Want to play a game? – Linux Kernel Modules

ID1206 HT17

User space vs. Kernel space

Distinction of “rings” within the processor

User space programs run in one ring – kernel space in another

Allows for execution of privileged or sensitive instructions

Helps you not to crash your computer every other hour

In this assignment you will write a kernel space program

Either by compiling a custom kernel or a kernel module

Kernel Module

Macros

int main() vs entry and exit functions

User-defined and specified by macros ‘module_init’ and ‘module_exit’

MODULE_ macros provides info about the module. License, author etc.

printk()

Kernel modules doesn’t have a regular terminal thus nowhere to print to

If you want to print stuff use the kernel version of printf()

Printouts show up in dmesg

Communicating with your kernel module

Implementing system calls

UNIX's fetish of the file interface

- Just make a special file. Reading and writing to and from your module
- The /proc interface – The file structure representing your kernel
  - A different type of file – not saved ANYWHERE!?
- Block files, character files and special files – anyone know the difference?
    - /dev, /sys, /proc

file_operations()

Each field of the structure corresponds to the address of some function defined by the driver to handle a requested operation.

For example, every character driver needs to define a function that reads from the device. The file_operations structure holds the address of the module's function that performs that operation.

In this assignment we implement a module providing a file in /proc with the seq_file interface, and a device driver in /dev.

Writing to this file is done with the seq_printf() function

The ioctl system call

Sending requests to a file

Used to manipulate underlying device parameters of special files

The kernel module implements a handler function and acts according to the requests made

We implement a function that copies a string to a buffer using ‘copy_to_user’

Happy Hacking!