石育瑋 108062633

CS 342300 Operating System

OS HW1: Kernel Module

Implementation

1. List node structure

```
typedef struct student
{
    int id;
    char *birthday;
    struct list_head node_student;
} student_t;
```

根據 spec，自訂一個結構包含學生的 id 生日，還有一個 list_head 結構紀錄前一個節點與後一個節點.

2. Self defined **strcopy**

```
void my_strcpy(char *dst, const char *src)
{
    char *tmp_src = src, *tmp_dst = dst;

    while (*tmp_src != '\0')
    {
        *tmp_dst = *tmp_src;
        tmp_src++;
        tmp_dst++;
    }
    *tmp_dst = '\0';
}
```

因為一般的”strcpy”無法在 kernel 中使用，所以自解寫了一個簡單的複製 string 函數去複製學生的生日.

3. List node initializing

```
student_t *construct_student(const int id, const char *birthday)
{
    student_t *s;
    s = kmalloc(1 * sizeof(student_t), GFP_KERNEL);
    s->birthday = kmalloc(30 * sizeof(char), GFP_KERNEL);

    my_strcpy(s->birthday, birthday);
    // char buf[] = KERN_INFO " %s";
    // printk(buf, s->birthday);
    s->id = id;

    return s;
```

初始化節點需要使用 kmalloc 申請記憶體空間，需要申請的空間有 student 結構本身記憶體大小，與紀錄學生生日所需要的記憶體空間.

4. Contruct list & print student info

```
for (i = 0; i < 5; i++)
{
    tmp_s = construct_student(id_list[i], birth_list[i]);
    list_add_tail(&tmp_s->node_student, &class);
}
char buf[] = KERN_INFO "%d, %s.\n";
list_for_each_entry(tmp_s, &class, node_student)
{
    printk(buf, tmp_s->id, tmp_s->birthday);
}
```

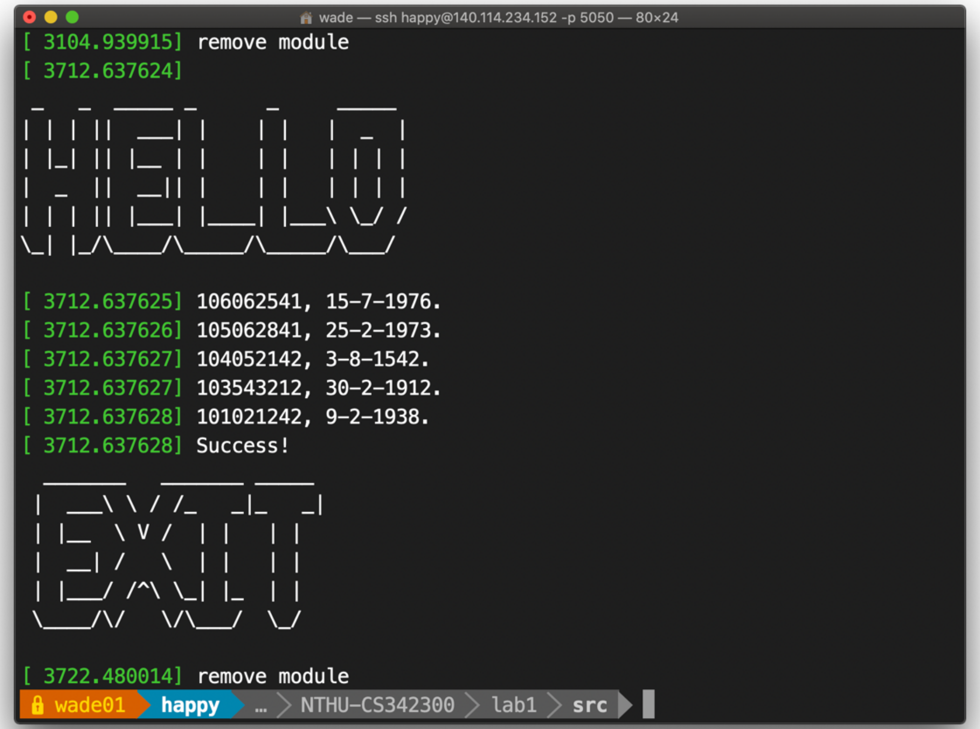
根據 spec 定義五個學生的 id, birthday array，遍歷兩個 array 初始化作業要求的 linked list，初始化完成後再用 list_for_each_entry teaverse linked list 中每一個節點，並 output 節點中儲存的資訊至 kernel log 中.

