Project 2

Wrap Up

Provided Elevator Source Code

- elevator/
  - Makefile
  - include/
    - syscalls.h
  - src/
    - syscalls.c
    - module.c
    - sys_start_elevator.c
    - sys_issue_request.c
    - sys_stop_elevator.c

Everything not listed was dynamically created

Makefile

Name of your Module

```
MODULE_NAME = elevator
PWD := $(shell pwd)

#KDIR := /lib/modules/$(shell uname -r)/build
KDIR := /lib/modules/4.2.0/build
ccflags-y += -I$(src)/include

obj-y := src/sys_start_elevator.o
obj-y += src/sys_issue_request.o
obj-y += src/sys_stop_elevator.o

$(MODULE_NAME)-objs += src/syscalls.o
$(MODULE_NAME)-objs += src/module.o
obj-m := $(MODULE_NAME).o

default:
$(MAKE) -C $(KDIR) SUBDIRS=$(PWD) modules

clean:
$(MAKE) -C $(KDIR) SUBDIRS=$(PWD) clean
```

Makefile

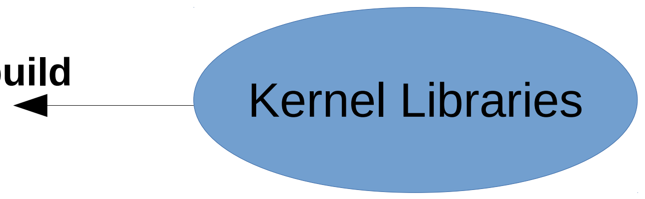
```
MODULE_NAME = elevator
PWD := $(shell pwd)
```

Point to Module code

```
#KDIR := /lib/modules/$(shell uname -r)/build
KDIR := /lib/modules/4.2.0/build
ccflags-y += -I$(src)/include

obj-y := src/sys_start_elevator.o
obj-y += src/sys_issue_request.o
obj-y += src/sys_stop_elevator.o

$(MODULE_NAME)-objs += src/syscalls.o
$(MODULE_NAME)-objs += src/module.o
obj-m := $(MODULE_NAME).o

default:
$(MAKE) -C $(KDIR) SUBDIRS=$(PWD) modules

clean:
$(MAKE) -C $(KDIR) SUBDIRS=$(PWD) clean
```

Makefile

```
MODULE_NAME = elevator
PWD := $(shell pwd)

#KDIR := /lib/modules/$(shell uname -r)/build
KDIR := /lib/modules/4.2.0/build
ccflags-y += -I$(src)/include

obj-y := src/sys_start_elevator.o
obj-y += src/sys_issue_request.o
obj-y += src/sys_stop_elevator.o

$(MODULE_NAME)-objs += src/syscalls.o
$(MODULE_NAME)-objs += src/module.o
obj-m := $(MODULE_NAME).o

default:
$(MAKE) -C $(KDIR) SUBDIRS=$(PWD) modules

clean:
$(MAKE) -C $(KDIR) SUBDIRS=$(PWD) clean
```



Makefile

```
MODULE_NAME = elevator
PWD := $(shell pwd)

#KDIR := /lib/modules/$(shell uname -r)/build
KDIR := /lib/modules/4.2.0/build
ccflags-y += -I$(src)/include

obj-y := src/sys_start_elevator.o
obj-y += src/sys_issue_request.o
obj-y += src/sys_stop_elevator.o

$(MODULE_NAME)-objs += src/syscalls.o
$(MODULE_NAME)-objs += src/module.o
obj-m := $(MODULE_NAME).o

default:
$(MAKE) -C $(KDIR) SUBDIRS=$(PWD) modules

clean:
$(MAKE) -C $(KDIR) SUBDIRS=$(PWD) clean
```

Compile Flags

Header Files

Makefile

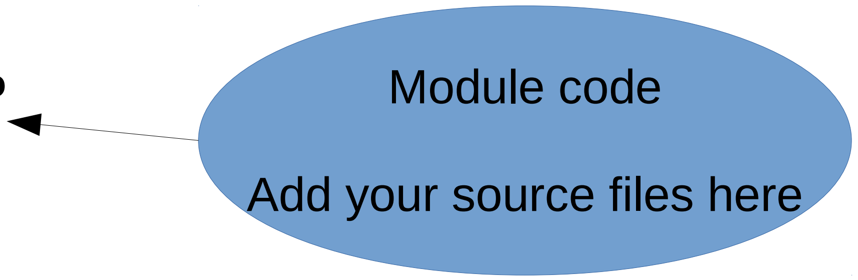
```
MODULE_NAME = elevator
PWD := $(shell pwd)

#KDIR := /lib/modules/$(shell uname -r)/build
KDIR := /lib/modules/4.2.0/build
ccflags-y += -I$(src)/include

obj-y := src/sys_start_elevator.o
obj-y += src/sys_issue_request.o
obj-y += src/sys_stop_elevator.o

$(MODULE_NAME)-objs += src/syscalls.o
$(MODULE_NAME)-objs += src/module.o
obj-m := $(MODULE_NAME).o

default:
$(MAKE) -C $(KDIR) SUBDIRS=$(PWD) modules

clean:
$(MAKE) -C $(KDIR) SUBDIRS=$(PWD) clean
```

Built-in code

Do not modifiy!

