| 과목명 | 운영체제 | 분반 | X | 담당교수 | 김화성 교수님 |
|---|---|---|---|---|---|
| 학과 | 전자통신공학과 | 학번 | 2016707079 | 이름 | 하상천 |
| 과제명: H/W3 - 리눅스 커널 모듈 프로그래밍 실습 및 과제 1 | | | | | |

1. 과제설명 ( 사용자 요구사항 기술: 과제에 대한 설명 및 목표 )

## 실습 1. "Hello World, Goodbye World" 출력 실습

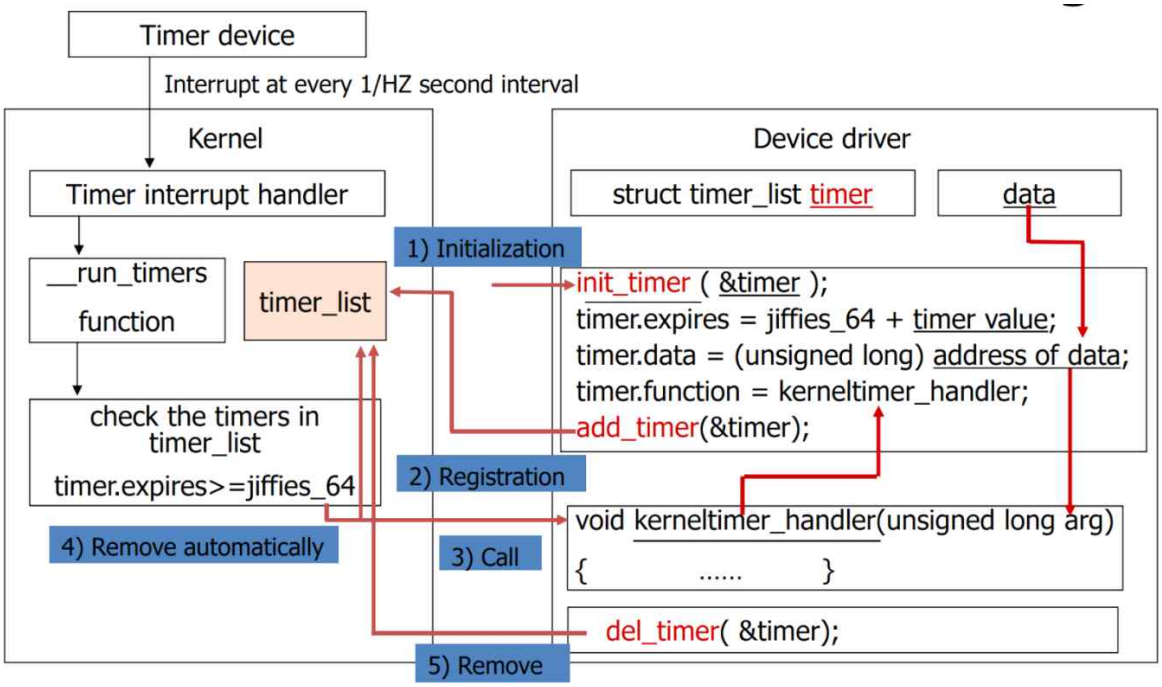
- 실습 내용은 결과만 출력 하여 보고서에 첨부

- 커널 로그에 로드 시 "학번 + Hello World" 출력, 커널 언로드 시 "학번 + Goodbye World"를 출력 하는 실습

## 과제 1. 커널 모듈 프로그래밍에서의 타이머 모듈 구현

linux/timer.h 파일에서 제공하는 타이머 함수를 이용하여 타이머 모듈을 구현한다
- module_init 에서는 setup_timer를 사용하여 타이머를 초기화 하고 mod_timer를 호출하여 타이머를 시작함
- 타이머가 만료되면 콜백함수 my_timer_callback을 호출한다.
- 모듈을 제거하면 타이머를 삭제한다.

