OPERATING SYSTEM (CS3431)

LINUX KERNEL MODULES

2.□

26

TABLE OF CONTENTS

01
Creating Makefile

02
Makefile Code

03
Creating File For Kernel Structures

04
Programming Simple.c

05
make Command

06
Loading Kernel Module

07
DMESG Command

08
Removing Kernel Module

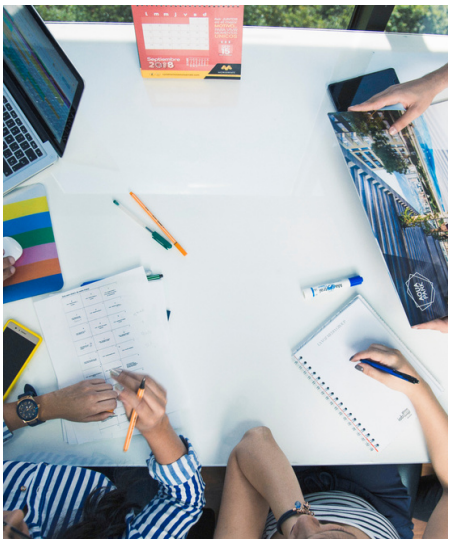
INTRODUCTION

In this project, you will learn how to create a kernel module and load it into the Linux kernel. The project can be completed using the Linux virtual machine that is available with this text. Although you may use an editor to write these C programs, you will have to use the terminal application to compile the programs, and you will have to enter commands on the command line to manage the modules in the kernel. As you'll discover, the advantage of developing kernel modules is that it is a relatively easy method of interacting with the kernel, thus allowing you to write programs that directly invoke kernel functions. It is important for you to keep in mind that you are indeed writing kernel code that directly interacts with the kernel. That normally means that any errors in the code could crash the system! However, since you will be using a virtual machine, any failures will at worst only require rebooting the system.

OUR GROUP MEMBERS

Saif-Ur-Rehman (BCS211119)
Roshail Azhar (BCS211130)
Kashan Mazhar (BCS211139)

Submitted to: Sir Siraj Rathore



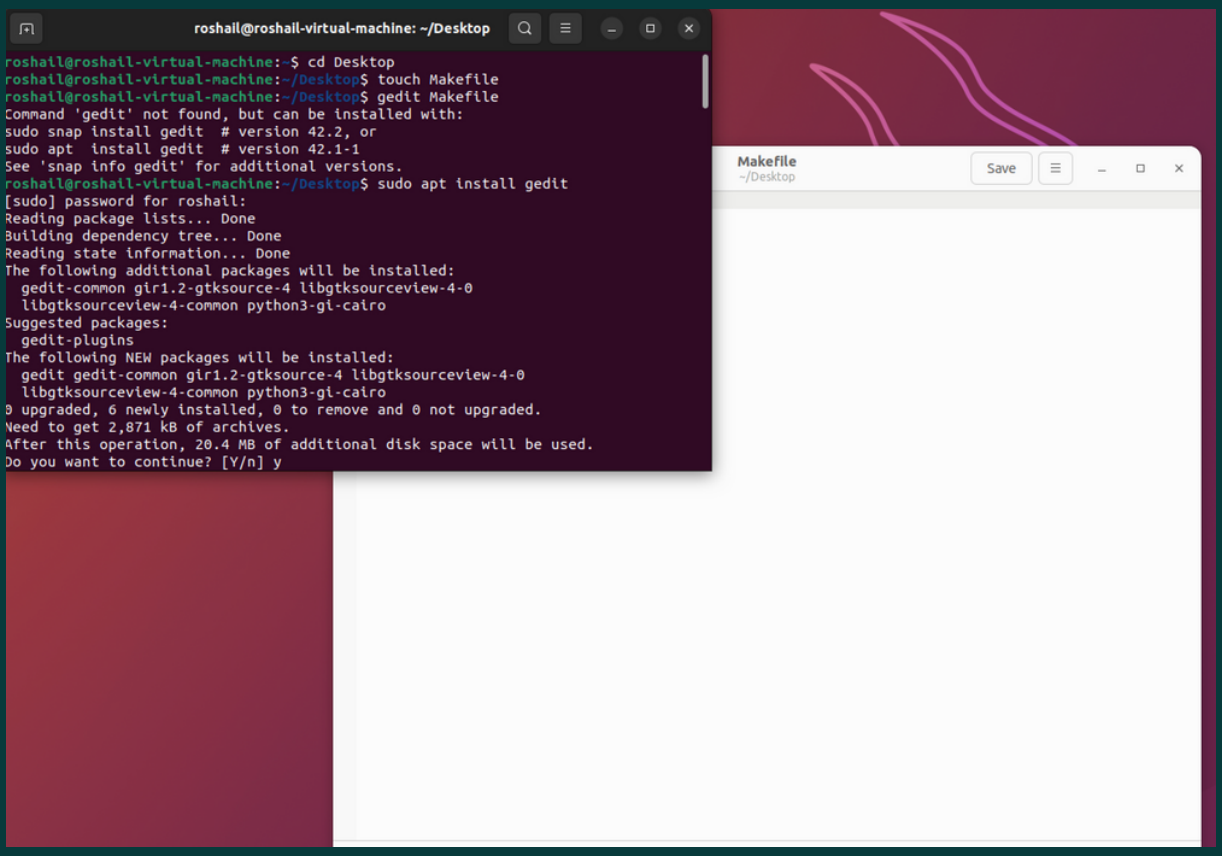
This Project is related to kernel modules and data structures.

