Buffers

Let’s go for a swim
Buffers

• A buffer is simply a collection of bytes
  – a char[] if you will.
• Any information that a program uses can be stored in a buffer.
• If you have to store a lot of information you may have more than one buffer to store the information
• It makes sense to organize all of these buffers in some sort of structure.
• A buffer pool
Buffer Pools

• A buffer pool can be as simple as an array of char[].
• You can use the buffers as you need them for quick access to the information if it is stored in the pool.
• If it’s not in the pool then you can put the information in it.
• The problem comes in when the pool is full.
Organizing Buffer Pools

- Which buffer should be replaced when new data must be read?
  - First-in, First-out: Use the first one on the queue.
  - Least Frequently Used (LFU): Count buffer accesses, reuse the least used.
  - Least Recently used (LRU): Keep buffers on a linked list. When buffer is accessed, bring it to front. Reuse the one at end.
Buffer Pool Implementations

• Two main options
  – Message passing
    • The pool user asks/gives the buffer pool for a certain amount of information and tells it where to put it.
  – Buffer passing
    • The pool user is given a pointer to the block they request.

• Both options have their advantages and disadvantages.
Design Issues

Disadvantage of message passing:
- Messages are copied and passed back and forth.

Disadvantages of buffer passing:
- The user is given access to system memory (the buffer itself)
- The user must explicitly tell the buffer pool when buffer contents have been modified, so that modified data can be rewritten to disk when the buffer is flushed.
- The pointer might become stale when the buffer pool replaces the contents of a buffer.
Buffer Pool Applications

- Usually, buffer pools are used to mediate between a program and disk access.
- A program will use a buffer pool to make requests for information.
- The buffer pool will go to the disk and get the information if it doesn’t have it.
Buffer Pool Strategies

• When a buffer pool needs to get the information it has a couple different options
  – It can get just the information it needs from the file
  – It can get the entire block with the information in it

• To the end user the result is the same, the requested information is returned to them
More on Strategies

• If you remember the access times for a byte and for a block:
  – 15.05 ms/byte
  – 15.1 ms/block
• It makes sense to grab the entire block at a time
• And then have some logic that goes in the buffer once it’s in memory and return the requested info
What does this mean for me?

• For major project 1 we are asking you to implement a buffer pool that will buffer individual records.
• Therefore, when a record is requested from the buffer pool it checks to see if it has that record:
  – If it does it returns that record
  – If not, it must go to disk and get it.