The third exam will be in class on Monday, April 20. If you have an unavoidable conflict with the date of an exam, please see me as soon as possible (before the exam date!) to discuss options for rescheduling. Last minute rescheduling/extensions will not be accommodated for something known about in advance.

The exam covers material from OSTEP chapters 21–23 and 26–29 (swapping, complete VM systems, and concurrency up through threads, locks, and thread-safe data structures but not condition variables and semaphores).

The exam is closed book. You may have a single page of notes (8.5x11", one side) which will be handed in with your exam. This page may be handwritten or typed and can contain whatever you would like, but it must be a hardcopy — on a piece of paper, not a laptop, tablet, phone, or other device — and must be personally prepared by you — you may not copy another student's page or hand out copies of yours to others. Creating your own notes is an essential part of the learning process — deciding what to include requires engagement with the material which reinforces understanding and improves long-term retention of the material, provides an opportunity for review in order to identify gaps in your knowledge in time to ask questions before the exam, increases confidence in what you do know, and encourages taking ownership of your own learning.

You should be familiar with the material covered in class and in OSTEP chapters 21–23 and 26–29. Key terms and ideas include:

- swapping
  - terminology: demand paging, swap space, swap partition, swap file, page fault, page replacement, thrashing, hit rate, compulsory miss, cold-start miss, capacity miss, optimal page replacement policy, Belady's anomaly, page selection, prefetching
  - concepts: why swapping is needed, why the page replacement policy matters
  - hardware and OS support for swapping
  - how page faults are handled (what happens, hardware and OS support)
  - general strategies for efficient handling of page replacement
  - specific page replacement policies: optimal, FIFO, random, LRU, LFU, clock algorithm (for each, what the policy is, key properties, when it performs well, when it performs poorly, other pros and cons)
  - other VM policies (besides page replacement): what the policies are for, alternatives
- complete VM systems
  - the VAX/VMS and 32-bit x86 Linux address spaces: layout, virtual addresses, mapping virtual addresses to physical addresses

  - policies: what each is, benefits (what problem(s) are being addressed), limitations/drawbacks
    * page replacement: segmented FIFO, second-chance lists, and clustering; modified 2Q
    * demand zeroing
    * copy-on-write

- concurrency
  - terminology: concurrency, critical section, race condition, mutual exclusion, atomicity
  - the challenges of concurrency when shared memory is involved
  - common synchronization needs: atomicity, ordering
- threads
  - terminology: thread
  - concepts: thread vs process, advantages of threads, when to use threads vs processes
  - reading knowledge of `pthreads` API for threads and locks (you should be familiar with the main API operations for creating and managing threads and locks as discussed in class and should recognize code using those operations if you see it, but you won't be asked to write specific syntax)
- locks
  - terminology: spinlocks
  - concepts: locks (what they are, why use them)
  - lock implementation
    * requirements for a working lock implementation
    * the main approaches (controlling interrupts, software-only, hardware supported) and the pros/cons/usefulness of each approach
    * specific implementations: simple flag (and its problems), spinlocks using test-and-set
    * performance considerations: pros and cons of spinlocks, avoiding spinning (specific strategies, pros and cons of each), two-phase locks (what they are, pros/cons)
  - priority inversion: what it is, how it can arise, how to address it
- thread-safe data structures
  - terminology: thread-safe
  - concepts: the problem (why thread safety is an issue, where problems arise with data structures)
  - the challenges (performance, scalability) and the sources of performance impacts

  - solutions and solution strategies: single lock, aggregating updates locally (approximate counting), finer-grained locking (linked lists, queues, hashtables)
    * how each works
    * pros and cons
  - lessons and design principles

Specific things you should be able to do include:

- swapping
  - trace a sequence of memory accesses and compute the hit rate under a particular page replacement policy
- concurrency
  - identify critical sections in C code
  - protect critical sections using locks

You should also have done homeworks 6–8 (from OSTEP chapters 21–22, 26–28) and be familiar with the concepts explored, the inputs analyzed, and the output of specific tools run — while the homework problems themselves are largely not suitable for an exam environment (and you won't be asked about specific command lines), you may be given similar inputs or outputs to work with or asked about concepts from those problems.