CREATING KERNEL MODULES & BUILDING KERNEL DATA STURCTURES

The first part of this project involves following a series of steps for creating and inserting a module into the Linux kernel.

01 — Creating Make File

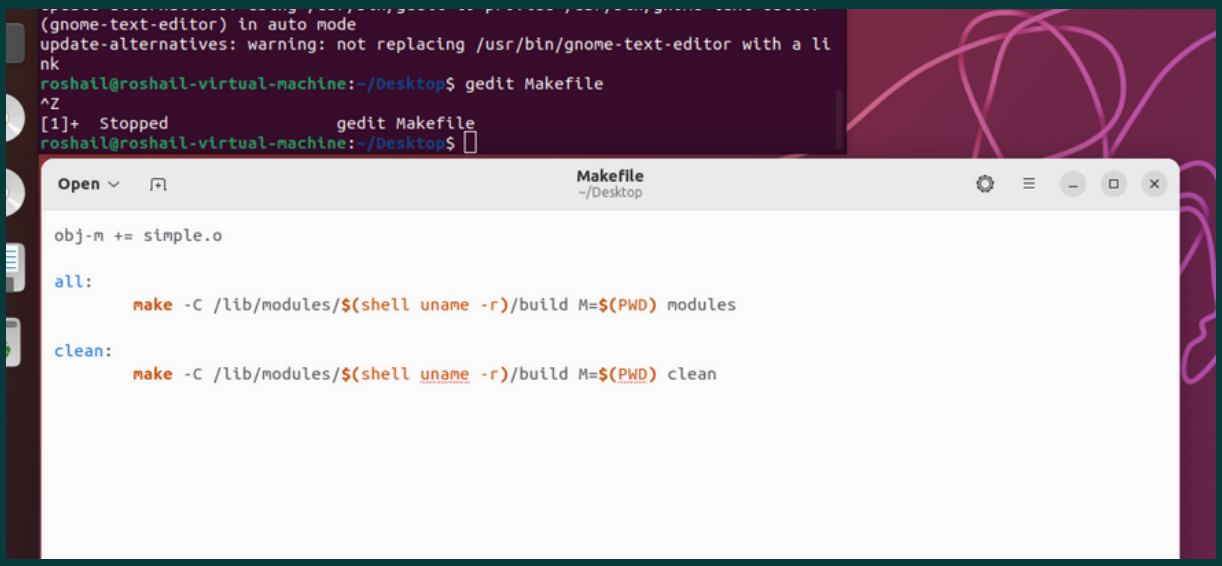
Go to terminal and type command: $ touch Makefile
This will create your Makefile



