CSci 5103: Operating Systems

Project 4: Linux Device Drivers

Due: 10 PM May 1, 2012

Overview

In this project you (and your group) are asked to implement a character device called scullbuffer implementing a bounded buffer of fixed size to synchronize any number of producer and consumer processes. It will implement the character device interface. Both producer and consumer processes will first call the open function to access the device. At end of their use of the device, they will call the release function. The buffer will store “items”, where each item will be a block of up to 512 bytes and an integer count indicating the number of bytes in the block.

A producer process will open the device in “write” mode. It will call the write function to deposit an “item” in the buffer. If a producer makes the write call to deposit more than 512 bytes of data, the driver will accept only the first 512 bytes of data. The return value of the write

function will be the number of bytes written into a buffer item, or -1 if there was any error. It will return 0 if the buffer is full and there are no consumer processes. If the buffer is full and there are some consumer processes, then the producer process would get blocked in the write function.

A consumer processes will open the device in the “read” mode. It will call the read function to retrieve an item from the buffer. The read function will copy the data block of the next available item into the address-space of the process, at starting location specified in the read function call. The return value of this function will be the number of bytes copied from the device buffer to the process address-space. This function will block if there is no item currently available to be consumed and there are some producer processes. If the buffer is empty and there no producer processes, then the read call will return with value zero.

In your program, use counting semaphores (See Chapter 5 of the Linux Device Drivers book). You will allocate a buffer in the kernel memory using the function kmalloc to hold up to NITEMS, which will be a parameter to your driver and specified at the device installation time. You will be implementing the following functions: open, release, read, and write. As part of the open function, you will need to maintain the counts of the currently active producer and consumer processes. The functions read and write will perform the appropriate synchronization using counting semaphores.

A script has been included for installing and removing the scullbuffer device. This script must be modified in order to pass different arguements to your driver. How to pass arguements to your scullbuffer module must be decribed in a README included with your submission. To test your module, each group will be responsible for writing producer and consumer processes to test your device.

As with our other projects your submissions must compile and run on the course's virtual machine. It is advised that students make use of this virtual machine during development as bugs in kernel modules can result in undefined system behavior and leveraging the VM will ensure this unwanted behavior is isolated from the host OS.

Submission

- scullbuffer.c
- producer.c
- consumer.c
- Makefile for compiling your solution
- Scripts for installing and removing the scullbuffer device
- A README file including:
    - Your names
    - Your student IDs
    - Instructions on how to build your module, producer, and consumer (make or make all preferably)
    - Information regarding any installation arguements accepted by your module.