Makefile

```
MODULE_NAME = elevator
PWD := $(shell pwd)

#KDIR := /lib/modules/$(shell uname -r)/build
KDIR := /lib/modules/4.2.0/build
ccflags-y += -I$(src)/include

obj-y := src/sys_start_elevator.o
obj-y += src/sys_issue_request.o
obj-y += src/sys_stop_elevator.o

$(MODULE_NAME)-objs += src/syscalls.o
$(MODULE_NAME)-objs += src/module.o
obj-m := $(MODULE_NAME).o

default:
$(MAKE) -C $(KDIR) SUBDIRS=$(PWD) modules

clean:
$(MAKE) -C $(KDIR) SUBDIRS=$(PWD) clean
```



Makefile

```
MODULE_NAME = elevator
PWD := $(shell pwd)

#KDIR := /lib/modules/$(shell uname -r)/build
KDIR := /lib/modules/4.2.0/build
ccflags-y += -I$(src)/include

obj-y := src/sys_start_elevator.o
obj-y += src/sys_issue_request.o
obj-y += src/sys_stop_elevator.o

$(MODULE_NAME)-objs += src/syscalls.o
$(MODULE_NAME)-objs += src/module.o
obj-m := $(MODULE_NAME).o

default:
$(MAKE) -C $(KDIR) SUBDIRS=$(PWD) modules

clean:
$(MAKE) -C $(KDIR) SUBDIRS=$(PWD) clean
```

Module object

Makefile

```
MODULE_NAME = elevator
PWD := $(shell pwd)

#KDIR := /lib/modules/$(shell uname -r)/build
KDIR := /lib/modules/4.2.0/build
ccflags-y += -I$(src)/include

obj-y := src/sys_start_elevator.o
obj-y += src/sys_issue_request.o
obj-y += src/sys_stop_elevator.o

$(MODULE_NAME)-objs += src/syscalls.o
$(MODULE_NAME)-objs += src/module.o
obj-m := $(MODULE_NAME).o

default:
$(MAKE) -C $(KDIR) SUBDIRS=$(PWD) modules

clean:
$(MAKE) -C $(KDIR) SUBDIRS=$(PWD) clean
```

Compile

Makefile

```
MODULE_NAME = elevator
PWD := $(shell pwd)

#KDIR := /lib/modules/$(shell uname -r)/build
KDIR := /lib/modules/4.2.0/build
ccflags-y += -I$(src)/include

obj-y := src/sys_start_elevator.o
obj-y += src/sys_issue_request.o
obj-y += src/sys_stop_elevator.o

$(MODULE_NAME)-objs += src/syscalls.o
$(MODULE_NAME)-objs += src/module.o
obj-m := $(MODULE_NAME).o

default:
$(MAKE) -C $(KDIR) SUBDIRS=$(PWD) modules

clean:
$(MAKE) -C $(KDIR) SUBDIRS=$(PWD) clean
```

Clean

syscalls.h

```
#ifndef __ELEVATOR_SYSCALLS_H
#define __ELEVATOR_SYSCALLS_H

void elevator_syscalls_create(void);
void elevator_syscalls_remove(void);

#endif /*__ELEVATOR_SYSCALLS_H*/
```

References for module to call

Passes creation/removal to syscalls.c

syscalls.c

```
#include <syscalls.h>
#include <linux/printk.h>

extern long (*STUB_start_elevator)(void);
    long start_elevator(void) {
    printk("Starting elevator\n");
    return 0;
}
extern long (*STUB_issue_request)(int,int,int);
    long issue_request(int passenger_type, int start_floor, int destination_floor) {
    printk("New request: %d, %d => %d\n", passenger_type, start_floor, destination_floor);
    return 0;
}
extern long (*STUB_stop_elevator)(void);
    long stop_elevator(void) {
    printk("Stopping elevator\n");
    return 0;
}

void elevator_syscalls_create(void) {
    STUB_start_elevator =& (start_elevator);
    STUB_stop_elevator =& (stop_elevator);
}
void elevator_syscalls_remove(void) {
    STUB_start_elevator = NULL;
    STUB_stop_elevator = NULL;
}
```

