CS 4540 – Operating Systems

Instructor: Lilien, Western Michigan University

Justin Lanyon

**SOFTWARE LIFE CYCLE REPORT – FOR ASSIGNMENT 04**

**PHASE 1: SPECIFICATION (“What do we build?”)**

**Entry point**. In the module entry point, create a linked list containing six struct student elements for a single students, that is with the same value of studentNumber(and different values of courseCredit and grade). Traverse the linked list and output its contents to the kernel log buffer. After outputting contents to the log buffer, calculate the current GPA for the student and output the result to the kernel log buffer. Invoke the dmesg command to ensure the list is properly constructed once the kernel module has been loaded.

**Exit point**. 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.

**PHASE 2: DESIGN**

**2.1 Modules and Their Basic Structure**

simple\_init method: creates all student record structs and adds them to the linked list. Then runs throuh the list printing them and calculates gpa and prints.

simple\_exit method: goes through links and deletes them then frees their memory for the kernel.

**Phase 3: RISK ANALYSIS**

There are no risks (to cost, human health, timetable) are identifiable.

**PHASE 4: VERIFICATION**

This algorithm has a very standard path that has been verified for its correctness by me through analyzing its steps in comparison to the specifications.

**PHASE 5: CODING**

**5a) Code Refinement #1(class structure with pseudocode only;)**

/\* INCLUDES\*/

/\*module initialize method\*/

/\*module exit method\*/

/\*module license description author\*/

**5b) Code Refinement #2**

/\* INCLUDES\*/

/\*struct for student info\*/

/\*module initialize method\*/

/\* create 6 links in list\*/

/\*print links and gpa\*/

/\*module exit method\*/

/\*delete links\*/

/\*module license description author\*/

**5c) Code Refinement #3**

/\* INCLUDES\*/

#include <linux/init.h>

#include <linux/kernel.h>

#include <linux/module.h>

#include <linux/slab.h>

#include <linux/list.h>

/\*struct for student info\*/

/\*module initialize method\*/

int simple\_init(void)

{

/\* create 6 links in list\*/

/\*print links and gpa\*/

}

/\*module exit method\*/

int simple\_exit(void)

{

/\*delete links\*/

}

/\*module license description author\*/

module\_init(simple\_init);

module\_exit(simple\_exit);

MODULE\_LICENSE("GPL");

MODULE\_DESCRIPTION("Simple Module");

MODULE\_AUTHOR("SGG");

**5d) Code Refinement #4 (complete program--with complete fields/properties, code for constructor/methods)**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Course: CS 4540 – Fall 2015

\* Assignment 4

\* Name: Justin Lanyon

\* E-mail: justin.j.lanyon@wmich.edu

\* Submitted: 11/23/2015

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* INCLUDES\*/

#include <linux/init.h>

#include <linux/kernel.h>

#include <linux/module.h>

#include <linux/slab.h>

#include <linux/list.h>

/\*struct for student info\*/

struct student {

int studentNumber;

int courseCredit;

float grade;

struct list\_head list;

};

/\*module initialize method\*/

int simple\_init(void)

{

/\* create 6 links in list\*/

/\*print links and gpa\*/

float totalPoint=0;

int totalHour=0;

int beforeDec =0;

int afterDec =0;

float gpa =0;

float temp =0;

printk(KERN\_INFO "Loading Module\n");

/\*1st Class\*/

struct student \*studentDemo;

studentDemo = kmalloc(sizeof(\*studentDemo), GFP\_KERNEL);

studentDemo->studentNumber = 760120495;

studentDemo->courseCredit = 3;

studentDemo->grade = 3.0;

INIT\_LIST\_HEAD(&studentDemo->list);

list\_add\_tail(&studentDemo->list, &student\_list);

/\*2nd Class\*/

struct student \*studentDemo2;

studentDemo2 = kmalloc(sizeof(\*studentDemo2), GFP\_KERNEL);

studentDemo2->studentNumber = 760120495;

studentDemo2->courseCredit = 2;

studentDemo2->grade = 1.0;

list\_add\_tail(&studentDemo2->list, &student\_list);

/\*3rd Class\*/

struct student \*studentDemo3;

studentDemo3 = kmalloc(sizeof(\*studentDemo3), GFP\_KERNEL);

studentDemo3->studentNumber = 760120495;

studentDemo3->courseCredit = 4;

studentDemo3->grade = 3.5;

list\_add\_tail(&studentDemo3->list, &student\_list);

/\*4th Class\*/

struct student \*studentDemo4;

studentDemo4 = kmalloc(sizeof(\*studentDemo4), GFP\_KERNEL);

studentDemo4->studentNumber = 760120495;

studentDemo4->courseCredit = 1;

studentDemo4->grade = 3.5;

list\_add\_tail(&studentDemo4->list, &student\_list);

/\*5th Class\*/

struct student \*studentDemo5;

studentDemo5 = kmalloc(sizeof(\*studentDemo5), GFP\_KERNEL);

studentDemo5->studentNumber = 760120495;

studentDemo5->courseCredit = 2;

studentDemo5->grade = 3.0;

list\_add\_tail(&studentDemo5->list, &student\_list);

/\*6th Class\*/

struct student \*studentDemo6;

studentDemo6 = kmalloc(sizeof(\*studentDemo6), GFP\_KERNEL);

studentDemo6->studentNumber = 760120495;

studentDemo6->courseCredit = 3;

studentDemo6->grade = 2.0;

list\_add\_tail(&studentDemo6->list, &student\_list);

struct student \*ptr;

list\_for\_each\_entry(ptr, &student\_list, list) {

/\* on each iteration ptr points \*/

/\* to the next student struct \*/

temp = ptr->grade;

beforeDec = temp;

temp = temp-beforeDec;

afterDec = temp\*100;

printk("Student: %d Course Credit: %d Grade: %d.%d\n",ptr->studentNumber,

ptr->courseCredit, beforeDec, afterDec);

totalHour += ptr->courseCredit;

totalPoint += (ptr->courseCredit \* ptr->grade);

}

//printk("totalHour = %d\n",totalHour);

gpa = totalPoint/totalHour;

beforeDec =0;

beforeDec =gpa;

gpa = gpa-beforeDec;

afterDec = gpa\*100;

//printk("totalPoint = %d.%d\n",beforeDec,afterDec);

printk("Student: %d GPA %d.%d\n",studentDemo->studentNumber, beforeDec, afterDec);

return 0;

}

/\*module exit method\*/

int simple\_exit(void)

{

/\*delete links\*/

struct student \*ptr, \*next;

printk(KERN\_INFO "Removing Module\n");

list\_for\_each\_entry\_safe(ptr, next, &student\_list, list){

list\_del(&ptr->list);

kfree(ptr);

}

return 0;

}

/\*module license description author\*/

module\_init(simple\_init);

module\_exit(simple\_exit);

MODULE\_LICENSE("GPL");

MODULE\_DESCRIPTION("Simple Module");

MODULE\_AUTHOR("SGG");

**PHASE 6: TESTING**

One can test it by changing some of the grades or credit hours, then running the makefile and inserting the new kernel object and checking dmesg.

**PHASE 7: REFINING THE PROGRAM**

No refinements are needed. In this program, I have already included all required features!

**PHASE 8: PRODUCTION**

I prepared a copy of the entire program for Lab TA’s evaluation, as specified by the TA.! Then, I sent

electronically the copy to the Lab TA.

**PHASE 9: MAINTENANCE**

To fully benefit from the program evaluation feedback received from the Lab TA, I will perform program

maintenance. This means that I should use all TAs feedback to improve my program.