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．

1. 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．

1. 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．

1. 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．

1. 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);

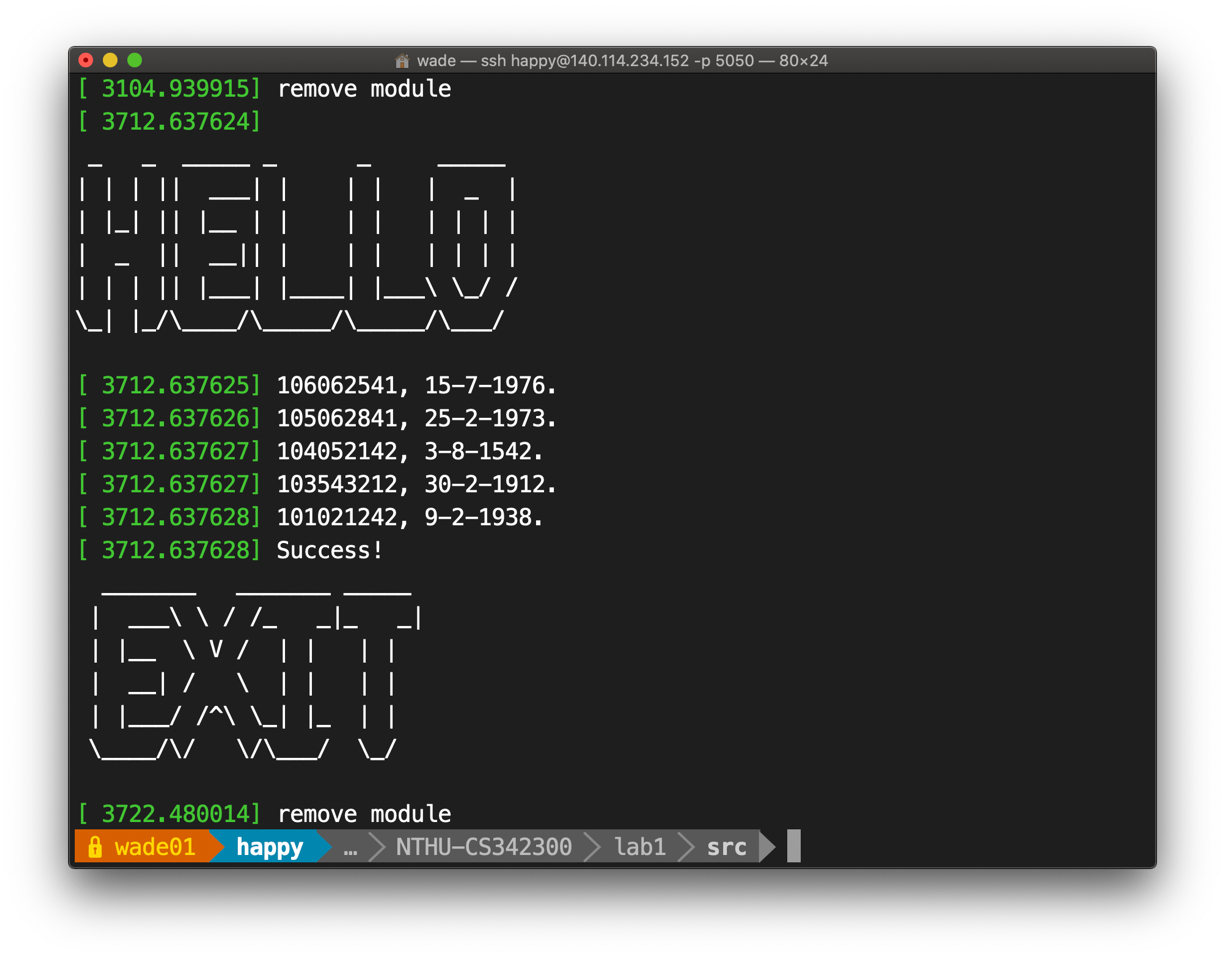
kfree(tmp\_s->birthday);

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**

**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);