Linux Kernel Module Programming

Tushar B. Kute,
http://tusharkute.com

Kernel Modules

- Kernel modules are piece of code, that can be loaded and unloaded from kernel on demand.
- Kernel modules offers an easy way to extend the functionality of the base kernel without having to rebuild or recompile the kernel again. Most of the drivers are implemented as a Linux kernel modules. When those drivers are not needed, we can unload only that specific driver, which will reduce the kernel image size.
- The kernel modules will have a **.ko** extension. On a normal linux system, the kernel modules will reside inside /lib/modules/<kernel_version>/kernel/ directory.

Examples:

- Typical modules:
  - Device drivers
  - File system drivers
  - System calls

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 are much faster to maintain and debug.
- Modules once loaded are in as much fast as kernel.

Utilities to manipulate kernel modules

- `lsmod`
- `insmod`
- `modinfo`
- `rmmod`
- `modprobe`

lsmod

- Lists modules that are loaded already.

```
tushar@tushar-laptop ~ $ lsmod
Module                  Size  Used by
nls_utf8               12493  1
udf                    83847  1
nls_iso8859_1          12617  1
crc_itu_t              12627  1 udf
rfcomm                 53664  8
bnep                   18895  2
binfmt_misc            13140  1
hid_generic            12492  0
dm_multipath           22402  0
scsi_dh                14458  1 dm_multipath
usbhid                 47070  0
hid                    87604  2 hid_generic,usbhid
uvcvideo               71309  0
videobuf2_vmalloc      13048  1 uvcvideo
videobuf2_memops       13170  1 videobuf2_vmalloc
videobuf2_core         39258  1 uvcvideo
```

- Check the command: `cat /proc/modules`

modinfo

- Display module information.

```
tushar@tushar-laptop ~ $ modinfo usb_storage
filename:       /lib/modules/3.13.0-37-generic/kernel/drivers/usb/storage/usb-storag
e.ko
license:        GPL
description:    USB Mass Storage driver for Linux
author:         Matthew Dharm <mdharm-usb@one-eyed-alien.net>
srcversion:     13955DAA5B7302244B5FD1E
alias:          usb:v*p*d*dc*dsc*dp*ic08isc06ip50in*
alias:          usb:v*p*d*dc*dsc*dp*ic08isc05ip50in*
alias:          usb:v*p*d*dc*dsc*dp*ic08isc04ip50in*
alias:          usb:v*p*d*dc*dsc*dp*ic08isc03ip50in*
alias:          usb:v*p*d*dc*dsc*dp*ic08isc02ip50in*
alias:          usb:v*p*d*dc*dsc*dp*ic08isc01ip50in*
alias:          usb:v*p*d*dc*dsc*dp*ic08isc06ip00in*
alias:          usb:v*p*d*dc*dsc*dp*ic08isc05ip00in*
alias:          usb:v*p*d*dc*dsc*dp*ic08isc04ip00in*
alias:          usb:v*p*d*dc*dsc*dp*ic08isc03ip00in*
```

insmod

- Insert module into kernel.
- Syntax:
  - `insmod <module_name>.ko`

rmmod

- Removes module from kernel. You cannot remove a module which is already used by any program.
- Syntax:
  - `rmmod <module_name>.ko`

Kernel Module Implementation

- The kernel considers only modules that have been loaded into RAM by the *insmod* program and for each of them allocates memory area containing:
  - A module object.
  - Null terminated string that represents module's name.
  - The code that implements the functions of the module.