Stubs from built in files

Add implementation here

syscalls.c

```
#include <syscalls.h>
#include <linux/printk.h>

extern long (*STUB_start_elevator)(void);
   long start_elevator(void) {
   printk("Starting elevator\n");
   return 0;
}
extern long (*STUB_issue_request)(int,int,int);
   long issue_request(int passenger_type, int start_floor, int destination_floor) {
   printk("New request: %d, %d => %d\n", passenger_type, start_floor, destination_floor);
   return 0;
}
extern long (*STUB_stop_elevator)(void);
   long stop_elevator(void) {
   printk("Stopping elevator\n");
   return 0;
}

void elevator_syscalls_create(void) {
   STUB_start_elevator =& (start_elevator);
   STUB_stop_elevator =& (stop_elevator);
}
void elevator_syscalls_remove(void) {
   STUB_start_elevator = NULL;
   STUB_stop_elevator = NULL;
}
```

Handles setting up the stubs under module insertion/removal

module.c

```
#include <linux/init.h>
#include <linux/module.h>
#include <syscalls.h>

MODULE_LICENSE("GPL");
MODULE_AUTHOR("Britton");
MODULE_DESCRIPTION("Simple module designed to illustrate scheduling");

static int hello_init(void) {
    printk("Inserting Elevator\n");
    elevator_syscalls_create();
    return 0;
}
static void hello_exit(void) {
    printk("Removing elevator\n");
    elevator_syscalls_remove();
}
module_init(hello_init);
module_exit(hello_exit);
```

Inserts system calls

module.c

```
#include <linux/init.h>
#include <linux/module.h>
#include <syscalls.h>

MODULE_LICENSE("GPL");
MODULE_AUTHOR("Britton");
MODULE_DESCRIPTION("Simple module designed to illustrate scheduling");

static int hello_init(void) {
    printk("Inserting Elevator\n");
    elevator_syscalls_create();
    return 0;
}
static void hello_exit(void) {
    printk("Removing elevator\n");
    elevator_syscalls_remove();
}
module_init(hello_init);
module_exit(hello_exit);
```

Removes system calls

sys_start_elevator.c

```
#include <linux/linkage.h>
#include <linux/kernel.h>
#include <linux/module.h>

long (*STUB_start_elevator)(void) = NULL;
EXPORT_SYMBOL(STUB_start_elevator);
asmlinkage long sys_start_elevator(void) {
    if (STUB_start_elevator)
        return STUB_start_elevator();
    else
        return -ENOSYS;
}
```

Installs system call stub into kernel

Don't change unless you
wish to reinstall kernel

sys_issue_request.c

```
#include <linux/linkage.h>
#include <linux/kernel.h>
#include <linux/module.h>

long (*STUB_issue_request)(int,int,int) = NULL;
EXPORT_SYMBOL(STUB_issue_request);
asmlinkage long sys_issue_request(int passenger_type, int
start_floor, int destination_floor) {
    if (STUB_issue_request)
        return STUB_issue_request(passenger_type, start_floor, destination_floor);
    else
        return -ENOSYS;
}
```

Installs system call stub into kernel

Don't change unless you wish to reinstall kernel

sys_stop_elevator.c

```
#include <linux/linkage.h>
#include <linux/kernel.h>
#include <linux/module.h>

long (*STUB_stop_elevator)(void) = NULL;
EXPORT_SYMBOL(STUB_stop_elevator);
asmlinkage long sys_stop_elevator(void) {
    if (STUB_stop_elevator)
        return STUB_stop_elevator();
    else
        return -ENOSYS;
}
```

Installs system call stub into kernel

Don't change unless you wish to reinstall kernel

Submission

- You'll need to submit your code to black board by the due date (November 2nd)
  - e.g. p2_dennis_rahman.tar
  - Only contains your documentation and the code you wrote, not any of the additional files you had to modify
  - This is to checkpoint your progress, I'll require you to use this code in the demo
  - This is also how I'll grade your documentation
- You'll need to sign up for a demonstration date

Demonstration

- You will have 20 minutes
- I'll tell you to download my drivers, issue commands, and run your code / my drivers
- You will be tested on
  - Code correctness
  - Answering questions
    - Design
    - Implementation

Doodle

- Here is a link to sign up for the demo
  - http://doodle.com/poll/t3qm5v9m7zuis2zp
- Your time may be canceled at the last minute due to not knowing when the lab is in use
  - I've been slowly narrowing down the meeting times for the other classes to prevent this from happening
  - We can reschedule for another time the following week
- When signing up, use your machine number as well as the first name of each member of your group
  - It makes it easier for me to track
  - Helps me detect typos and other problems

Drivers

- I'll provide some drivers for you to test your code today
  - consumer.c
  - producer.c
- The drivers for the demo will be different
  - Designed around the specification
  - Designed to stress test your implementation
  - Designed to time your scheduler

consumer.c

- Starts elevator
- Stops elevator

producer.c

- Creates a person
  - Random type
  - Random starting floor
  - Random destination floor
- Chance of being an invalid entry