Linux Kernel Module

How to write, load, and unload a Kernel module

- Write a Kernel module
- Load a Kernel module
- Remove/unload a Kernel module

List all kernel modules that are currently loaded

- *lsmod*

- three columns: name, size,
  and where the module is being used

```
ueransim@ueransim:~$ lsmod
Module                  Size  Used by
xt_nat                 16384  4
xt_tcpudp              20480  4
veth                   32768  0
xt_conntrack           16384  2
xt_MASQUERADE          20480  2
nf_conntrack_netlink    49152  0
nfnetlink              20480  2 nf_conntrack_netlink
xfrm_user              40960  1
xfrm_algo              16384  1 xfrm_user
iptable_nat            16384  2
nf_nat                 49152  3 xt_nat,iptable_nat,xt_MASQUERADE
nf_conntrack          172032  5 xt_conntrack,nf_nat,xt_nat,nf_conntrack_netlink,xt_MASQUERADE
nf_defrag_ipv6         24576  1 nf_conntrack
nf_defrag_ipv4         16384  1 nf_conntrack
libcrc32c              16384  2 nf_conntrack,nf_nat
xt_addrtype            16384  2
iptable_filter         16384  1
bpfilter               16384  0
br_netfilter           28672  0
bridge                307200  1 br_netfilter
stp                    16384  1 bridge
llc                    16384  2 bridge,stp
aufs                  270336  0
overlay               151552  1
nls_iso8859_1          16384  1
snd_intel8x0           49152  2
snd_ac97_codec        155648  1 snd_intel8x0
ac97_bus               16384  1 snd_ac97_codec
```

A simple kernel program

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

/* This function is called when the module is loaded. */
int simple_init(void)
{
   printk(KERN_INFO "Loading Module\n");

   return 0;
}

/* This function is called when the module is removed. */
void simple_exit(void)
{
   printk(KERN_INFO "Removing Module\n");
}

/* Macros for registering module entry and exit points. */
module_init(simple_init);
module_exit(simple_exit);

MODULE_LICENSE("GPL");
MODULE_DESCRIPTION("Simple Module");
MODULE_AUTHOR("SGG");
```

inside the init()

Printk : Similar to printf statement

KERN-INFO :

module entry Point

module exit point

simple.c

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

int simple_init(void)
{
        printk(KERN_INFO "Loading Module\n");
        return 0;
}

void simple_exit(void)
{
        printk(KERN_INFO "Removing Module\n");
}

module_init(simple_init);
module_exit(simple_exit);


MODULE_LICENSE("GPL");
MODULE_DESCRIPTION("Simple Module");
MODULE_AUTHOR("SGG");
```

Create a Makefile

```
obj-m += simple.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
```

Makefile

```
obj-m += simple.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
```

Run the Makefile

- I
- o
- m

```
ueransim@ueransim:~$ make -I Makefile
make -C /lib/modules/5.15.0-84-generic/build M=/home/ueransim modules
make[1]: Entering directory '/usr/src/linux-headers-5.15.0-84-generic'
  CC [M]  /home/ueransim/simple.o
  MODPOST /home/ueransim/Module.symvers
  CC [M]  /home/ueransim/simple.mod.o
  LD [M]  /home/ueransim/simple.ko
  BTF [M] /home/ueransim/simple.ko
Skipping BTF generation for /home/ueransim/simple.ko due to unavailability of vmlinux
make[1]: Leaving directory '/usr/src/linux-headers-5.15.0-84-generic'
```

Load the module into kernel

- sudo insmod simple.ko

```
ueransim@ueransim:~$ sudo insmod simple.ko
ueransim@ueransim:~$ lsmod | grep "simple"
simple                 16384  0
ueransim@ueransim:~$
```

Remove the module

- sudo rmmod simple

```
ueransim@ueransim:~$ sudo rmmod simple
ueransim@ueransim:~$ lsmod | grep "simple"
ueransim@ueransim:~$
```