Linux Kernel Module Programming

- Write a `hello_proc.c` program
- Create a Makefile
- The program and Makefile should be kept in a single folder.
- Change directory to this folder and execute following:
  - **`make`**
  - **`insmod hello_proc.ko`**
  - **`dmesg`** (see the kernel buffer contents, reads the kernel log file /var/log/syslog)
  - **`lsmod`**
  - **`cat /proc/hello_proc`**

Files created after building the module

- `hello.o`
  - Module object file before linking.
- `hello.mod.c`
  - Contains module's information.
- `hello.mod.o`
  - After compilation and linking of hello.mod.c.
- `modules.order`
  - The order in which two or three modules get linked.
- `Modules.symvers`
  - Symbol versions if any.
- `hello.ko`
  - A module kernel object file after linking hello.o and hello.mod.o

Example: hello.c

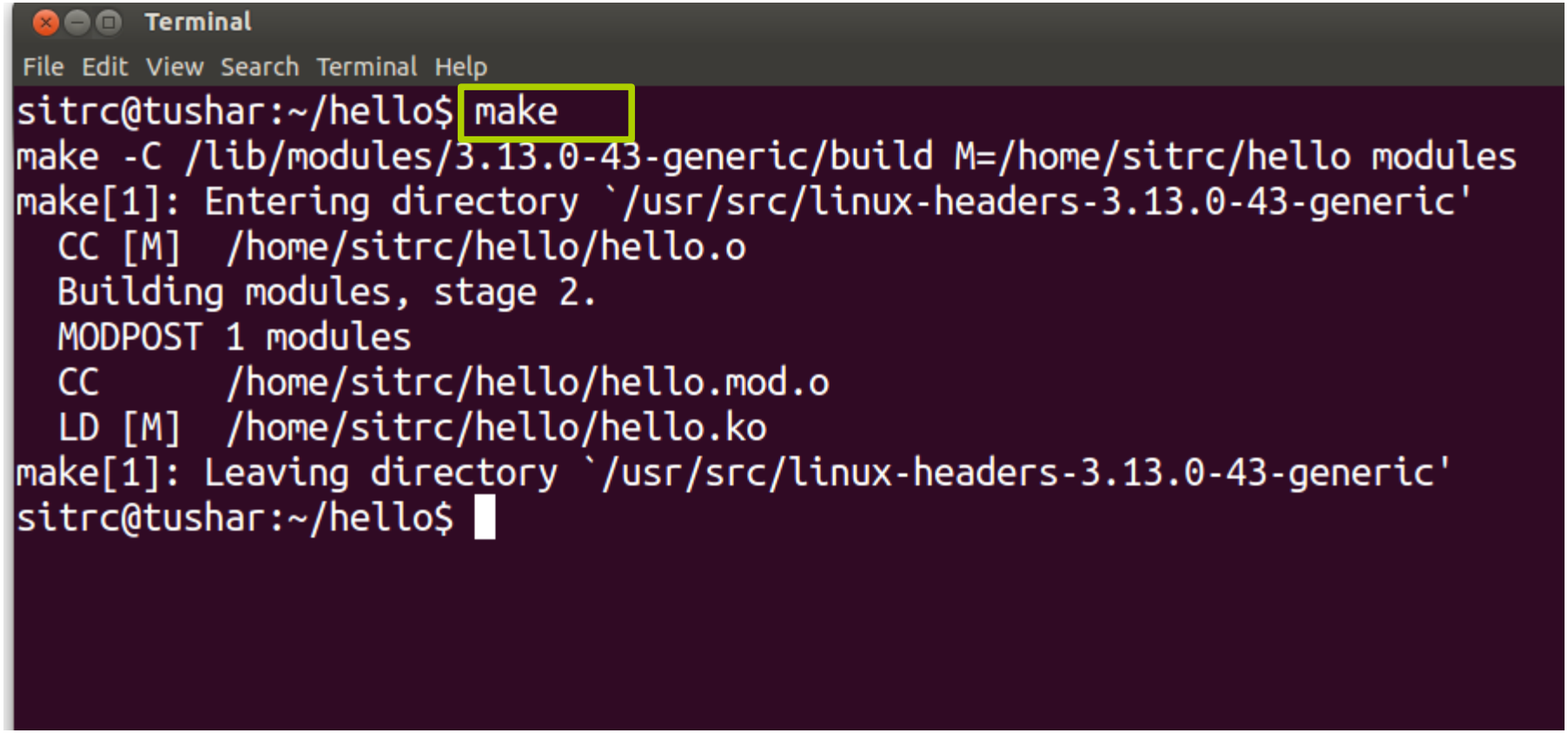
```
#include <linux/module.h>     // included for all kernel modules
#include <linux/kernel.h>     // included for KERN_INFO
#include <linux/init.h>       // included for __init and __exit macros
MODULE_LICENSE("GPL");
MODULE_AUTHOR("Tushar B Kute");
MODULE_DESCRIPTION("A Simple Hello World module");
static int __init hello_init(void)
{
    printk(KERN_INFO "Hello world!\n");
    return 0; // Non-zero return means that the module couldn't be
loaded.
}
static void __exit hello_cleanup(void)
{
    printk(KERN_INFO "Good Bye.\n");
}
module_init(hello_init);
module_exit(hello_cleanup);
```

