Linux Kernel Programming

Kernel Modules and Memory Allocation

Pierre Olivier

Systems Software Research Group @ Virginia Tech

February 1, 2017

Virginia Tech

Outline

Virginia Tech

Outline

Virginia Tech

Kernel modules: presentation

General information

- **Modules** are pieces of kernel code that can be **dynamically loaded and unloaded at runtime**
  - No need to reboot
- Appeared in Linux 1.2 (1995)
- Numerous Linux features can be compiled as modules
  - Selection in the configuration `.config` file
    - Ex: device/filesystem drivers
    - Generated through the `menuconfig` make target
  - Opposed to **built-in** in the kernel binary executable `vmlinux`

Virginia Tech

Kernel modules: presentation

Benefits of kernel modules

- Modules benefits:
  - **No reboot**
    - Saves a lot of time when developing/debugging
  - **No need to compile the entire kernel**
  - **Saves memory and CPU time** by running on-demand
  - No performance difference between module and built-in kernel code
  - Help **identifying buggy code**
    - Ex: identifying a buggy driver compiled as a module by selectively running them

Virginia Tech

Outline

Virginia Tech

Writing a kernel module

Basic C file for a module

```
1  #include <linux/module.h> /* Needed by all modules */
2  #include <linux/kernel.h> /* KERN_INFO */
3  #include <linux/init.h>   /* Init and exit macros */
4
5  static int answer __initdata = 42;
6
7  static int __init lkp_init(void)
8  {
9    printk(KERN_INFO "Module loaded ...\n");
10     printk(KERN_INFO "The answer is %d ...\n", answer);
11
12     /* Return 0 on success, something else on error */
13     return 0;
14 }
15
16 static void __exit lkp_exit(void)
17 {
18   printk(KERN_INFO "Module exiting ...\n");
19 }
20
21 module_init(lkp_init);
22 module_exit(lkp_exit);
23
24 MODULE_LICENSE("GPL");
25 MODULE_AUTHOR("Pierre Olivier <polivier@vt.edu");
26 MODULE_DESCRIPTION("Sample kernel module");
```

- Create a C file anywhere on the filesystem
  - No need to be inside the kernel sources
- Init. & exit functions
  - Launched at load/unload time
- `MODULE_*` macros
  - General info about the module

Virginia Tech

Writing a kernel module

Kernel namespace

- Module is linked against the entire kernel:
  - Module has visibility on all of the kernel global variables
  - To avoid namespace pollution and involuntary reuse of variables names:
    - Use a well defined naming convention. Ex:
      `my_module_function_a()`
      `my_module_function_b()`
      `my_module_global_variable`
    - Use `static` as much as possible
- Kernel symbols list is generally present in:
  `/proc/kallsyms`

Virginia Tech

Outline

Virginia Tech

Compiling a kernel module

Kernel sources & module Makefile

- Need to have the kernel sources somewhere on the filesystem
- Create a `Makefile` in the same directory as the module source file

```
1  # let's assume the module C file is named lkp.c
2  obj-m := lkp.o
3  KDIR := /path/to/kernel/sources/root/directory
4  # Alternative: Debian/Ubuntu with kernel-headers package
       :
5  # KDIR := /lib/modules/$(shell uname -r)/build
6  PWD := $(shell pwd)
7
8  all: lkp.c
9    make -C $(KDIR) SUBDIRS=$(PWD) modules
10
11 clean:
12   make -C $(KDIR) SUBDIRS=$(PWD) clean
```

- Multiple source files?

```
1  obj-m += file1.c
2  obj-m += file2.c
3  # etc.
```

- After compilation, the compiled module is the file with `.ko` extension

Outline

Virginia Tech

Launching a kernel module

insmod/rmmod

- Needs administrator privileges (root)
  - You are executing kernel code!
- Using `insmod`:

```
1 sudo insmod file.ko
```

  - Module is loaded and init function is executed
- Note that **a module is compiled against a specific kernel version and will not load on another kernel**
  - This check can be bypassed through a mechanism called `modversions` but it can be dangerous
- Remove the module with `rmmod`:

```
1 sudo rmmod file
2 # or:
3 sudo rmmod file.ko
```

