- 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

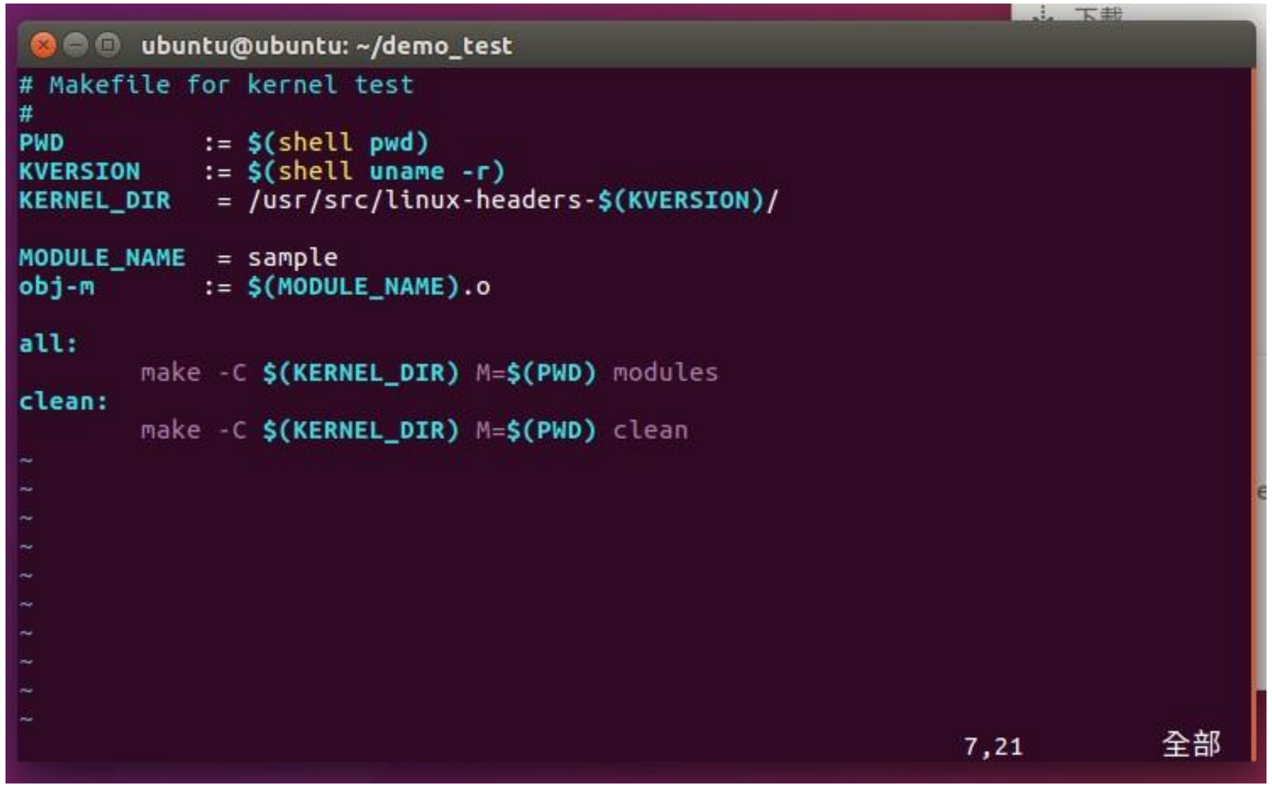
```c
#include <linux/init.h>
#include <linux/kernel.h>
#include <linux/module.h>

/* 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



```
# Makefile for kernel test
#
PWD          := $(shell pwd)
KVERSION     := $(shell uname -r)
KERNEL_DIR    = /usr/src/linux-headers-$(KVERSION)/

MODULE_NAME   = sample
obj-m        := $(MODULE_NAME).o

all:
        make -C $(KERNEL_DIR) M=$(PWD) modules
clean:
        make -C $(KERNEL_DIR) M=$(PWD) clean
~
~
~
~
~
~
~
~
~
~
                                                              7,21           全部
```

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

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.

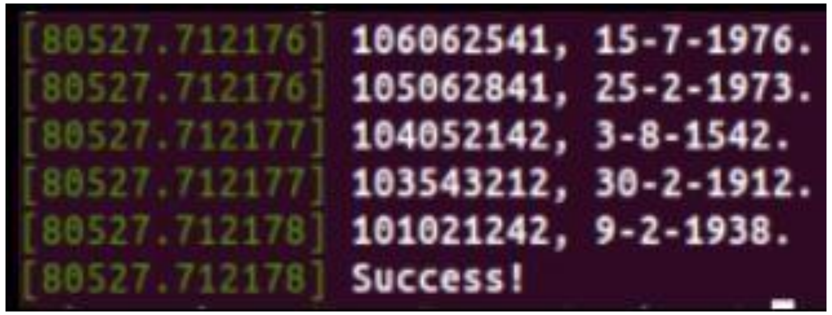
- In the module exit point, delete the elements from the linked list and return the free memory back to the kernel. Again, invoke the dmesg command to check that the list has been removed once the kernel module has been unloaded.

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



```
[80527.712176] 106062541, 15-7-1976.
[80527.712176] 105062841, 25-2-1973.
[80527.712177] 104052142, 3-8-1542.
[80527.712177] 103543212, 30-2-1912.
[80527.712178] 101021242, 9-2-1938.
[80527.712178] Success!
```

Report

- Explain your code
- Screenshot for your code and output

Hints

- List_head, list_for_each, etc.
- Note that the linked list provided in **list.h is doubly and circularly.**
- There may be some difference in traversing the linked list when you build and delete the linked list.
- There are some memory operations which provided in another header file <linux/slab.h>, these operations may be needed when you build and deleted the linked list.
- Cautions: you cannot just build and delete the linked list with **pure C language, i.e. you have to use data type and functions defined in list.h.**

Grading

- Program: 90% (70% + 20%)
  - Linked list structure: 70%
  - Remove and free the space when removed: 20%
- Report: 10%
- Deadline : 3/26 (Thu.) 23:59
- No Delay is allowed !!
- 0 will be given to cheaters, do not copy & paste

- Upload：
  - code
  - Report (StudentID.pdf)
  - File name：hw1_StudentID.zip
    1. hw1.c
    2. report.docx

- We will choose ¼ students for demo.