Character Device Driver
Loadable Kernel Module

Guide and Documentation

By Jacob Werner

Contents

Summary

When creating a device driver in Linux-based systems, device I/O is modeled with file systems. Using this method, reading and writing to a driver file will invoke the associated device driver to do the actual reading and writing.

This simple character device driver is a Loadable Kernel Module that writes characters inputted to a device file and then can read those characters. This driver was originally developed for use with Ubuntu 14.04 and was developed using the C language. This package comes with a test script to ensure functionality of the character driver and to demonstrate its capabilities.

Technical Features

The following features are included in this character driver to ensure proper and streamlined functionality:

- Due to the nature of the LKM structure, users can install and uninstall this character driver without restarting their system.
- init() and exit() functions to register/unregister the driver with the system registry and memory management.
- A dynamically allocated kernel buffer with a fixed size of 1024 bytes.
- Read and write functions to manipulate and read data from the device file.
- A seek function that determines the offset of an inputted value in the device file.
- Open and close functions that logs the amount of times the device file has been opened or closed.
- A basic command line test script to test all functions within the character driver.

Technical Limitations

The following are documented and expected limitations when working with character drivers:

- Character drivers operate in blocking mode, so users must wait until any write operation being done by a character driver completes.
- Due to the fixed size of the kernel buffer in this driver, inputs larger than 1024 bytes will not be accepted.

Specifications and Technical Minimums

The following are the minimum recommended specifications for running this character driver at the time of install. Any configurations below or outside the purview of these specified parameters may result in unpredictable or unresponsive behavior.

- Ubuntu 14.04
- Major driver identifier 240 is unallocated

File and Function Descriptions

Makefile - Simple makefile to build code package.
SCDTest.c - Source code for the test script
simple_character_device.c - File contains open, close, read, write, seek, init, and exit operations for driver operation

- **`int simple_character_device_init(void)`**
    - Creates kernel buffer and registers device with major 240. Will print to the kernel upon success.

- **`void simple_character_device_exit(void)`**
    - Frees kernel buffer and unregisters the device.

- **`int my_open(struct inode*, struct file*)`**
    - 2 parameters:
        - A inode struct pointer to represent the device file.
        - A file struct pointer for the state of the file.
    - Logs the number of times the device file has been opened and prints that number to the kernel.
    - Returns an integer to represent success/failure.

- **`int my_close(struct inode*, struct file*)`**
    - 2 parameters:
        - A inode struct pointer to represent the device file.
        - A file struct pointer for the state of the file.
    - Logs the number of times the device file has been closed and prints that number to the kernel.
    - Returns an integer to represent success/failure.

Continued on next page

- **`ssize_t my_read(struct file*, char __user*, size_t, loff_t*)`**
    - 4 parameters:
        - A file struct pointer for the device file.
        - A pointer to the user-space buffer.
        - The size of the user-space buffer.
        - A pointer to the current offset in the device file.
    - Copies data from the kernel buffer to the user-space buffer. Updates the offset and prints the number of bytes copied.
    - Returns an integer of the size of the kernel buffer.

- **`ssize_t my_write(struct file*, const char __user, size_t, loff_t*)`**
    - 4 parameters: refer to my_read for parameter information
    - Copies data from the user-space buffer to the kernel buffer. Updates the offset and prints the number of bytes copied.
    - Returns an integer of the size of the kernel buffer.

- **`loff_t my_seek(struct file*, loff_t, int)`**
    - 3 parameters:
        - A file pointer for the device file.
        - The offset value.
        - The value whence. Describes how to interpret the offset (positive or negative).
    - Operation
        - If whence = 0 (SEEK_SET), the file position is set to the value of the offset
        - If whence = 1 (SEEK_CUR), the current file position is incremented (decremented if offset is negative) by offset bytes.
        - If whence = 2 (SEEK_END), the file position is set to offset bytes from the end of the file.
    - Returns the new offset value.

Installation

1) While in the code directory, build the LKM using the terminal

```
sudo make -C
/home/kernel.linux-hwe-4.15.0/CUSTOM_PATH/modules/charModule M=$PWD
```

2) Build the test program using the terminal

```
gcc -o test simple_character_driver.c SCDTest.c
```

3) Check for proper insertion of LKM

```
lsmod
```

4) Create a device file

```
sudo mknod -m 777 /dev/simple_character_driver c 240 0
```

5) Run the test script

```
./SCDTest
```