Makefile

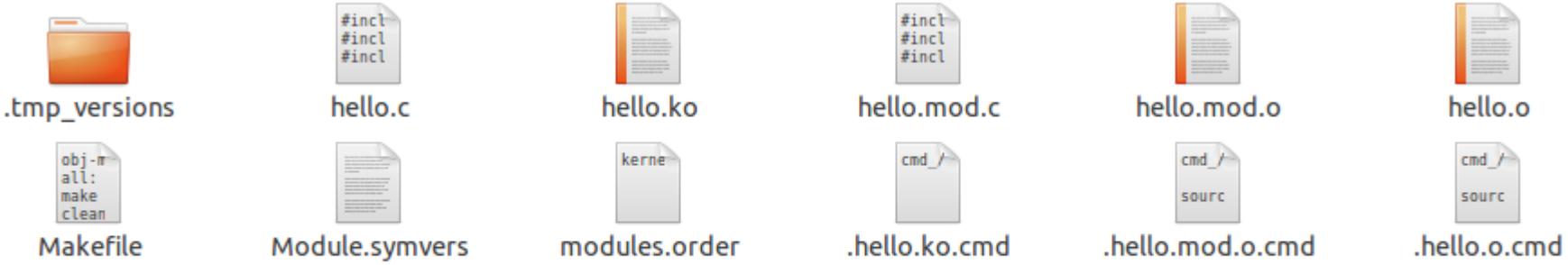
```
obj-m += hello.o

all:
    make -C /lib/modules/$(shell uname -r)/build M=$(PWD)
    modules

clean:
    make -C /lib/modules/$(shell uname -r)/build M=$(PWD)
    clean
```

