Final exam review
Wednesday, December 15, 2010  6:46 PM

Final format
  11 questions
  10 points each
  short answer

Basic design
  1/3 easy questions
  1/3 moderately difficult questions
  1/3 "unfair" questions

What you definitely need to know for the final:
scheduling and Little's laws
I/O and filesystems
literacy: explain why things were done
taxonomy of an OS
producer/consumer
knowledge boundaries
Basic principles of taking my exams:
  I tend to be rough on irrelevant answers.
  At best I ignore them.
  At worst I deduct for it.
  => You should insure that you're answering the question that I asked.
  => Read and revise your answer after you write it.
    Make sure it answers the question.
    Make sure it covers all possible interpretations.
    When in doubt, explain your interpretation of the question.
For short essay answers:
  Make an outline first.
  Critically assess the outline and whether it's complete.
  THEN write the paragraph.
  (use the liberal white space on the back for outlines. If you run out of time, I WILL give partial credit for an outline!)

To approach a problem:
  Determine what's relevant. Where in the course did the problem come from?
  Determine what's different.
  Make a decision on technique and implement.
Warnings:
There may be several correct answers (and an infinity of incorrect ones!)
There may be a "new twist" or not.
Twists may be obvious or extremely subtle.
How I grade essays:
  pick out and mark the relevant points.
  pick out and mark the incorrect statements.
  pile exams into equivalence classes.
  each equivalence class gets the same grade.

To get to the top of the pile
  Nail the relevant points.
  Avoid irrelevant/incorrect statements.
  Don't spend time on potentially irrelevant things.
Survival

Answer the questions you're sure about, first.
Avoid peripheral issues.
After an initial pass, "grade" each of your answers:
Did you cover the key points?
Is anything unsure?
Surviving a short-answer question
Get to the point fast!
Avoid writing down peripherally relevant things
Unless you have extra time...
Many review sheet/class exercise problems are "long answer"
- Open-ended situations.
- Often a complex answer.
- Lots of flexibility.

Many final exam problems are "short answer"
- More limited situations.
- Simpler answer.
- Less degrees of freedom.
I/O and producer/consumer

character write to a block device
Producer: write
Consumer: paging subsystem
   Reads block
   Writes character in context
   Marks block dirty
   Eventually flushes the block

This particular consumer gets reused a lot
How does it get used for ext2 filesystems?
Example: creating a file
   Allocate inode
   Mark inode used
   Allocate one block for inode
   Mark block used.
   Write something into the block.
   Write inode into directory block.
   update the super-block.
Be able to describe any common file operation this way: create, update, delete, rename, ....

Other questions
   how does one restore a journal?
   What does a journal do? Speeds up
recovery of consistency....
What is the difference between power fails on journaled and unjournaled systems

Answer: time in which changes are lost.
Scheduling and Little's law

When is a system in balance?
Average input rate = average output rate.

When does Little's law apply?
Only for systems in balance.

What happens when input rate > output rate?
The queue of jobs grows without bound.

What happens when input rate < output rate?
Approaches balance incrementally until input = output.

What is memorylessness?
M/M/1 means
memoryless arrival
arrival events are independent
memoryless processing
processing time is independent

Other scheduling topics
Kinds of scheduling
Fairness (or lack thereof)
O(1) scheduling -> variant of round robin
Completely fair scheduling (CFS)
Round-robin
Batch

Epochs:
An epoch is a scheduling phase, after which the scheduling algorithm repeats.
Know what the terms mean
   driver
   buffer
   cache

and the basic principles
   numbers, not names
   names are overlaid upon numbers via several mechanisms
   Disk directories
   Information services:
      /etc/passwd and LDAP for user NAMES.
      /etc/group and LDAP for group NAMES.

Why is there only one (unique) kernel descriptor per file?
   write is atomic
   each write resets the write pointer to the position after it.
   writes append.

Weird arrays
   driver table
   fixed during operations.
a literal array

block table in an inode (hierarchical array) expands efficiently to represent files with a Pareto distribution; large files are uncommon.

page table array for responding with page attributes.

segments can expand.
One removes a file by erasing its directory entry. What is the state of the file after this? A: blocks are unallocated but not overwritten.

How can one ensure that a file is gone? A: The program "eraser" can write over the exact file with garbage, multiple times.

How many operations does it take to create a file?

Normal disks wear out due to duty cycle, i.e., what time the power is on. Flash drives wear out due to write cycles, i.e., every block can be erased and rewritten only 1000 times. How does this change the design of a filesystem for flash?

(The "wear leveling" algorithm for flash: Maintain a map between logical and physical blocks. Never write the same physical block twice in a row. Instead, we reassign the logical block to a new physical block!)
Some kind of hashing:
   Input: logical byte address, process identity.
   Output: page descriptor.

The hash gives same answer for things in the same page.

The page descriptor contains:
   • Physical address of the page.
   • Logical address of the page + process identity
     \( <= \) recovers from hash collisions.
   • Read/Write, Read-only, Copy-on-Write.

In the OS, things are organized like this:

I need a very efficient way of storing this, that
   • allows me to add pages to a segment in \( O(1) \).
   • allows lookup in \( O(1) \)
Applying Little's Laws

- \( L = \lambda W \)
- \( W = \text{time in system} \)
- \( L = \lambda W \)
- \( W = \text{processing time} \)
- \( L^-1 = \lambda W \)
- \( W = \frac{L^-1}{\lambda} \)

The system

- \( L = 1 \)
Memorylessness means temporal independence:

- **memoryless arrivals**: the inter-arrival time of the next event does not depend upon the inter-arrival time of the previous event.

- **memoryless service**: the service time for an event does not depend upon the service time for a previous event.
The dynamic priority is real priority plus a number between 0 - 5
The priority determines slice length in the epoch.

Algorithm:
When a process blocks due to I/o, increase the dynamic priority (and thus the next slice time)
When a process runs to the end of its slice, decrease the dynamic priority.

This does not account for how much time was lost!
It only works if the time lost is an average.
Bitmaps function the following way:
The size of a word in the bitmap is the bus size. (64 bits)

Bit number n
bit word = n/64
bit offset in word = n%64.

Bit i is non-zero when bitarray[i/64] & (1<<(i%64)) is non-zero.
Difference between the raw disk driver and the filesystem driver.

Raw disk driver:
geometry determines track and sector counts. This is a lie.
So you ignore it and write everything in terms of block offset, starting at 0.

Even if there is any validity to the track sector map, the sequence map for blocks is statistically correct. With probability near 1, block x is next to block x+1

make a filesystem: decide
length of each block group.
number of inodes in the group.
number of blocks in the group.
(size of superblock)
tell these decisions to mkfs (or use newfs, which makes them for you)

mkfs partitions the disk into block groups, each with the decided number of blocks, inodes, etc. Spaced equally. If there's space left over, it is unused. Uses raw disk driver.

Then you mount the drive. This invokes the
filesystem driver.
   At this point, the raw driver is only used by the filesystem driver.
Effects of using a journal

<table>
<thead>
<tr>
<th>no journal</th>
<th>a journal</th>
</tr>
</thead>
<tbody>
<tr>
<td>window for consistency errors is 5 seconds</td>
<td>1/10 second</td>
</tr>
<tr>
<td>time for recovery (6 tb disk) is 6 hours</td>
<td>6 seconds</td>
</tr>
</tbody>
</table>

Having a journal does not eliminate consistency errors; it makes them less likely. Having a journal does not eliminate need for recovery; it just makes recovery go very fast!