Operating System
Assignment 7 – Procfs

1. **Task given:**
   a. Create a node in /proc file system
   b. To read/write from and into the created node.

2. **Code Description:**

   The kernel module code is explained below:

   a. Program header: It contains the code that adds references to the Linux kernel header files that are required for all the kernel-level functions and structures that are utilized in the program. This section also contains certain variables that are used throughout the program, such as the MAX_buff_SIZE and NAME.

```
#include <linux/module.h>
#include <linux/kernel.h>
#include <linux/init.h>
#include <linux/proc_fs.h>
#include <linux/uaccess.h>
#define MAX_buff_SIZE 1024
#define NAME "buffer"
```

   b. Objects used: Defines three globally used data objects which contains the file entry in /proc, the character buffer used by the proc file and the size of the text that is inserted into the proc file.

```
/* This structure hold information about the /proc file */
static struct proc_dir_entry *PROCFILE;
/* The buff used to store character for this module  */
static char procfs_buff[MAX_buff_SIZE];
/*The size of the buff */
static unsigned long procfs_buff_size = 0;
```

c. Module initialization and termination: These functions handle the creation of the proc file and the removal of the proc file

```
/* This function is executed when the module is inserted*/
static int __init Lab7_init(void)
{
    /* create the /proc file */
    proc_file = proc_create(NAME, 0644, NULL, &fops_struct);
    if (proc_file == NULL) {
        remove_proc_entry(NAME, NULL);
        printk(KERN_ALERT "Error: Could not initialize /proc/%s\n",
            NAME);
        return -ENOMEM;
    }

    printk(KERN_INFO "/proc/%s created\n", NAME);
    return 0;
}

/* This function is called when the module is unloaded */

static void __exit Lab7_exit(void)
{
    remove_proc_entry(NAME, NULL);
    printk(KERN_INFO "/proc/%s removed\n", NAME);
}

module_init(Lab7_init);
module_exit(Lab7_exit);
```

d. The file_operations object: It is about the file_operations structure that is used, which refers the operations detailed in the next sections to its object.

```
static struct file_operations fops_struct = {
    .read = proc_file_read,
    .write = proc_file_write,
};
```

e. The procfile_read function: This section explains the operation that takes place when the text is to be read from the proc file. After this, the size of the buffer is sent back as the return value.

```
/* This function is called then the /proc file is read */
static ssize_t proc_file_read(struct file *file, char *buff, size_t buff_length, loff_t *offset)
{
    static int flag = 0;
    if(flag) {
    printk(KERN_INFO "read : END\n");
        flag = 0;
        return 0;
    }
    flag =1;
    printk(KERN_INFO "reading from (/proc/%s) :\n",NAME);
    return  sprintf(buff, procfs_buff);
}
```

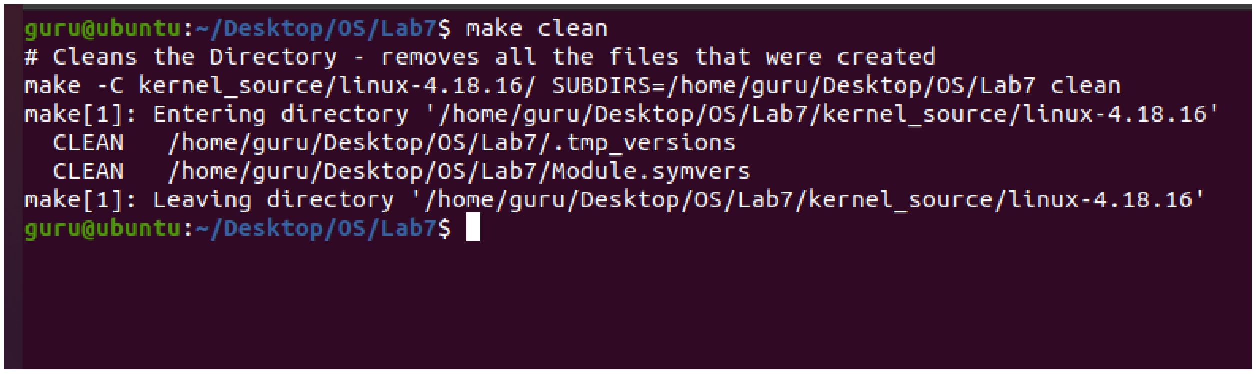
f. The procfile_write function: In this operation takes place when some text is written to the proc file. Whenever some text is written to the proc file then the procfile_write function is called. When this happens, the text length is first checked to see if it exceeds the size of the buffer. If yes, then the text is truncated to the maximum length supported by limiting the text length to the maximum length. Then the text is copied to the character buffer using the copy_from_user function, using the length as the limiter. If that runs without any error, it will return the buffer size, else it will return the error code corresponding to -EFAULT

```
/* This function is called with the /proc file is written */
static ssize_t proc_file_write(struct file *file,const char *buff, size_t count, loff_t *offset)
{
    /* get buff size */
    procfs_buff_size = count;
    if (procfs_buff_size > MAX_buff_SIZE ) {
            procfs_buff_size = MAX_buff_SIZE;
        }
        /* write data to the buff */
        if ( copy_from_user(procfs_buff, buff, procfs_buff_size) ) {
            return -EFAULT;
        }
        return procfs_buff_size;
    }
```