02 — Make File Code

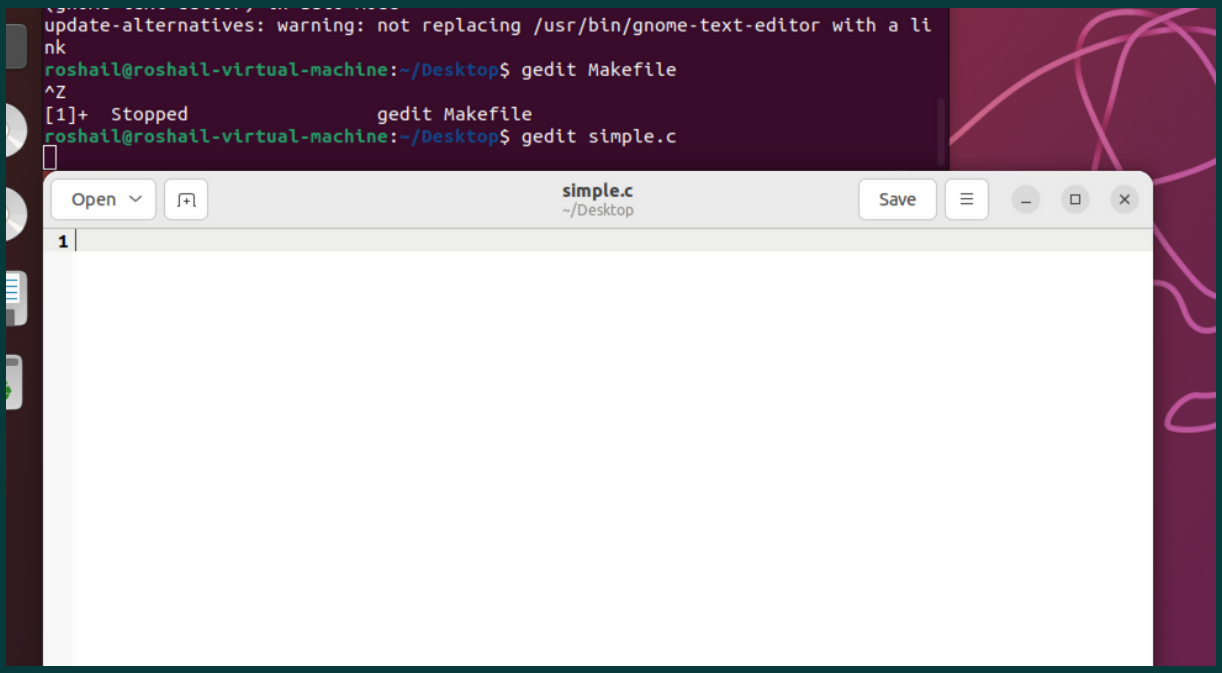
Code:

```
obj-m += simple.o

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

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



03 — Creating File For Kernel Structure

Go to terminal and type command: $ gedit simple.c
This will create your simple.c file



04 — Programming Simple.c

This part of this project involves modifying the kernel module so that it uses the kernel linked-list data structure.

