- You can list all kernel modules that are currently loaded by entering the command

  ```
  $ lsmod
  ```

- The following program illustrates a very basic kernel module that prints appropriate messages when the kernel module is loaded and unloaded.

Basic Module Structure

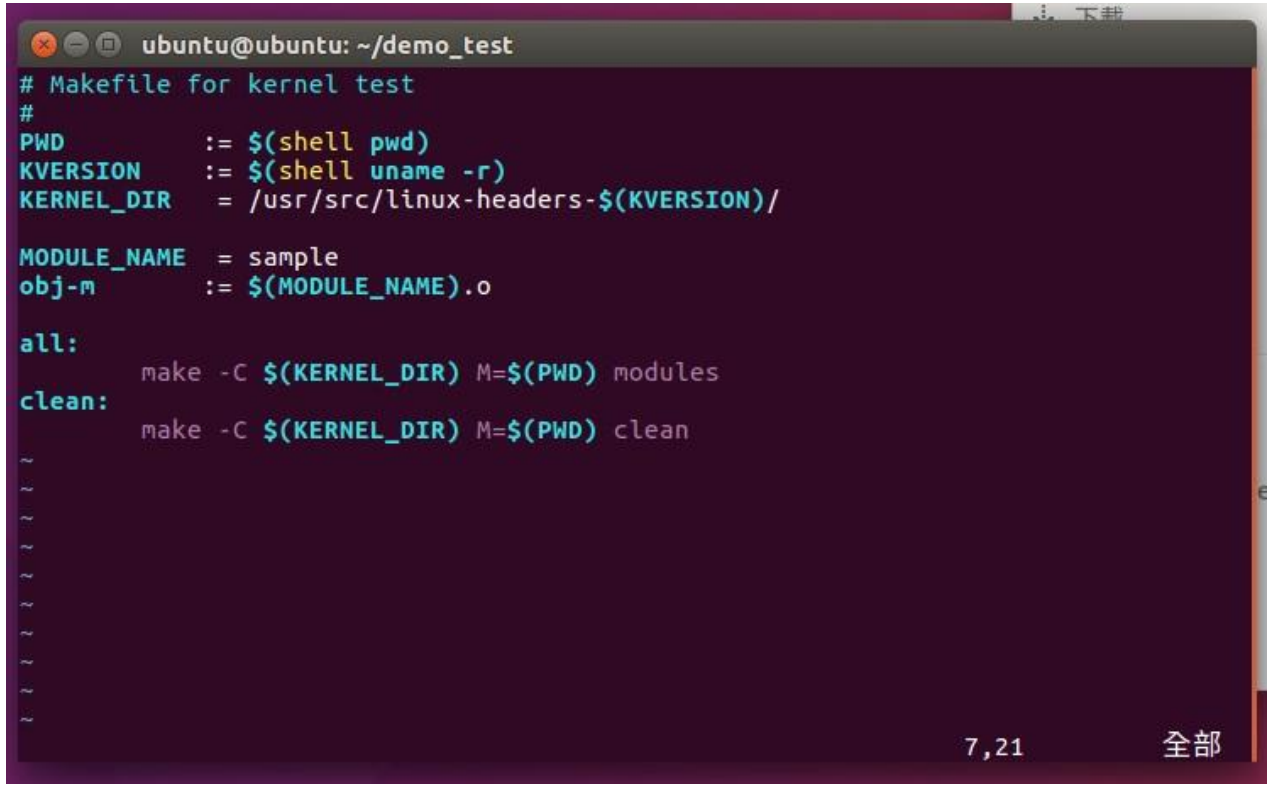
```
#include <linux/init.h>
#include <linux/kernel.h>
#include <linux/module.h>

/* 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");
```

Makefile

- This kernel module simple.c is compiled using the Makefile accompanying the source code with this project.