2. 사용자 요구사항을 정형적 방법으로 기술 (UML, Pseudo code, 그림등을 이용하여 기술)




커널 2.6 버전의 타이머 동작

3. 알고리즘 및 자료구조 설계 내용

| <linux/kernel.h> 선언 내용 | | | 우선 순위 |
|---|---|---|---|
| #define | KERN_EMERG | <0> | 우선 순위가 가장 높음, 시스템이 비정상적인 상황 |
| #define | KERN_ALERT | <1> | 반드시 바로 처리되어야 할 상황 |
| #define | KERN_CRIT | <2> | 심각(Critical)한 상황 |
| #define | KERN_ERR | <3> | 오류(Error)가 발생한 상황 |
| #define | KERN_WARNING | <4> | 경고(Warnng) 상황 |
| #define | KERN_NOTICE | <5> | 일반적이나 중요한 상황 |
| #define | KERN_INFO | <6> | 단순히 정보 전달 차원의 상황 |
| #define | KERN_DEBUG | <7> | 우선순위가 가장 낮음, 디버깅 레벨 |

커널 모듈 프로그래밍은 main() 함수가 필요 없다.
커널 모듈이 Load 되고, Unload 될 때 module_init() 함수와 module_exit() 함수가 호출된다.
커널 모듈은 독립적인 프로세스가 아니라, kernel context의 일부로 실행된다.

4. 소스코드 설명 ( 직접 작성한 소스코드중에 핵심 부분을 발췌하여 설명 )

```
void my_timer_callback(struct timer_list *timer)
{
        printk(KERN_NOTICE "%s called (%ld)₩n", __func__, jiffies);
}
```

타이머가 만료되면 콜백 함수(my_timer_callback)가 호출된다. 파라미터로는 커널 타이머 구조체의 주소값을 받는다. jiffies는 커널 타이머를 실행하는 시간 단위이고, HZ는 진동수이다. 예를 들어 HZ가 300이라고 하면 1초에 jiffies(지피스)가 300번 업데이트 된다.

```
static int __init timer_module_init(void)
{
        int ret;
        printk(KERN_NOTICE "%s : Timer module loaded!!!₩n", __func__);

        setup_timer(&my_timer, my_timer_callback, 0);
        printk(KERN_NOTICE "%s : Setup timer to fire in 1000ms (%ld)₩n", __func__,
jiffies);
        ret = mod_timer(&my_timer, jiffies + msecs_to_jiffies(1000));

        if(ret) printk(KERN_NOTICE "Error in mod_timer₩n");
        return 0;
}
```

전체 타이머 구조는 timer_module_init에서 setup_timer 함수를 사용하여 타이머를 초기화하고, mod_timer함수를 호출하여 타이머를 시작한다. 타이머가 만료되면 콜백 함수가 호출되고, 마지막으로 모듈을 제거하면 del_timer 함수를 통해 타이머가 삭제된다. mod_timer 함수의 반환값으로는 동적 타

이머 만료 시각을 설정했으면 0, 아닌 경우는 1을 반환한다. 따라서 타이머 설정을 하지 못했다면 if문에 걸려서 에러 메시지를 출력한다. 파라미터로는 동적 타이머를 표현하는 struct timer_list 구조체 주소와 동적 타이머 만료 시각(HZ단위)이다.

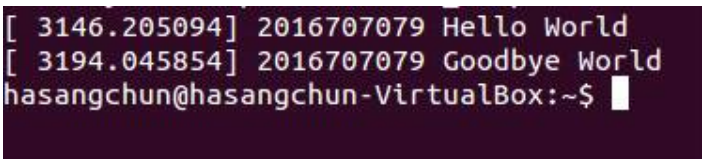
```
static void __exit timer_module_exit(void)
{
        int ret;
        ret = del_timer(&my_timer);
        if(ret) printk(KERN_NOTICE "The timer is still in use.₩n");
        printk(KERN_NOTICE "%s : Timer module unloaded!!!₩n"), __func__);
}
```

del_timer 함수를 통해 커널 타이머 목록에서 등록된 것을 제거한다. 반환 값으로는 타이머가 활성화 상태였다면 1을 반환하고, 그렇지 않은 경우는 0을 반환한다. 타이머가 활성화 상태이면 1을 반환하여 if문에 걸리게 되고 메시지를 출력한다.