```
update-alternatives: warning: not replacing /usr/bin/gnome-text-editor with a li
nk
roshail@roshail-virtual-machine:~/Desktop$ gedit Makefile
^Z
[1]+  Stopped                 gedit Makefile
roshail@roshail-virtual-machine:~/Desktop$ gedit simple.c
```

simple.c
~/Desktop

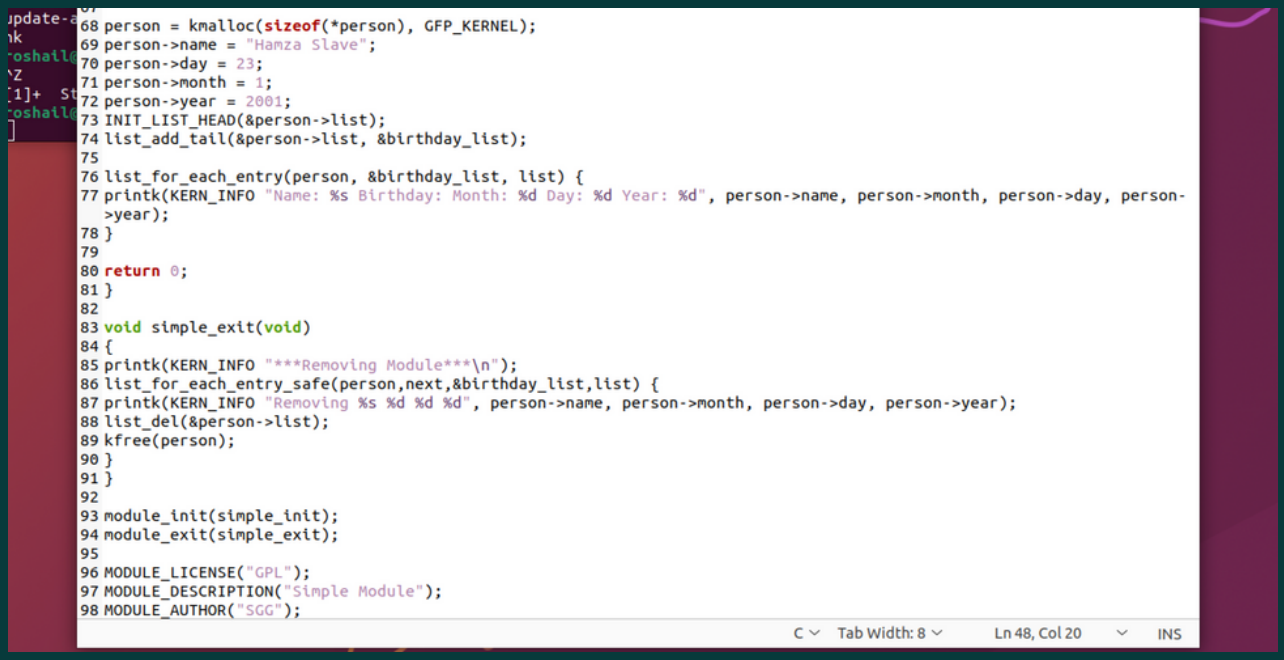
```c
#include <linux/init.h>
#include <linux/module.h>
#include <linux/kernel.h>
#include <linux/list.h>
#include <linux/slab.h>

struct birthday
{
int month;
int day;
int year;
char *name;
struct list_head list;
};

/**
* The following defines and initializes a list_head object named birthday_list
*/
static LIST_HEAD(birthday_list);
struct birthday *person;
struct birthday *next;


int simple_init(void)
```

C Tab Width: 8 Ln 48, Col 20 INS

```c
int simple_init(void)
{
printk(KERN_INFO "***Loading Module***\n");

person = kmalloc(sizeof(*person), GFP_KERNEL);
person->name = "Saif";
person->day = 29;
person->month = 7;
person->year = 2001;
INIT_LIST_HEAD(&person->list);
list_add_tail(&person->list, &birthday_list);

person = kmalloc(sizeof(*person), GFP_KERNEL);
person->name = "Roshail";
person->day = 8;
person->month = 11;
person->year = 2000;
INIT_LIST_HEAD(&person->list);
list_add_tail(&person->list, &birthday_list);

person = kmalloc(sizeof(*person), GFP_KERNEL);
person->name = "Zoha";
person->day = 18;
person->month = 12;
person->year = 1998;
INIT_LIST_HEAD(&person->list);
list_add_tail(&person->list, &birthday_list);

person = kmalloc(sizeof(*person), GFP_KERNEL);
person->name = "Kashan Sunny Mazhar";
person->day = 13;
person->month = 4;
person->year = 2001;
INIT_LIST_HEAD(&person->list);
list_add_tail(&person->list, &birthday_list);

person = kmalloc(sizeof(*person), GFP_KERNEL);
person->name = "Sufyan";
person->day = 23;
person->month = 1;
person->year = 1999;
INIT_LIST_HEAD(&person->list);
list_add_tail(&person->list, &birthday_list);

person = kmalloc(sizeof(*person), GFP_KERNEL);
```

C Tab Width: 8 Ln 48, Col 20 INS

04 — Continued



```
68 person = kmalloc(sizeof(*person), GFP_KERNEL);
69 person->name = "Hamza Slave";
70 person->day = 23;
71 person->month = 1;
72 person->year = 2001;
73 INIT_LIST_HEAD(&person->list);
74 list_add_tail(&person->list, &birthday_list);
75
76 list_for_each_entry(person, &birthday_list, list) {
77 printk(KERN_INFO "Name: %s Birthday: Month: %d Day: %d Year: %d", person->name, person->month, person->day, person-
   >year);
78 }
79
80 return 0;
81 }
82
83 void simple_exit(void)
84 {
85 printk(KERN_INFO "***Removing Module***\n");
86 list_for_each_entry_safe(person,next,&birthday_list,list) {
87 printk(KERN_INFO "Removing %s %d %d %d", person->name, person->month, person->day, person->year);
88 list_del(&person->list);
89 kfree(person);
90 }
91 }
92
93 module_init(simple_init);
94 module_exit(simple_exit);
95
96 MODULE_LICENSE("GPL");
97 MODULE_DESCRIPTION("Simple Module");
98 MODULE_AUTHOR("SGG");
```

Code:

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

struct birthday
{
int month;
int day;
int year;
char *name;
struct list_head list;
};

/**
* The following defines and initializes a list_head object named
birthday_list
*/
static LIST_HEAD(birthday_list);
struct birthday *person;
struct birthday *next;
```

