Operating Systems (SFWRENG 3SH3), Winter 2026

Prof. Neerja Mhaskar

Extra material for Assignment 1

/proc File System

For Kernel Versions <= 6.2.x (as per the textbook edition) please refer to the `oldKernel` folder, otherwise `for newer Linux distributions please` refer to the `newKernel` folder provided. To find the kernel version please run the following command on the terminal ``uname -r``.

Note: `hello_newKernel.c` and `hello_oldKernel.c` is simply referred to as `hello.c` from here on to avoid duplication of material.

The `/proc` file system is a "pseudo" file system that exists only in kernel memory and is used primarily for querying various kernel and per-process statistics. This exercise involves designing kernel modules that create additional entries in the `/proc` file system involving both kernel statistics and information related to specific processes.

We begin by describing how to create a new entry in the `/proc` file system. The program example named `hello.c` (included with this PDF) creates a `/proc` entry named `/proc/hello`. If a user enters the command

```
cat /proc/hello
```

the `Hello World` message is returned.

Older Kernels (using struct file_operations):

In older kernels, we create a new `/proc/hello` entry using the `proc_create()` function in the module entry point `proc_init()`. The `proc_create()` function is passed a reference to a `struct file_operations`, which defines the operations that can be performed on the /proc/hello file. This structure initializes members such as `.owner` and `.read`. The `.owner` member is set to `THIS_MODULE`, which tracks the module that owns the /proc entry, while the .read member points to the `proc_read()` function. This function will be invoked whenever the `/proc/hello` file is read, ensuring that the module provides the correct behavior when userspace accesses the /proc entry.

Newer Kernels (using struct proc_ops):

In newer kernels, the `proc_create()` function uses a `struct proc_ops` instead of `struct file_operations` for handling `/proc` entries. The `proc_ops` structure replaces the older `file_operations` for `/proc` file system operations, providing a more focused set of operations specific to `/proc` file handling. The `.proc_read` member in `proc_ops` replaces `.read`, and it is similarly assigned to the `proc_read()` function. This change simplifies handling /proc files while keeping other file operations separate. As in older kernels, `proc_create()` still associates the module with the `/proc/hello` entry and ensures the `proc_read()` function is called when userspace reads from the file.

Examining this `proc_read()` function, we see that the string “`Hello World\n`” is written to the variable buffer where buffer exists in kernel memory. Since `/proc/hello` can be accessed from user space, we must copy the contents of buffer to user space using the kernel function `copy_to_user()`. This function copies the contents of kernel memory buffer to the variable `usr_buf`, which exists in user space.

Each time the `/proc/hello` file is read, the `proc_read()` function is called repeatedly until it returns `0`, so there must be logic to ensure that this function returns `0` once it has collected the data (in this case, the string “`Hello World\n`”) that is to go into the corresponding `/proc/hello` file.

Finally, notice that the `/proc/hello` file is removed in the module exit point `proc_exit()` using the function `remove_proc_entry()`.

Please use the following commands to try out the above hello kernel module.

```
1. make
2. sudo insmod hello.ko (hello_oldKernel.ko or hello_newKernel.ko)
3. cat /proc/hello
4. sudo rmmod hello (hello_oldKernel or hello_newKernel)
```