3. Result:

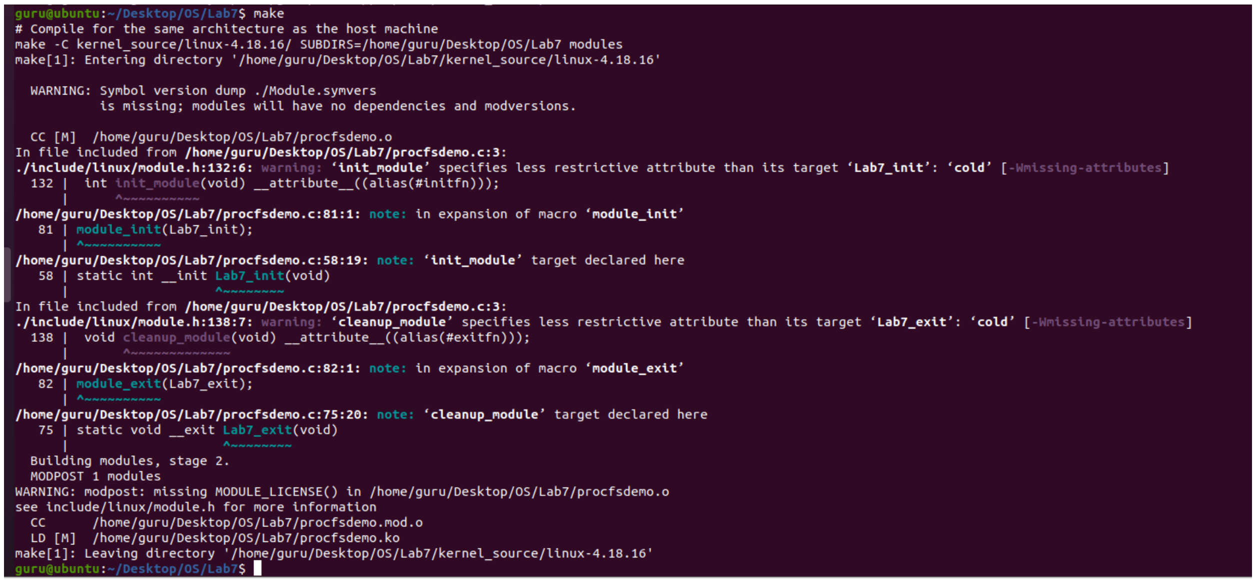
Procedure to build and run the module.

a. Create a clean build environment using ‘make clean’ command



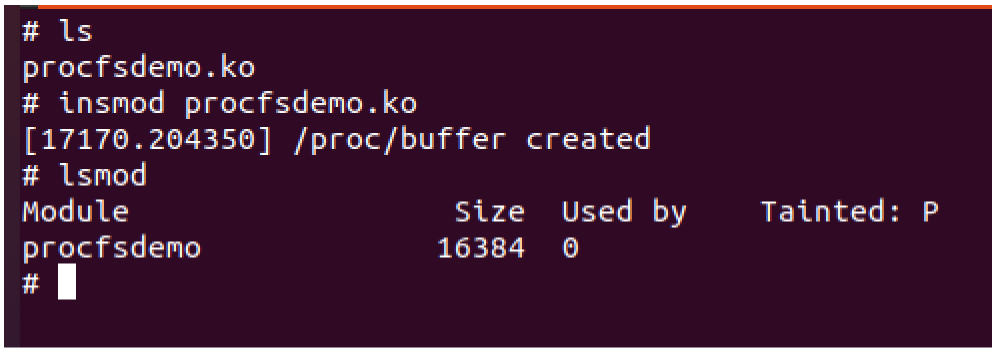
```
guru@ubuntu:~/Desktop/OS/Lab7$ make clean
# Cleans the Directory - removes all the files that were created
make -C kernel_source/linux-4.18.16/ SUBDIRS=/home/guru/Desktop/OS/Lab7 clean
make[1]: Entering directory '/home/guru/Desktop/OS/Lab7/kernel_source/linux-4.18.16'
  CLEAN   /home/guru/Desktop/OS/Lab7/.tmp_versions
  CLEAN   /home/guru/Desktop/OS/Lab7/Module.symvers
make[1]: Leaving directory '/home/guru/Desktop/OS/Lab7/kernel_source/linux-4.18.16'
guru@ubuntu:~/Desktop/OS/Lab7$
```