5. Release memory

```
student_t *tmp_s, *_tmp_s;
list_for_each_entry_safe(tmp_s, _tmp_s, &class, node_student)
{
    list_del(&tmp_s->node_student);
    kfree(tmp_s->birthday);
    kfree(tmp_s);
}
```

使用 kfree 釋放記憶體空間，因為刪除節點會刪除該節點的 list_head 導致 linked list 斷掉，所以需要使用 list_for_each_entry_safe 確保 traverse 時不會因為刪除 node 而無法完成遍歷.

Result



```
[ 3104.939915] remove module
[ 3712.637624]
 _   _ _____ _      _     _____
| | | ||  ___| |    | |   |  _  |
| |_| || |__  | |    | |   | | | |
|  _  ||  __|| |    | |   | | | |
| | | || |___| |____| |___\ \_/ /
\_| |_/\____/\_____/\_____/\___/

[ 3712.637625] 106062541, 15-7-1976.
[ 3712.637626] 105062841, 25-2-1973.
[ 3712.637627] 104052142, 3-8-1542.
[ 3712.637627] 103543212, 30-2-1912.
[ 3712.637628] 101021242, 9-2-1938.
[ 3712.637628] Success!
  ______   ______ _____
 |  ___\ \ / /_   _|_   _|
 | |__  \ V /  | |   | |
 |  __| /   \  | |   | |
 | |___/ /^\ \_| |_  | |
 \____/\/   \/\___/  \_/

[ 3722.480014] remove module
wade01  happy  … NTHU-CS342300  lab1  src
```

Reference

[1] The Linux Kernel API (https://www.kernel.org/doc/htmldocs/kernel-api/Appendix)

Appendix (hw.c)

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

typedef struct student
{
    int id;
    char *birthday;
    struct list_head node_student;
} student_t;

void my_strcpy(char *dst, const char *src)
{
    char *tmp_src = src, *tmp_dst = dst;

    while (*tmp_src != '\0')
    {
        *tmp_dst = *tmp_src;
        tmp_src++;
        tmp_dst++;
    }
    *tmp_dst = '\0';
}

student_t *construct_student(const int id, const char *birthday)
{
    student_t *s;
    s = kmalloc(1 * sizeof(student_t), GFP_KERNEL);
    s->birthday = kmalloc(30 * sizeof(char), GFP_KERNEL);

    my_strcpy(s->birthday, birthday);
    // char buf[] = KERN_INFO " %s";
    // printk(buf, s->birthday);
    s->id = id;

    return s;
}

struct list_head class;
char buf_info[] = KERN_INFO " %s";

// init function
int hw_init(void)
```

```
{
    char *welcome = "\n\r _   _  _____ _      _     _____ \n \
                     \r| | | ||  ___| |    | |   |  _  |\n \
                     \r| |_| || |__ | |    | |   | | | |\n \
                     \r|  _  ||  __|| |    | |   | | | |\n \
                     \r| | | || |___| |____| |___\\ \\_/ /\n \
                     \r\\_| |_/\\____/\\_____/\\_____/\\___/ \n \
                     ";
    printk(buf_info, welcome);
    // init list head
    INIT_LIST_HEAD(&class);

    student_t *tmp_s;
    int id_list[5] = {106062541, 105062841, 104052142,
                      103543212, 101021242};
    char *birth_list[5] = {"15-7-1976", "25-2-1973", "3-8-1542",
                           "30-2-1912", "9-2-1938"};
    int i;
    for (i = 0; i < 5; i++)
    {
        tmp_s = construct_student(id_list[i], birth_list[i]);
        list_add_tail(&tmp_s->node_student, &class);
    }
    char buf[] = KERN_INFO "%d, %s.\n";
    list_for_each_entry(tmp_s, &class, node_student)
    {
        printk(buf, tmp_s->id, tmp_s->birthday);
    }
    printk(KERN_INFO "Success!\n");

    return 0;
}

void hw_exit(void)
{
    char *exit = "\r  _______   _______ _____ \n \
                  \r |  ___\\ \\ / /_   _|_   _|\n \
                  \r | |__  \\ V /  | |   | |  \n \
                  \r |  __| /   \\  | |   | |  \n \
                  \r | |___/ /^\\ \\_| |_  | |  \n \
                  \r \\____/\\/   \\/\\___/  \\_/  \n \
                  ";
    printk(buf_info, exit);

    student_t *tmp_s, *_tmp_s;
    list_for_each_entry_safe(tmp_s, _tmp_s, &class, node_student)
    {
        list_del(&tmp_s->node_student);
        kfree(tmp_s->birthday);
        kfree(tmp_s);
    }

    printk(KERN_INFO "remove module\n");
}

module_init(hw_init);
module_exit(hw_exit);
```