CS 452 Review 4

Scheduling Algorithms

First Come First Served

Shortest Time First

Round Robin

Priority Scheduling

Process	Arrival Time	Execute Time
A	0	3
B	0	6
C	0	4
D	0	2

Assignment 2

Implement a Dice Device Driver

The driver will support the read() system call

Reading a byte will return a number 1-5

read(dice_fd, buffer, 1) // This will read 1 byte with a value 1-5 into the buffer
read(dice_fd, buffer, 5) // This will read 5 bytes with values 1-5 into the buffer
read(dice_fd, buffer, 0) // This will do nothing

Printk

Printk is a linux function, not a C function, so you can use it for debugging.

printk(log_level, “string to print”)

printk(KERN_ALERT, “you can use string interpolation with printk\n”)

printk(KERN_ALERT, “Hello w%drld\n”, 0)

Initializing a Device Driver

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

static int __init
your_driver_init(void){
        return 0;
}

module_init(your_driver_init);
```

Uninitializing a Device Driver

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

static int __exit
your_driver_exit(void){
        return 0;
}

module_init(your_driver_exit);
```

Enabling system calls for your driver

```
static const struct file_operations your_driver_fops = {
        .owner    = THIS_MODULE,
        .read     = your_drivers_read_function;
}

static struct miscdevice your_dev_config = {
        MISC_DYNAMIC_MINOR,       //Dynamically assign driver number
        “Your Device Driver’s Name”    //name your device creating /dev/dev_name
        &your_driver_fops              //Struct above
}
```

Registering Device operations

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

static int __init
your_driver_init(void){
        return misc_register(&your_dev_config);
}

module_init(your_driver_init);
```

Deregister Device

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

static int __exit
your_driver_exit(void){
        return misc_deregister(&your_dev_config);
}

module_init(your_driver_exit);
```

Implementing read system call

```
static size_t your_drivers_read_function
(struct file* fd, char* buf, size_t amount_to_read, ioff_t* offset_into_file) {
        int num_of_bytes_read = 0;
        return num_of_bytes_read;
}

static const struct file_operations your_driver_fops = {
        .owner    = THIS_MODULE,
        .read     = your_drivers_read_function;
}
```