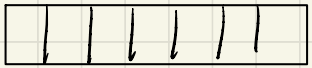
4/17/24

Producer Consumer / Bounded Buffer Problem



fixed size buffer of N entry

Producer = Put one item into the next empty slot, blocks until buffer has room to put

Consumer = Removes an item from the next nonempty slot, blocks until buffer is not empty.

```
Lock lk;
Item buffer[N];
int consume_ofs = 0;   // read offset
int produce_ofs = 0;   // write offset
int total_items = 0;   // available items in the buffer

produce(item) { }      consume() { }
```

```
Lock lk;  Item buffer[N];  Condvar not_full_cv;  Condvar not_empty_cv;
int consume_ofs = 0;  int produce_ofs = 0;  int total_items = 0;

produce(item) {
    lk.acquire();
    while (total_items == N) {
        not_full_cv.wait(lk);
    }
    buffer[produce_ofs] = item;
    produce_ofs = (produce_ofs+1) % N;
    total_item++;
    not_empty_cv.signal();
    lk.release();
}

consume() {
    lk.acquire();
    while (total_items == 0) {
        not_empty_cv.wait(lk);
    }
    item = buffer[consume_ofs];
    consume_ofs = (consume_ofs+1) % N;
    total_items--;
    not_full_cv.signal();
    lk.release();
}
```

→ Pipe: Interprocess Communication

→ pipe() returns 2 fds

read fd, write fd

→ can perform read / write on the fd

→ allows for partial reads & writes (read less than requested bytes)

* xv6 requires full atomic writes (must write all requested bytes)

for educational purposes → writes cannot interleave

→ Locks, monitors, reader writer lock

→ ways to synchronize threads accessing shared data

if all accesses are reads
do we need to synchronize?

if all accesses are writes
do we need to synchronize?

Top Comment

- mostly reads
- occasional writes

1st approach:

lock around every access (read & write),
poor performance (unnecessarily serializing reads)

2nd approach:

only lock writes, don't lock reads (good perf.),
reader could see partial writes (mismatching profile & comment)

✱ we want protected concurrent reads & exclusive write!

no write while reads are happening

mutual exclusion

Reader Writer Lock

→ APIs: read_acquire, read_release, write_acquire, write_release

↳ do we always want to allow new readers to join in?

→ Read Preferring vs Write Preferring

- allow new readers to read even if there are writers waiting
- wake up readers when there are waiting readers & writers.

- pause new readers if there are waiting writers
- wake up a writer when there are waiting readers & writers

Write Preferring Reader Writer Lock

```
Lock lk; Condvar reader_cv; Condvar writer_cv;
int active_readers=0; int waiting_writers=0; bool active_write=False;

read_acquire(){
  lk.acquire();
  while(active_write ||
   waiting_writers >0){
      reader_cv.wait(lk);
  }
  active_readers++;
  lk.release();
}
// holds the read lock
   upon success

write_acquire(){
  lk.acquire();
  waiting_writers++;
  while(active_write||
   active_readers >0){
      writer_cv.wait(lk);
  }
  waiting_writers--;
  active_write=True;
  lk.release();
}
// holds the write lock
   upon success

read_release(){}
write_release(){}
```