- To compile the module, enter the following on the command line:

```
$ make
```



- You should put your module and makefile in same directory.

```
osta@osta-VirtualBox:~$ make
make -C /usr/src/linux-headers-4.15.0-29-generic/ M=/home/osta modules
make[1]: Entering directory '/usr/src/linux-headers-4.15.0-29-generic'
Makefile:976: "Cannot use CONFIG_STACK_VALIDATION=y, please install libelf-dev,
libelf-devel or elfutils-libelf-devel"
  CC [M]  /home/osta/simple.o
  Building modules, stage 2.
  MODPOST 1 modules
  CC      /home/osta/simple.mod.o
  LD [M]  /home/osta/simple.ko
make[1]: Leaving directory '/usr/src/linux-headers-4.15.0-29-generic'
```

- After use make command you will get simple.ko.

```
osta@osta-VirtualBox:~$ ls
Desktop     examples.desktop  Module.symvers  Public      simple.mod.c  Templates
Documents   Makefile          Music           simple.c    simple.mod.o  Videos
Downloads   modules.order     Pictures        simple.ko   simple.o
```

Load kernel modules

- Using the *insmod* command to load the kernel modules.

  $ sudo insmod simple.ko

- Also use the *rmmod* command to remove modules.

  $ sudo rmmod simple.ko

- Use *dmesg* to check the kernel log buffer

```
[ 3875.008997] Loading Module
[ 3875.008998] 107062555, 1-1-1994
[ 3875.008999] 107065510, 8-4-1994
[ 3875.009000] 107062031, 15-7-1994
[ 3875.009001] 107065531, 22-10-1994
[ 3875.009002] 107065513, 29-13-1994
[ 3884.620973] Removing Module
[ 3884.620974] freeing node 107062555
[ 3884.620975] freeing node 107065510
[ 3884.620975] freeing node 107062031
[ 3884.621046] freeing node 107065531
```

More information about Linux Kernel Module

https://jerrynest.io/how-to-write-a-linux-kernel-module/?fbclid=IwAR1IkA_K2ZJe7D9hoLQHBesnQHPG_ybUmlomnKeoh2kj73LfyedCGsmEpNU

Homework

- In the module entry point, create a linked list containing five struct birthday elements. Traverse the linked list and output its contents to the kernel log buffer. Invoke the dmesg command to ensure the list is properly constructed once the kernel module has been loaded.

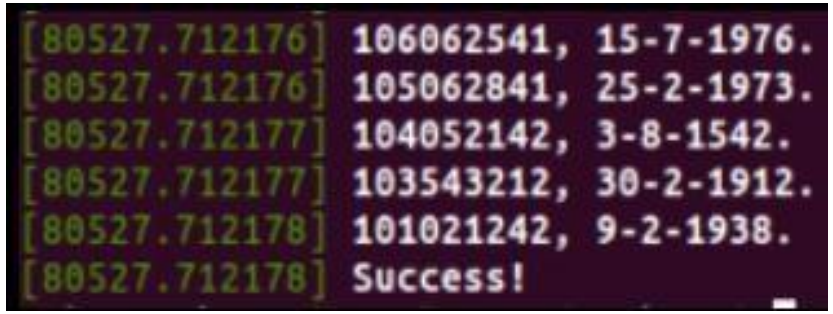
- In the module exit point, delete the elements from the linked list and return the free memory back to the kernel. Again, invoke the dmesg command to check that the list has been removed once the kernel module has been unloaded.

Notice

- Trace the include file <linux/list.h>
- Learn the doubly linked list structures provided by the Linux kernel
- Construct the linked list once the module is loaded
- Delete and free the linked list when the module is removed

Student_ID	Year	Month	Day
106062540	1976	..	..
106062899	..	..	..
106062569	..	..	..
106061359	..	..	..
106054893	..	..	..

Order should not be wrong



Report

- Explain your code
- Screenshot for your code and output

Grading

- Program 90%
- Linked list structure 70%
- Remove and free the space when removed 20%
- Report 10%
- Deadline : 3/21 (Thur.) 23:59
- No Delay is allowed !!
- 0 will be given to cheaters, do not copy & paste

- Upload：
- code
- Report (StudentID.pdf)
- Format：
- hw1_StudentID.zip
- We will choose ¼ students for demo.