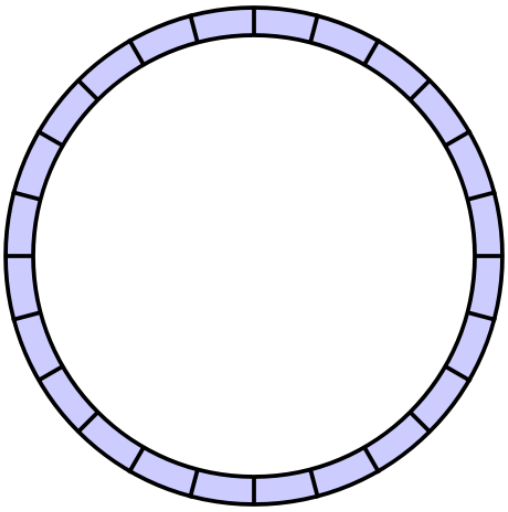
CMSC 328: Embedded Operating Systems

Thread-Safe Ring Buffers

Just as `malloc()` can be used to dynamically allocate contiguous chunks of memory, e.g. for variably sized collections, another common memory usage pattern that often appears in embedded systems is the ring or circular buffer. To store log data, debugging data, or intermediate results in a concurrent computation, a fixed-size FIFO queue is quite useful. For example, Linux's `dmesg` communicates logging information in the Linux kernel using a ring buffer.



In this lab, you are provided with a simple implementation of a ring buffer. Your task is to make this ring buffer thread-safe: multiple threads should be able to concurrently add and remove items to the queue using `rb_get` and `rb_put`. You should use mutexes and condition variables to provide thread-safe access to the ring buffer.

Testing

You can modify the Makefile to link the test programs against your thread-safe implementation (**ringbuffer_ts.c**) by adding one line to the Makefile:

```
TS=_ts
```

Deliverables

lastname_lab5/

ringbuffer_ts.c	your thread-safe implementation of ringbuffer.c
lab5.txt	Explain the differences between POSIX mutexes, condition variables and read/write locks.

Extra

- How could the provided ring buffer implementation be improved?
- How might the ring buffer be used for debugging and logging in embedded systems?
- Make your `bmalloc` and `bfree` thread safe.