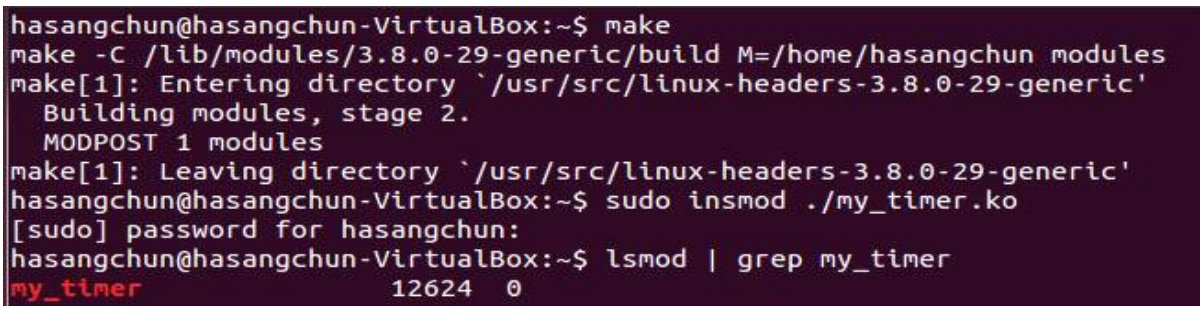
```
MODULE_LICENSE("GPL");
module_init(timer_module_init);
module_exit(timer_module_exit);
```

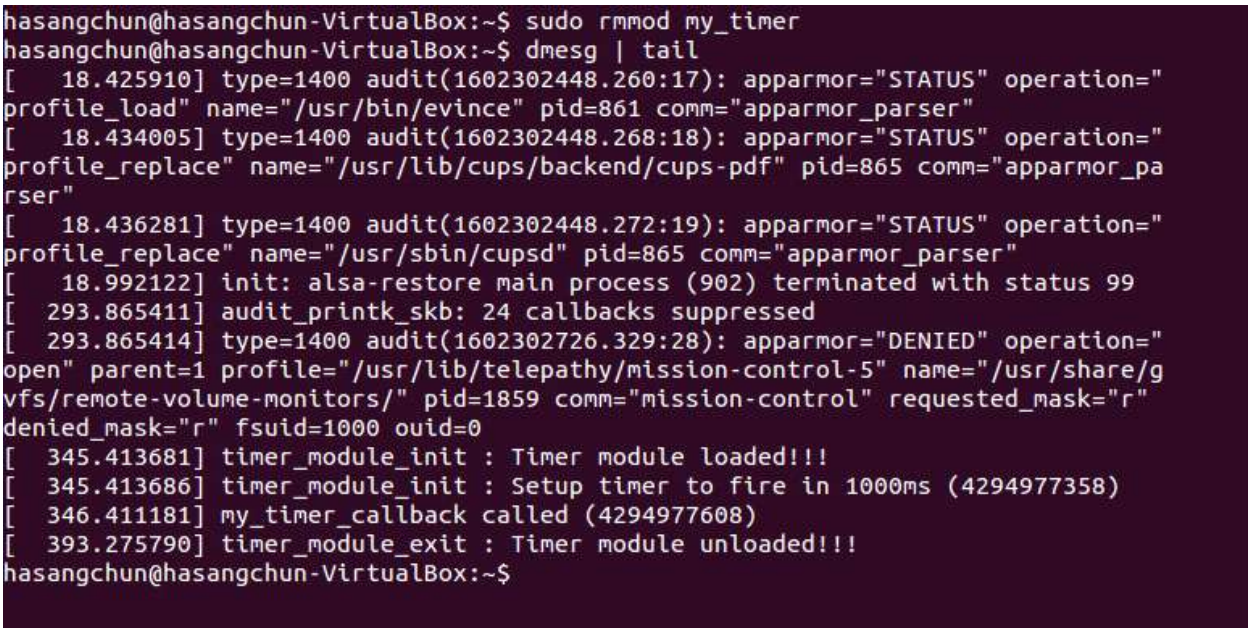
커널 모듈이 Load 될 때, module_init(timer_module_init) 함수가 호출되고
Unload 될 때, module_exit(timer_module_exit) 함수가 호출된다.