Assignment

- Kernel module that communicates with /proc file system

Code – part 1

```
/**
 * hello.c
 *
 * Kernel module that communicates with /proc file system.
 * */

#include <linux/init.h>
#include <linux/module.h>
#include <linux/kernel.h>
#include <linux/proc_fs.h>
#include <asm/uaccess.h>

#define BUFFER_SIZE 128

#define PROC_NAME "hello"
#define MESSAGE "Hello World\n"
```

Code – part 2

```
/**
 * Function prototypes
 */
static ssize_t proc_read(struct file *file, char *buf, size_t count,
loff_t *pos);

static struct proc_ops proc_ops = {
        .proc_read = proc_read,
};
```

Code – part 3

```
/* This function is called when the module is loaded. */
static int proc_init(void)
{

        // creates the /proc/hello entry
        // the following function call is a wrapper for
        // proc_create_data() passing NULL as the last argument
        proc_create(PROC_NAME, 0, NULL, &proc_ops);

        printk(KERN_INFO "/proc/%s created\n", PROC_NAME);

                return 0;
}
```

Code – part 4

```
/* This function is called when the module is removed. */
static void proc_exit(void) {

        // removes the /proc/hello entry
        remove_proc_entry(PROC_NAME, NULL);

        printk( KERN_INFO "/proc/%s removed\n", PROC_NAME);
}
```

Code – part 5

```
static ssize_t proc_read(struct file *file, char __user *usr_buf, size_t count, loff_t *pos)
{
        int rv = 0;
        char buffer[BUFFER_SIZE];
        static int completed = 0;

        if (completed) {
                completed = 0;
                return 0;
        }

        completed = 1;

        rv = sprintf(buffer, "Hello World\n");

        // copies the contents of buffer to userspace usr_buf
        copy_to_user(usr_buf, buffer, rv);

        return rv;
}
```

```
/**
 * This function is called each time the /proc/hello is read.
 *
 * This function is called repeatedly until it returns 0, so there
must be logic that ensures it ultimately returns 0 once it has
collected the data that is to go into the corresponding /proc
file.
 * params:
 *
 * file:
 * buf: buffer in user space
 * count:
 * pos:
 */
```

Coe – part 6

```
/* Macros for registering module entry and exit points. */
module_init( proc_init );
module_exit( proc_exit );

MODULE_LICENSE("GPL");
MODULE_DESCRIPTION("Hello Module");
MODULE_AUTHOR("SGG");
```

```
ueransim@ueransim:~$ make
make -C /lib/modules/5.15.0-84-generic/build M=/home/ueransim modules
make[1]: Entering directory '/usr/src/linux-headers-5.15.0-84-generic'
  CC [M]  /home/ueransim/hello_1.o
/home/ueransim/hello_1.c: In function ‘proc_read’:
/home/ueransim/hello_1.c:88:9: warning: ignoring return value of ‘copy_to_user’, declared with attribute warn_unused_result [-Wunused-result]
   88 |         copy_to_user(usr_buf, buffer, rv);
      |         ^~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
  MODPOST /home/ueransim/Module.symvers
  CC [M]  /home/ueransim/hello_1.mod.o
  LD [M]  /home/ueransim/hello_1.ko
  BTF [M] /home/ueransim/hello_1.ko
Skipping BTF generation for /home/ueransim/hello_1.ko due to unavailability of vmlinux
make[1]: Leaving directory '/usr/src/linux-headers-5.15.0-84-generic'
ueransim@ueransim:~$ sudo insmod hello_1.ko
[sudo] password for ueransim:
ueransim@ueransim:~$ lsmod | grep "hello"
hello_1                16384  0
ueransim@ueransim:~$ cat /proc/hello
Hello World
ueransim@ueransim:~$ cat /proc/hello_1
cat: /proc/hello_1: No such file or directory
ueransim@ueransim:~$
```