

Steve Martin
CS162 Fall 2003
Midterm II Review Topics

Note: All topics from previous midterm are also fair game! Anticipate a question or two from that section of the class.

1. Paging
 - a. Know basics
 - b. I.e. how to get address widths, what the levels are, etc.
 - c. 2 level paging
 - d. Including 2 level page tables and segmenting with paging.
 - e. Be able to do problems like homework question 1
2. Translation Lookaside Buffers
 - a. What is a TLB?
 - i. How fast are they?
 - ii. How big are they?
 - iii. How are entries searched?
 - iv. What exactly does it do?
 1. Be able to list exactly/draw a flow chart of what happens when there is a page fault with a TLB!
 - b. Where a TLB fits into the paging scheme
 - i. Why do we use them?
 - ii. Be able to draw a complete picture of the paging process using the TLB!
 - c. What principle does the TLB exploit in order to work so well with so few entries?
 - i. How does this work?
 - d. What extra overhead does the TLB add to context switches?
3. More on Page Faults
 - a. What is the principle of. . .
 - i. Temporal Locality?
 - ii. Spatial Locality?
 - b. Is there something the user could do when writing programs to minimize page faults?
 - c. What is the typical layout of a Page Table Entry (PTE), and what is each bit for?
 - d. What is a core map?
 - e. What is. . .
 - i. Paging out?
 - ii. Paging out a process?
 - iii. Paging in a process?
 - f. After a page fault, can a process be restarted directly?
 - i. How many page faults can occur in one instruction?
 - g. What is a. . .

- i. Page fetch algorithm?
 - ii. Page replacement algorithm?
 - iii. Page placement algorithm?
- 4. Page Fetch Algorithms
 - a. Demand Paging
 - i. What is this?
 - ii. Why is it non-optimal?
 - b. Request Paging
 - i. What is this?
 - ii. Why is this somewhat non-feasible?
 - c. Prefetching
 - i. What is this?
 - ii. What's the problem with it?
 - d. Overlays
 - i. What is this?
 - e. Be able to work each of these into any flowchart/description of paging!
- 5. Page Replacement Algorithms
 - a. Random
 - i. What is this?
 - ii. Why isn't it used?
 - b. FIFO
 - i. How does this work?
 - ii. What are its limitations?
 - 1. What is Belady's Anomaly—i.e. does giving this algorithm more pages always improve its performance?
 - c. LRU
 - i. What is this?
 - ii. What are some limitations in implementing it?
 - iii. How do we approximate it?
 - d. FINUFO/Clock
 - i. What is this an approximation of?
 - ii. What aspect of PTE layout does this algorithm take advantage of?
 - iii. How does it work?
 - iv. Drawbacks?
 - e. MIN/OPT
 - i. What is the exact definition of an optimal page replacement algorithm?
 - ii. Why can't we implement it?
 - f. Be able to draw a flowchart of how paging works with page replacement, TLBs, and any flavor of multi-level paging.
 - g. How do we evaluate paging algorithms?
 - h. What is the difference between global and local replacement?
- 6. Thrashing and Working Sets
 - a. What is thrashing?
 - b. How do we fix this problem?
 - i. There are several solutions!

- c. What is a working set?
 - i. Be able to define this formally
 - ii. Why is it useful?
 - iii. When is it useful?
- d. What is the Working Set Paging algorithm?
 - i. How is it implemented?
 - 1. Can we do it exactly?
- e. Page Fault Frequency algorithm
 - i. What is this?
 - ii. How can we implement it?
 - iii. Why was it invented?

7. Page Size

- a. What size should pages be?
- b. What are some pros/cons of large pages?
- c. Can we page the operating system?
- d. How do we study paging algorithms?

8. File Structure

- a. What is a file?
- b. How are files accessed?
 - i. Know the different types!
- c. What are the primary issues file systems should address?
- d. What are the usual operating characteristics of a file system?
 - i. Most files large or small?
 - ii. More reads or writes?
 - iii. What kind of file IO is most prevalent?
- e. How should the blocks of a file be placed on disk?
 - i. What are the pros and cons of each method?
 - 1. Be sure to know these!!
- f. How do we find the blocks of a file?
 - i. What is a File Descriptor? (inode)
 - ii. What is it used for?
 - iii. How does it vary with the method used to place file blocks on disk?
- g. How do we allocate a new block for a file?
 - i. How do we find the block?
 - ii. How do we allocate it?

9. I/O Optimization

- a. How do we make file operations fast?
 - i. What is the importance of block size?
- b. What characteristics of disk hardware most effect performance?
 - i. Know all of these!
- c. What if there is more than one outstanding disk request?
 - i. How should we schedule them?
 - 1. FCFS
 - 2. SSTF
 - 3. SCAN

4. C-SCAN

- ii. Know the differences, pros, and cons of each one of these
- iii. Are they all fair?
- iv. Are any of them optimal?
- d. What is rotational scheduling?
- e. What are some problems we can run into when reading specific blocks?
 - i. What is a solution to some of these?
- f. How can we minimize seek distance?
- g. What is a disk cache for?
 - i. How does it help performance?
 - ii. What are some of the drawbacks of a disk cache?
- h. What can we do at the file system level to increase performance?
 - i. What is. . .
 1. Pre-fetching?
 2. Data Reorganization?
 3. Data Replication?
 - ii. What are the pros and cons of each of these?
- i. What is RAID?
 - i. Why was RAID developed?
 - ii. What are its primary advantages?
 - iii. What is. . .
 1. RAID 0?
 2. RAID 1?
 3. RAID 3/4?
 4. RAID 5?
 - iv. What are the pros and cons of each of these RAID levels?