04 — Continued

```
int simple_init(void)
{
printk(KERN_INFO "**Loading Module**\n");

person = kmalloc(sizeof(*person), GFP_KERNEL);
person->name = "Saif";
person->day = 29;
person->month = 7;
person->year = 2001;
INIT_LIST_HEAD(&person->list);
list_add_tail(&person->list, &birthday_list);

person = kmalloc(sizeof(*person), GFP_KERNEL);
person->name = "Kashan Mazhar";
person->day = 11;
person->month = 3;
person->year = 2000;
INIT_LIST_HEAD(&person->list);
list_add_tail(&person->list, &birthday_list);

person = kmalloc(sizeof(*person), GFP_KERNEL);
person->name = "Roshail";
person->day = 18;
person->month = 12;
person->year = 2001;
INIT_LIST_HEAD(&person->list);
list_add_tail(&person->list, &birthday_list);

person = kmalloc(sizeof(*person), GFP_KERNEL);
person->name = "Zoha";
person->day = 13;
person->month = 4;
person->year = 1999;
INIT_LIST_HEAD(&person->list);
list_add_tail(&person->list, &birthday_list);

person = kmalloc(sizeof(*person), GFP_KERNEL);
person->name = "Sufyan";
person->day = 23;
person->month = 1;
person->year = 2002;
INIT_LIST_HEAD(&person->list);
list_add_tail(&person->list, &birthday_list);
```

04 — Continued

```
list_for_each_entry(person, &birthday_list, list) {
printk(KERN_INFO "Name: %s Birthday: Month: %d Day: %d Year:
%d", person->name, person->month, person->day, person->year);
}

return 0;
}

void simple_exit(void)
{
printk(KERN_INFO "**Removing Module**\n");
list_for_each_entry_safe(person,next,&birthday_list,list) {
printk(KERN_INFO "Removing %s %d %d %d", person->name, person-
>month, person->day, person->year);
list_del(&person->list);
kfree(person);
}
}

module_init(simple_init);
module_exit(simple_exit);

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

04 — Continued

Code Synopsis:

Initially, we define a struct containing the elements that are to be inserted in the linked list. The following C struct defines birthdays:

struct birthday { int day; int month; int year; struct list head list; }

Inserting Elements into the Linked List

We declare a list head object, which we use as a reference to the head of the list by using the LIST_HEAD() macro

```
static LIST HEAD(birthday list);
```

This macro defines and initializes the variable birthday list, which is of type struct list_head.

We create and initialize instances of struct birthday as follows:

```
struct birthday *person;

person = kmalloc(sizeof(*person), GFP KERNEL);
person->day = 2; person->month= 8;
person->year = 1995;
INIT_LIST_HEAD(&person->list);
```

Traversing the Linked List:

- A pointer to the structure being iterated over
- A pointer to the head of the list being iterated over
- The name of the variable containing the list head structure

Removing Elements from the Linked List

Removing elements from the list involves using the list_del() macro, which is passed a pointer to struct list_head