  - Module exit function is called

Virginia Tech

Launching a kernel module

modprobe

- `make modules_install` from the kernel sources installs the modules in a standard location on the filesystem
  - Generally `/lib/modules/<kernel version>/`
- These modules can be inserted through `modprobe`:

```
1 sudo modprobe <module name>
```

  - No need to point to a file, just give the module name
- Contrary to `insmod`, `modprobe` handles modules dependencies
  - Dependency list generated in `/lib/modules/<kernel version/modules.dep`
- Remove using `modprobe -r <module name>`
- Such installed modules can be loaded automatically at boot time by editing `/etc/modules` or the files in `/etc/modprobe.d`

Virginia Tech

Outline

Virginia Tech

Modules: miscellaneous information

Modules parameters

- **Parameters** can be entered from the command line at launch time

```c
#include <linux/module.h>
/* ... */

static int int_param = 42;
static char *string_param = "default value";

module_param(int_param, int, 0);
MODULE_PARM_DESC(int_param, "A sample integer kernel module parameter");
module_param(string_param, charp, S_IRUSR | S_IWUSR | S_IRGRP | S_IROTH);
MODULE_PARM_DESC(string_param, "Another parameter, a string");

static int __init lkp_init(void)
{
    printk(KERN_INFO "Int param: %d\n", int_param);
    printk(KERN_INFO "String param: %s \n", string_param);

    /* ... */
}

/* ... */
```

```
sudo insmod lkp.ko int_param=12 string_param="hello"
```

Virginia Tech

Modules: miscellaneous information

modinfo, lsmod

- `modinfo`: info about a kernel module
  - Description, kernel version, parameters, author, etc.

```
1  modinfo my_module.ko
2  filename:       /tmp/test/my_module.ko
3  description:    Sample kernel module
4  author:         Pierre Olivier <polivier@vt.edu
5  license:        GPL
6  srcversion:     A5ADE92B1C81DCC4F774A37
7  depends:
8  vermagic:       4.8.0-34-generic SMP mod_unload modversions
9  parm:           int_param:A sample integer kernel module parameter (int)
10 parm:           string_param:Another parameter, a string (charp)
```

- `lsmod`: list currently running modules
  - Can also look in `/proc/modules`

Modules: miscellaneous information

Additional sources of information on kernel modules

- The linux kernel module programming guide:
  - `http://www.tldp.org/LDP/lkmpg/2.6/html/index.html`
- Linux loadable kernel module howto
  - `http://www.tldp.org/HOWTO/Module-HOWTO/index.html`
- Linux sources → `Documentation/kbuild/modules.txt`

Virginia Tech

Outline

Virginia Tech

Memory allocation

kmalloc

- Allocate memory that is virtually and **physically contiguous**
  - For DMA, memory-mapped I/O, and performance (large pages)
- Because of that property, maximum allocated size through one `kmalloc` invocation is limited
  - 4MB on x86 (architecture dependent)

```
1 #include <linux/slab.h>
2 /* ... */
3 char *my_string = (char *)kmalloc(128, GFP_KERNEL);
4 my_struct my_struct_ptr = (my_struct *)kmalloc(sizeof(my_struct), GFP_KERNEL);
5 /* ... */
6 kfree(my_string);
7 kfree(my_struct_ptr);
```

- Returns a pointer to the allocated memory or `NULL` in case of failure
- Mostly used **flags**:
  - `GFP_KERNEL`: ***might sleep***
  - `GFP_ATOMIC`: do not block, but higher chance of failure

Virginia Tech

Memory allocation

vmalloc

- Allocate memory that is **virtually contiguous, but not physically contiguous**
- No size limit other than the amount of free RAM (at least on 64 bit architectures
- ***Might sleep***

```
1 #include <linux/vmalloc.h>
2 /* ... */
3 char *my_string = (char *)vmalloc(128);
4 my_struct my_struct_ptr = (my_struct *)vmalloc(sizeof(my_struct));
5 /* ... */
6 vfree(my_string);
7 vfree(my_struct_ptr);
```

- Returns a pointer to the allocated memory or `NULL` in case of failure

Virginia Tech