5. 실행결과 및 설명 ( 실행 결과를 캡쳐하여 첨부한 후 설명 )




실습 실행결과






과제 실행결과

(그림을 문서에 포함, 글자처럼 취급 옵션, 잉크 절약과 잘 보이게 하기위해 그림 반전)

6. 고찰 ( 과제를 진행하면서 배운점 이나, 시행 착오 내용, 기타 느낀점 )

이번 실습과 과제는 리눅스 커널 모듈 프로그래밍에 관한 것이었다. 커널 모듈 프로그래밍에 대해 하나도 알지 못했는데 실습내용을 진행하면서 조금씩 알게 되었다. 우선 지금까지 사용했던 printf() 함수의 출력 대상은 표준 출력 디바이스(Standard Output)인 모니터이지만, printk() 함수는 커널 모듈처럼 운영체제가 실행될 때 문자열을 출력시키는 함수이므로 출력 대상은 커널이 실행중인 메모리이다.

또 신기했던 것은 커널 모듈 프로그래밍은 main() 함수가 필요 없다는 점이였다. 그리고 커널 모듈이 Load 될 때는 module_init() 함수가 호출되고 Unload 될 때는 module_exit() 함수가 호출되었다. MakeFile을 만들고 로드와 언로드 명령어를 실행한 후 dmesg 명령어를 실행하여 실습결과를 확인할 수 있었다.

과제는 커널 모듈 프로그래밍에서의 타이머 모듈 구현이었다. 실습과 대체적으로 비슷했지만, linux/timer.h 파일에서 제공하는 타이머 함수를 이용하여 타이머 모듈을 구현하는 것이었다. 전체적인 구조는 timer_module_init에서 setup_timer 함수를 사용하여 타이머를 초기화하고, mod_timer함수를 호출하여 타이머를 시작한다. 타이머가 만료되면 콜백 함수가 호출되고, 마지막으로 모듈을 제거하면 del_timer 함수를 통해 타이머가 삭제되는 구조이다. 여기서 mod_timer 함수를 인터넷에 검색하여 찾아본 결과, mod_timer 함수 안에는 __mod_timer 함수가 있었다. __가 앞에 왜 붙어있는지 몰랐는데 공부해보니, 리눅스 커널 코딩 룰에 따르면 __가 붙은 함수는 커널 함수 내에서만 호출할 수 있고 다른 커널 서브 시스템에서는 호출하지 못하는 것이었다. 파이썬 코드 style guide인 PEP 8이 있는 것은 알고 있었지만, 커널 안에도 코딩 룰이 있는지는 처음 알았다.

7. 전체 소스코드 ( 글자크기 9에 줄간격을 120%로 유지하고 한 줄이 너무 길지 않게 작성 )

```
#include<linux/kernel.h>
#include<linux/module.h>
#include<linux/timer.h>
#include<linux/init.h>
static struct timer_list my_timer;
void my_timer_callback(struct timer_list *timer)
```

```
{
        printk(KERN_NOTICE "%s called (%ld)₩n", __func__, jiffies);
}
static int __init timer_module_init(void)
{
        int ret;
        printk(KERN_NOTICE "%s : Timer module loaded!!!₩n", __func__);

        setup_timer(&my_timer, my_timer_callback, 0);
        printk(KERN_NOTICE "%s : Setup timer to fire in 1000ms (%ld)₩n", __func__,
jiffies);
        ret = mod_timer(&my_timer, jiffies + msecs_to_jiffies(1000));

        if(ret) printk(KERN_NOTICE "Error in mod_timer₩n");
        return 0;
}
static void __exit timer_module_exit(void)
{
        int ret;
        ret = del_timer(&my_timer);
        if(ret) printk(KERN_NOTICE "The timer is still in use.₩n");
        printk(KERN_NOTICE "%s : Timer module unloaded!!!₩n"), __func__);
}
MODULE_LICENSE("GPL");
module_init(timer_module_init);
module_exit(timer_module_exit);


/*      Makefile      */


obj-m := my_timer.o

KERNELDIR := /lib/modules/$(shell uname -r)/build
PWD := $(shell pwd)

default:
$(MAKE) -C $(KERNELDIR) M=$(PWD) modules

clean:
$(MAKE) -C $(KERNELDIR) M=$(PWD) clean
```

(글자크기는 10으로 유지하고 줄간격도 160%를 유지할 것)