```
list del(struct list_head *element)
```

This removes element from the list while maintaining the structure of the remainder of the list

04 — Continued

Code Synopsis Continued:

Additional Kernel Structure Requirements:

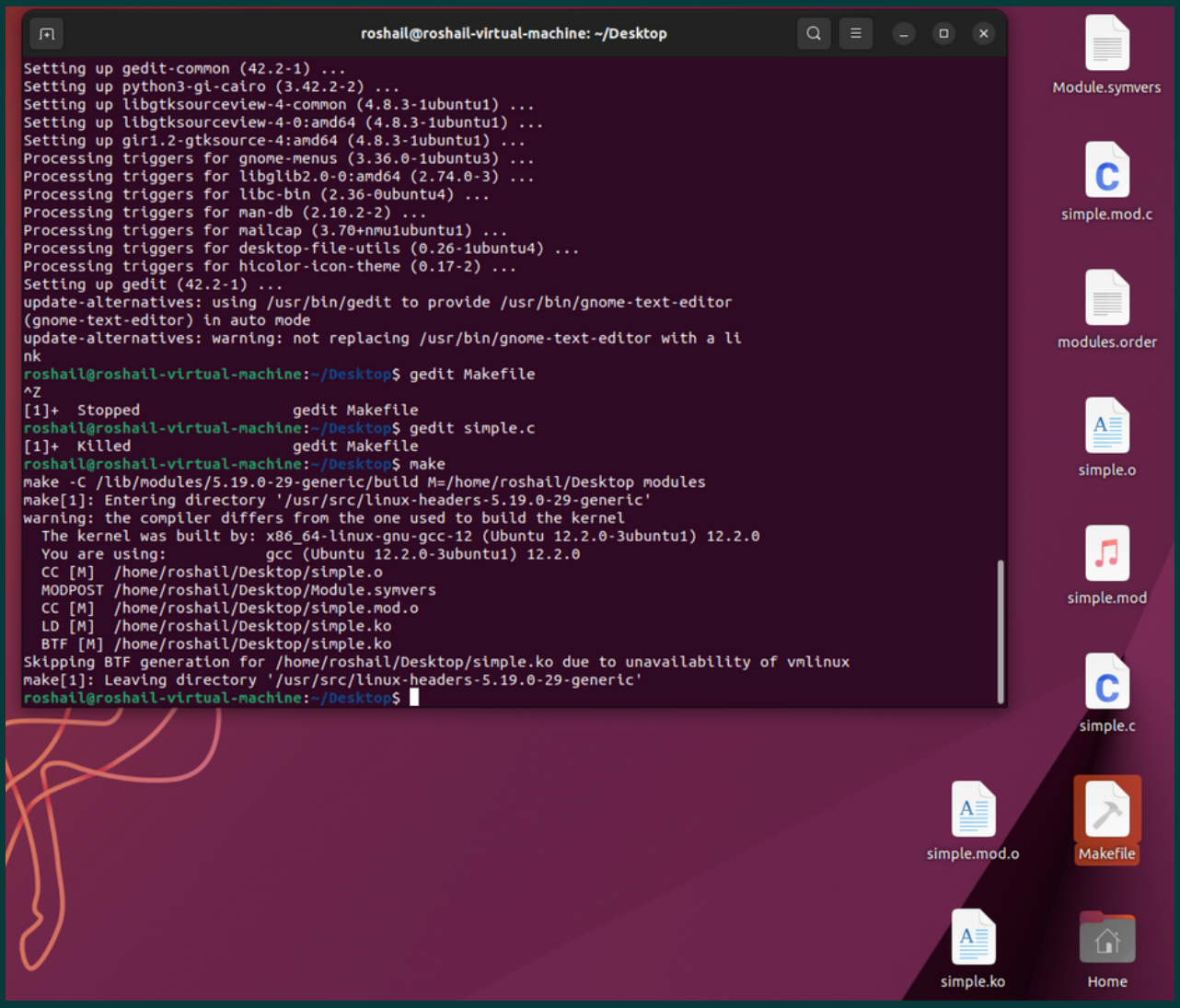
In the module entry point, we created a linked list containing five struct birthday elements. Traverse the linked list and output its contents to the kernel log buffer. Invoke the dmesg command to ensure the list is properly constructed once the kernel module has been loaded. In the module exit point,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

