COSC440 Assignment 2 Report

Ashley Manson

Circular Buffer

My circular buffer keeps track of a head and count. The head is for the starting index for the bottom half to start copying from the buffer to the page list, while the count keeps track of the number of bytes stored in the buffer. I decided to drop incoming bytes for when the buffer becomes full rather than overwriting the already stored bytes in the buffer. When the bottom half copies from the buffer to the page list, the head will get incremented and modulus by the buffer capacity ensuring it circles around the buffer limit, and the count will get decremented.

Bottom Half

I have used tasklets to schedule for when the bottom half should start copying the data from the buffer to the page list. A couple reason for this is due to the fact that only one tasklet can be scheduled at any one time by the operating system. This has the added benefit of avoiding race conditions as they are more unlikely to occur. Another reason is because the raspberry pi's only have a single core CPU, and tasklets can only be ran on the same CPU that schedules them.

Reading

For reading each session I have session array that keeps track of each sessions sizes, which are calculated when the sessions are stored in the page list. When reading byte by byte, the number of bytes is summed up, and when the null terminated byte is found, the session is stored in the array. I have a read and write offset to calculate where in a page the reading and writing are supposed to occur. Once something has been read from the page, the read offset is incremented to the session size so that when trying to read again, the data that has already been read won't get read again. When all the data from a page has been read, it is removed from the page list, rather than recycling the page. I thought this would be easier than trying to reuse the page over again.
Unfortunately my read fails to read from more than one page, if going to another page for reading, the original data that was read is somehow overridden and the read returns garbage for some reason. I was unable to diagnose the problem to find a fix for this issue. However reading from just one page at a time works completely fine.

Race Conditions

I believe there could have been a possible race condition for when the top half and bottom half are accessing the circular buffer. If the top half was adding data to the buffer while the bottom half was copying it, the count variable could have been updated incorrectly by either method. I used a spinlock to hopefully avoid the issue of of incorrect book keeping of the count variable. Another race condition that could have arisen was when session where getting copied to the page list and read at the same time. I chose to use a wait queue to remedy this, the wait queue does not allow for the device to be read if there is no data actually stored in the page list, nor does it allow for the page list to be accessed when the bottom half is copying data to it or when it is getting read from at the same time.