Lab 1 Report

Name: Ricky Fan
Email: fanh11@mcmaster.ca
Student ID: 400248976
Date: 02/10/19

jiffies.ko

This kernel module creates a /proc file named /proc/jiffies that reports the current value of *jiffies* when the /proc/jiffies file is read, such as with command:

```
cat /proc/jiffies
```

File /proc/jiffies is removed when the moduleis removed

```
/**
 * jiffies.c
 *
 * Kernel module that prints the current jiffies value
 * when cat /proc/jiffies is called
 *
 * fanh11@mcmaster.ca
 * Based on John Wiley & Sons - 2018. hello.c
 */

// For kernel programming
#include <linux/init.h>
#include <linux/module.h>
#include <linux/kernel.h>
#include <asm/uaccess.h>
// For proc read
#include <linux/proc_fs.h>
// For jiffies
#include <linux/jiffies.h>

// Define a constant buffer size for
// the string to be printed
```

```
// when cat /proc/jiffies is called
#define BUFFER_SIZE 128
// Defines the name of the proc
#define PROC_NAME "jiffies"

/**
 * Function prototypes
 */
ssize_t proc_read(struct file *file, char *buf, size_t
    count, loff_t *pos);

static struct file_operations proc_ops = {
    .owner = THIS_MODULE,
    .read = proc_read,
};

/* This function is called when the module is loaded
   */
int proc_init(void)
{
    // creates the /proc/jiffies entry
    // the following function call is a wrapper for
    // proc_create_data() passing NULL as the last
       argument
    proc_create(PROC_NAME, 0, NULL, &proc_ops);

    // logs kernel informational message "/proc/
       jiffies created"
    // to show /proc/jiffies entry is created
    printk(KERN_INFO "/proc/%s created\n", PROC_NAME);
    return 0;
}

/* This function is called when the module is removed
   */
void proc_exit(void)
{
    // removes the /proc/jiffies entry
    remove_proc_entry(PROC_NAME, NULL);
```

```
    // logs kernel informational message "/proc/
       jiffies removed"
    // to show /proc/jiffies entry is removed
    printk(KERN_INFO "/proc/%s removed\n", PROC_NAME);
}

/**
 * This function is called each time the /proc/jiffies
     is read
 */
ssize_t proc_read(struct file *file, char __user *
   usr_buf, size_t count, loff_t *pos)
{
    // use rv to store the number of characters
       printed
    int rv = 0;
    char buffer[BUFFER_SIZE];
    // use complete as boolean to prevent repeat reads
    static int complete = 0;

    // return 0 to prevent repeated reads if completed
    if (complete)
    {
        complete = 0;
        return 0;
    }

    // set to 1 after the proc is read
    complete = 1;

    // store the number of characters printed in rv
    // sprintf returns the number of characters
    // in buffer (excluding null padding characters)
    rv = sprintf(buffer, "jiffies: %lu\n", jiffies);

    // copies the contents of buffer to userspace
       usr_buf
    copy_to_user(usr_buf, buffer, rv);

    // return printed string length
```

```
    return rv;
}

/* Set the module entry and exit points */
module_init(proc_init);
module_exit(proc_exit);

/* Maintaining original license */
MODULE_LICENSE("GPL");
MODULE_DESCRIPTION("Jiffies Module");
MODULE_AUTHOR("RF");
```

seconds.ko

This kernel module creates a proc file named /proc/seconds taht reports the number of elapsed seconds since the kernel module was loaded. It uses the value *jiffies* and *HZ* rate. When a user enter command:

cat /proc/seconds

The kernel module will report the number oslf seconds the kernel module has been loaded. The file /proc/seconds is removed when the module is removed

```
/**
 * seconds.c
 *
 * Kernel module that prints the number of seconds
 * elapsed since the kernel module was loaded
 * when cat /proc/seconds is called
 *
 * fanh11@mcmaster.ca
 * Based on John Wiley & Sons - 2018. hello.c
 */

// For kernel programming
#include <linux/init.h>
#include <linux/module.h>
#include <linux/kernel.h>
#include <asm/uaccess.h>
```

```
// For proc read
#include <linux/proc_fs.h>
// For jiffies - the number of timer interrupt
#include <linux/jiffies.h>
// For HZ - the frequency of timer interrupt
#include <asm/param.h>

// Define a constant buffer size for
// the string to be printed
// when cat /proc/seconds is called
#define BUFFER_SIZE 128
// Defines the name of the proc
#define PROC_NAME "seconds"

// Define a static variable to store
// the value of jiffies when the
// kernel module first loaded
static unsigned long start_jiffies;

/**
 * Function prototypes
 */
ssize_t proc_read(struct file *file, char *buf, size_t
    count, loff_t *pos);

static struct file_operations proc_ops = {
    .owner = THIS_MODULE,
    .read = proc_read,
};

/* This function is called when the module is loaded
   */
int proc_init(void)
{
    // store the initial jiffies value at the start
    // of the kernel module
    start_jiffies = jiffies;

    // creates the /proc/seconds entry
    // the following function call is a wrapper for
```

```
    // proc_create_data() passing NULL as the last
       argument
    proc_create(PROC_NAME, 0, NULL, &proc_ops);

    // logs kernel informational message "/proc/
       seconds created"
    // to show /proc/seconds entry is created
    printk(KERN_INFO "/proc/%s created\n", PROC_NAME);
    return 0;
}

/* This function is called when the module is removed
   */
void proc_exit(void)
{
    // removes the /proc/seconds entry
    remove_proc_entry(PROC_NAME, NULL);

    // logs kernel informational message "/proc/
       seconds removed"
    // to show /proc/seconds entry is removed
    printk(KERN_INFO "/proc/%s removed\n", PROC_NAME);
}

/**
 * This function is called each time the /proc/seconds
     is read
 */
ssize_t proc_read(struct file *file, char __user *
   usr_buf, size_t count, loff_t *pos)
{
    // use rv to store the number of characters
       printed
    int rv = 0;
    char buffer[BUFFER_SIZE];
    // use secs to store the number of seconds elapsed
    // since the kernel was loaded
    unsigned long secs;
    // use complete as boolean to prevent repeat reads
    static int complete = 0;
```

```
    // return 0 to prevent repeated reads if completed
    if (complete)
    {
        complete = 0;
        return 0;
    }

    // set to 1 after the proc is read
    complete = 1;

    // use current jiffies value minus the initial
    // jiffies value then divide by HZ to calculate
    // the number of seconds elasped since the
    // kernel was loaded
    secs = (jiffies - start_jiffies) / HZ;

    // store the number of characters printed in rv
    // sprintf returns the number of characters
    // in buffer (excluding null padding characters)
    rv = sprintf(buffer, "seconds elapsed: %lu\n",
       secs);

    // copies the contents of buffer to userspace
       usr_buf
    copy_to_user(usr_buf, buffer, rv);

    // return printed string length
    return rv;
}

/* Set the module entry and exit points */
module_init(proc_init);
module_exit(proc_exit);

/* Maintaining original license */
MODULE_LICENSE("GPL");
MODULE_DESCRIPTION("Seconds Module");
MODULE_AUTHOR("RF");
```