public License v2]
"GPL and additional rights“	[GNU Public License v2 rights and more]
“Dual BSD/GPL"	[GNU Public License v2 or BSD license choice]
"Dual MIT/GPL"	[GNU Public License v2 or MIT license choice]
"Dual MPL/GPL"	[GNU Public License v2 or Mozilla license choice]
"Proprietary"	[Non free products]

Kernal Module

```
#include <linux/kernel.h>
#include <linux/module.h>

MODULE_LICENSE(“GPL");
static int test_hello_init(void)
{
   printk(KERN_INFO"%s: In init\n", __func__);
   return 0;
}

static void test_hello_exit(void)
{
   printk(KERN_INFO"%s: In exit\n", __func__);
}

module_init(test_hello_init);
module_exit(test_hello_exit);
```

To Build Modules:

make -C /lib/modules/`uname -r`/build M=${PWD} modules

To clean:

make -C /lib/modules/`uname -r`/build M=${PWD} clean

The above commands starts by changing its directory to the one provided with the -C option (that is your kernel source directory)There it finds the kernel's top level makefile.

The M option causes the Makefile to move back into your module source directory before trying to build the modules.

Note:

M is not make option but argument passed to it

obj-m refers to the list of modules

The kernel Makefile will read our Makefile to find out what to build, we specify that by writing **obj-m += hello.o**

Kernal Module

printk() is a kernel level function

The printk() is called with one more argument than printf(), like this:

printk(KERN_log_priority "hello world\n");

Here, log_priority is one of the eight values

*(predefined in **linux/kernel.h**, similar to **/usr/include/sys/syslog.h**)*

EMERG,
ALERT,
CRIT,
ERR,
WARNING,
NOTICE,
INFO,
DEBUG (in order of decreasing priority).

printk() writes to the kernel buffer

Connect-us

Developer Wiki

Open Discussions