```
roshail@roshail-virtual-machine:~/Desktop$ sudo rmmod simple
roshail@roshail-virtual-machine:~/Desktop$ dmesg
[ 4913.717122] Removing Hamza Slave 1 23 2001
[ 4936.958704] ***Loading Module***
[ 4936.958708] Name: Saif Birthday: Month: 7 Day: 29 Year: 2001
[ 4936.958711] Name: Roshail Birthday: Month: 11 Day: 8 Year: 2000
[ 4936.958713] Name: Zoha Birthday: Month: 12 Day: 18 Year: 1998
[ 4936.958714] Name: Kashan Sunny Mazhar Birthday: Month: 4 Day: 13 Year: 2001
[ 4936.958716] Name: Sufyan Birthday: Month: 1 Day: 23 Year: 1999
[ 4936.958717] Name: Hamza Slave Birthday: Month: 1 Day: 23 Year: 2001
[ 4992.756014] ***Removing Module***
[ 4992.756018] Removing Saif 7 29 2001
[ 4992.756021] Removing Roshail 11 8 2000
[ 4992.756023] Removing Zoha 12 18 1998
[ 4992.756024] Removing Kashan Sunny Mazhar 4 13 2001
[ 4992.756026] Removing Sufyan 1 23 1999
roshail@roshail-virtual-machine:~/Desktop$
```

*Please Refer to Heading 8 for the complete understanding of the above image

05 — make Command

The compilation produces several files. The file simple.ko represents the compiled kernel module. The following step illustrates inserting this module into the Linux kernel.



06 — Loading Kernel Module

Kernel module loaded using the insmod command, which is run as follows:

To check whether the module has loaded, enter the lsmod command and search for the module simple. Recall that the module entry point is invoked when the module is inserted into the kernel

```
roshail@roshail-virtual-machine:~/Desktop$ sudo insmod simple.ko
roshail@roshail-virtual-machine:~/Desktop$ lsmod
Module                  Size  Used by
simple                 16384  0
isofs                  53248  2
bnep                   28672  2
intel_rapl_msr         20480  0
intel_rapl_common      40960  1 intel_rapl_msr
crct10dif_pclmul       16384  1
ghash_clmulni_intel    16384  0
btusb                  61440  0
```

07 — dmesg

To check the contents of this message in the kernel log buffer, enter the command

```
roshail@roshail-virtual-machine:~/Desktop$
roshail@roshail-virtual-machine:~/Desktop$ sudo insmod simple.ko
roshail@roshail-virtual-machine:~/Desktop$ dmesg
[ 4913.717122] Removing Hamza Slave 1 23 2001
[ 4936.958704] ***Loading Module***
[ 4936.958708] Name: Saif Birthday: Month: 7 Day: 29 Year: 2001
[ 4936.958711] Name: Roshail Birthday: Month: 11 Day: 8 Year: 2000
[ 4936.958713] Name: Zoha Birthday: Month: 12 Day: 18 Year: 1998
[ 4936.958714] Name: Kashan Sunny Mazhar Birthday: Month: 4 Day: 13 Year: 2001
[ 4936.958716] Name: Sufyan Birthday: Month: 1 Day: 23 Year: 1999
roshail@roshail-virtual-machine:~/Desktop$
```

08 — Removing Module

Removing the kernel module involves invoking the rmmod command

Be sure to check with the dmesg command to ensure the module has been removed. Because the kernel log buffer can fill up quickly, it often makes sense to clear the buffer periodically. This can be accomplished as follows:

sudo dmesg -c

```
roshail@roshail-virtual-machine:~/Desktop$ sudo rmmod simple
roshail@roshail-virtual-machine:~/Desktop$ dmesg
[ 4913.717122] Removing Hamza Slave 1 23 2001
[ 4936.958704] ***Loading Module***
[ 4936.958708] Name: Saif Birthday: Month: 7 Day: 29 Year: 2001
[ 4936.958711] Name: Roshail Birthday: Month: 11 Day: 8 Year: 2000
[ 4936.958713] Name: Zoha Birthday: Month: 12 Day: 18 Year: 1998
[ 4936.958714] Name: Kashan Sunny Mazhar Birthday: Month: 4 Day: 13 Year: 2001
[ 4936.958716] Name: Sufyan Birthday: Month: 1 Day: 23 Year: 1999
[ 4936.958717] Name: Hamza Slave Birthday: Month: 1 Day: 23 Year: 2001
[ 4992.756014] ***Removing Module***
[ 4992.756018] Removing Saif 7 29 2001
[ 4992.756021] Removing Roshail 11 8 2000
[ 4992.756023] Removing Zoha 12 18 1998
[ 4992.756024] Removing Kashan Sunny Mazhar 4 13 2001
[ 4992.756026] Removing Sufyan 1 23 1999
roshail@roshail-virtual-machine:~/Desktop$
```