b. Build the kernel module using ‘make’ command



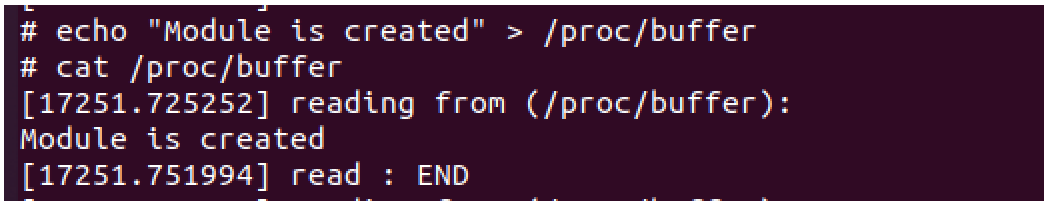
```
guru@ubuntu:~/Desktop/OS/Lab7$ make
# Compile for the same architecture as the host machine
make -C kernel_source/linux-4.18.16/ SUBDIRS=/home/guru/Desktop/OS/Lab7 modules
make[1]: Entering directory '/home/guru/Desktop/OS/Lab7/kernel_source/linux-4.18.16'

  WARNING: Symbol version dump ./Module.symvers
           is missing; modules will have no dependencies and modversions.

  CC [M]  /home/guru/Desktop/OS/Lab7/procfsdemo.o
In file included from /home/guru/Desktop/OS/Lab7/procfsdemo.c:3:
./include/linux/module.h:132:6: warning: ‘init_module’ specifies less restrictive attribute than its target ‘Lab7_init’: ‘cold’ [-Wmissing-attributes]
  132 |  int init_module(void) __attribute__((alias(#initfn)));
      |      ^~~~~~~~~~~
/home/guru/Desktop/OS/Lab7/procfsdemo.c:81:1: note: in expansion of macro ‘module_init’
   81 | module_init(Lab7_init);
      | ^~~~~~~~~~~
/home/guru/Desktop/OS/Lab7/procfsdemo.c:58:19: note: ‘init_module’ target declared here
   58 | static int __init Lab7_init(void)
      |                   ^~~~~~~~~
In file included from /home/guru/Desktop/OS/Lab7/procfsdemo.c:3:
./include/linux/module.h:138:7: warning: ‘cleanup_module’ specifies less restrictive attribute than its target ‘Lab7_exit’: ‘cold’ [-Wmissing-attributes]
  138 |  void cleanup_module(void) __attribute__((alias(#exitfn)));
      |       ^~~~~~~~~~~~~~
/home/guru/Desktop/OS/Lab7/procfsdemo.c:82:1: note: in expansion of macro ‘module_exit’
   82 | module_exit(Lab7_exit);
      | ^~~~~~~~~~~
/home/guru/Desktop/OS/Lab7/procfsdemo.c:75:20: note: ‘cleanup_module’ target declared here
   75 | static void __exit Lab7_exit(void)
      |                    ^~~~~~~~~
  Building modules, stage 2.
  MODPOST 1 modules
WARNING: modpost: missing MODULE_LICENSE() in /home/guru/Desktop/OS/Lab7/procfsdemo.o
see include/linux/module.h for more information
  CC      /home/guru/Desktop/OS/Lab7/procfsdemo.mod.o
  LD [M]  /home/guru/Desktop/OS/Lab7/procfsdemo.ko
make[1]: Leaving directory '/home/guru/Desktop/OS/Lab7/kernel_source/linux-4.18.16'
guru@ubuntu:~/Desktop/OS/Lab7$
```

c. Insert the module using ‘insmod procfsdemo.ko’



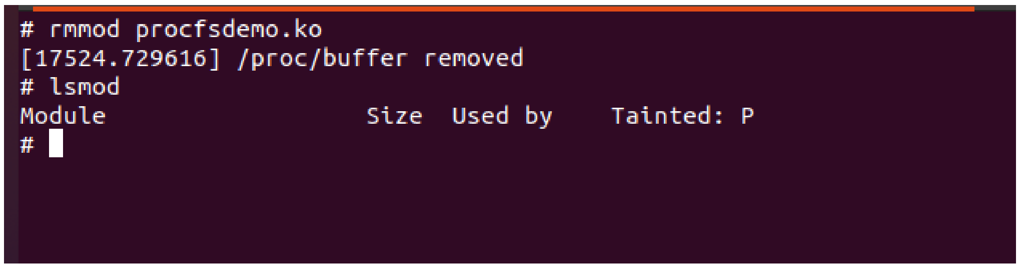
```
# ls
procfsdemo.ko
# insmod procfsdemo.ko
[17170.204350] /proc/buffer created
# lsmod
Module                  Size  Used by    Tainted: P
procfsdemo             16384  0
# 
```

d. Write and reading from proc file using command echo “Module is created” > /proc/buffer and cat /proc/buffer:



```
# echo "Module is created" > /proc/buffer
# cat /proc/buffer
[17251.725252] reading from (/proc/buffer):
Module is created
[17251.751994] read : END
```

e. Removing the module from kernel using command ‘rmmod procfsdemo.ko’



```
# rmmod procfsdemo.ko
[17524.729616] /proc/buffer removed
# lsmod
Module                  Size  Used by    Tainted: P
# 
```