Compile the program

- **`make`**

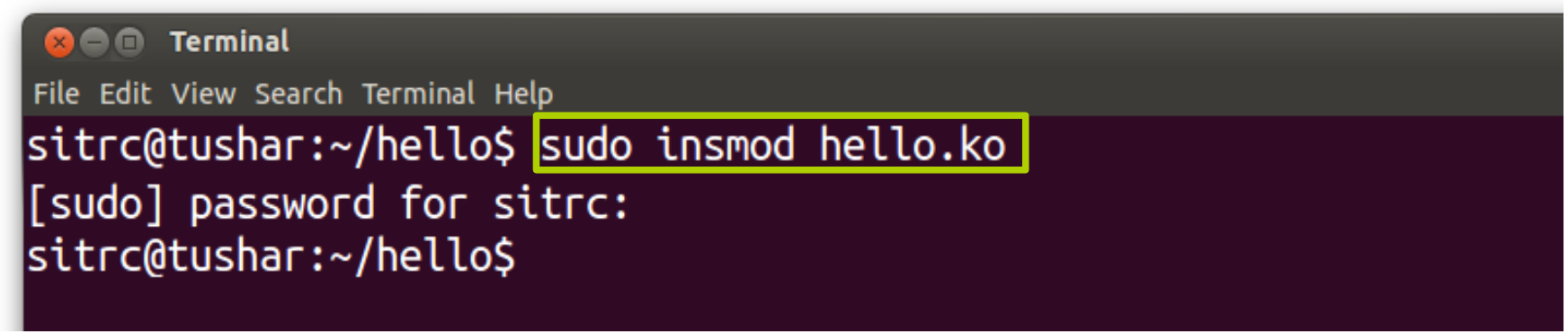


File generated



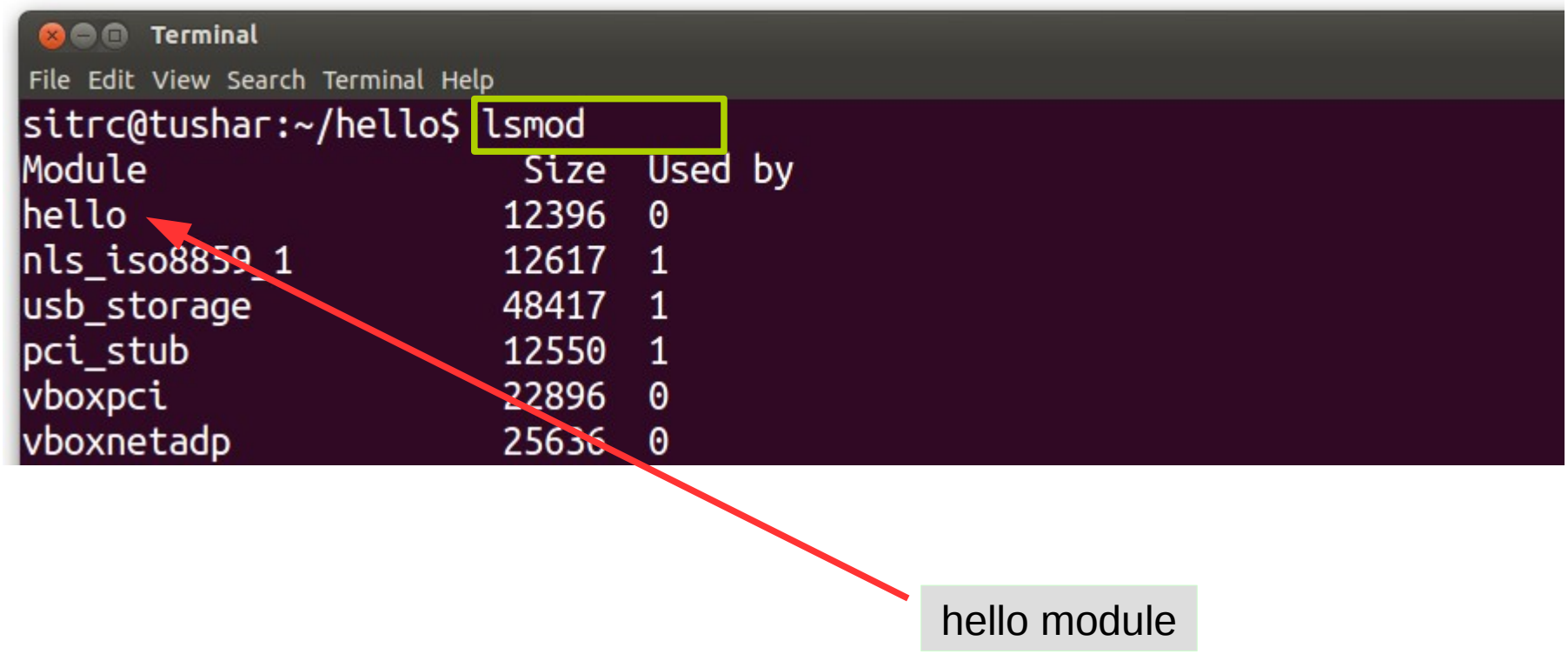
Insert the module

- **`insmod hello.ko`**



Check the module in the list

- **`lsmod`**



Check kernel ring buffer

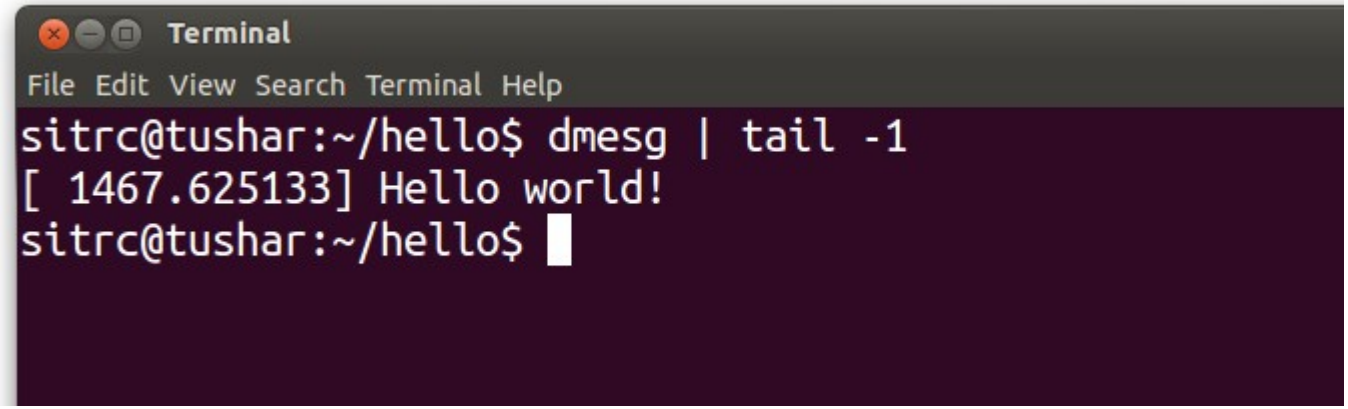
- **`dmesg`**

```
[  340.234241] sd 4:0:0:0: [sdb] Write cache: disabled, read cache: enabled, doe
[  340.239509]  sdb: sdb1
[  340.243618] sd 4:0:0:0: [sdb] Attached SCSI removable disk
[  341.180282] FAT-fs (sdb1): Volume was not properly unmounted. Some data may b
.
[  344.999200] systemd-hostnamed[2932]: Warning: nss-myhostname is not installed
ame might make it unresolveable. Please install nss-myhostname!
[  672.696406] systemd-hostnamed[3458]: Warning: nss-myhostname is not installed
ame might make it unresolveable. Please install nss-myhostname!
[ 1017.496450] perf samples too long (2523 > 2500), lowering kernel.perf_event_m
[ 1467.625133] Hello world!
sitrc@tushar:~/hello$
```

Output of hello module

Check kernel ring buffer

- **`dmesg | tail -1`**



Removing the module

- **rmmod hello.ko**
- **dmesg** **or**
- **dmesg | tail -1**

What is dmesg?

- The dmesg command is used to write the kernel messages in Linux and other Unix-like operating systems to standard output (which by default is the display screen).
- dmesg obtains its data by reading the kernel ring buffer. A buffer is a portion of a computer's memory that is set aside as a temporary holding place for data that is being sent to or received from an external device, such as a hard disk drive (HDD), printer or keyboard.
- A ring buffer is a buffer of fixed size for which any new data added to it overwrites the oldest data in it.

What is printk?

- The kernel print function, printk(), behaves almost identically to the C library printf() function.
- printk( ) is simply the name of the kernel's formatted print function. It is callable from just about anywhere in the kernel at any time.
- The major difference between printk() and printf() is the capability of the former to specify a loglevel.
- The kernel uses the loglevel to decide whether to print the message to the console. The kernel displays all messages with a loglevel below a specified value on the console.

printk : example

```
printk(KERN_WARNING "This is a warning!\n");
printk(KERN_DEBUG "This is a debug notice!\n");
printk("I did not specify a loglevel!\n");
```

Thank you

Web Resources
http://tusharkute.com

Blogs
http://digitallocha.blogspot.in
http://kyamputar.blogspot.in

tushar@tusharkute.com