University of California, Berkeley
College of Engineering
Computer Science Division – EECS

Spring 2006 Anthony D. Joseph

Midterm #2 Exam

April 26, 2006
CS162 Operating Systems

Your Name:	
SID AND 162 Login:	
TA Name:	
Discussion Section Time:	

General Information:
This is a **closed book and notes** examination. You have 90 minutes to answer as many questions as possible. The number in parentheses at the beginning of each question indicates the number of points given to the question; there are 100 points in all. You should read **all** of the questions before starting the exam, as some of the questions are substantially more time consuming.

Write all of your answers directly on this paper. ***Make your answers as concise as possible.*** If there is something in a question that you believe is open to interpretation, then please ask us about it!

Good Luck!!

Problem	Possible	Score
1	19	
2	36	
3	23	
4	22	
Total	**100**	

1. (19 points total) Short answer questions:
   a. (11 points) Translation Look-Aside Buffers:
      i) (6 points) On a TLB miss in Nachos, what data structures should you check to find a previously referenced page? *List the data structures in the order that they should be checked.*

      ii) (3 points) Give a two to three sentence description of "precise exceptions".

      iii) (2 points) Briefly (1-2 sentences) say why precise exceptions are nice to have.

   b. (4 points) Networking:
      i) (2 points) The current version of IP uses 32-bit addresses. List one reason that we cannot use all 2^{32} addresses.

ii) (2 points) List one reason why it would still be possible to connect more than 2^{32} computers to the Internet.

c. (4 points) Caching:

i) (2 points) What are the two types of locality exploited by caching?

ii) (2 points) Which kind of locality makes large cache lines a good idea, and why?

2. (36 points) File Systems and Disks.

a. (8 points) Consider a file system with 2048 byte blocks and 32-bit disk and file block pointers. Each file has 12 direct pointers, a singly-indirect pointer, a doubly-indirect pointer, and a triply-indirect pointer.
Express your answers in symbolic form for partial credit.

i) (4 points) How large of a disk can this file system support?

ii) (4 points) What is the maximum file size?

b. (12 points) Disk requests come into the disk driver for cylinders: 10, 22, 20, 2, 40, 6, and 38, in that order. The disk head is currently positioned over cylinder 20. A seek takes 6 milliseconds per cylinder moved. What is the sequence of reads and total seek time using each of the following algorithms?

i) (4 points) First-come, first-served:

ii) (4 points) Shortest Seek Time First:

iii) (4 points) SCAN (initially moving upwards):

c. (2 points) Briefly (2-3 sentences) state the difference between a hard link and a soft link.

d. (7 points) Read-ahead.

i) (3 points) Give a brief (2-3 sentences) description of read-ahead.

ii) (2 points) Briefly (2-3 sentences) state why read-ahead is useful.

iii) (2 points) Briefly (2-3 sentences) state what happens if you read ahead too much.

e. (7 points) RAID

i) (3 points) Give a brief (2-3 sentences) description of RAID 5.

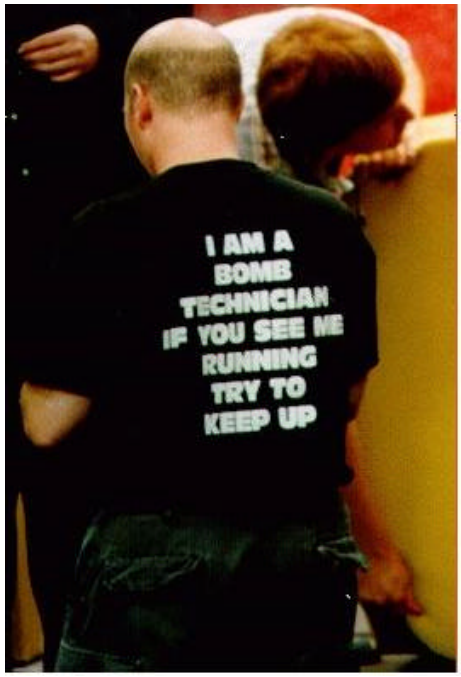
ii) (2 points) How many disk failures can RAID 5 tolerate without losing data?

iii) (2 points) How would you reconstruct a failed disk?

No Credit – **Problem X** (000000000000 points)

The importance of synchronization… or You're first – after me (This Only Happens in New York City).

Race Condition

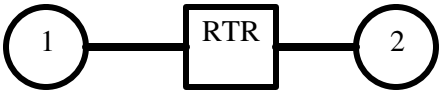


3. (23 points total) Reliable Peer-to-Peer (P2P) File Sharing.

Two computers running a P2P application using TCP/IP for connections are connected via a router (see below). To write data packets to the network, the P2P application issues a system call, and the OS copies the data first to a kernel buffer and then uses DMA to copy the data to the network controller board. Immediately after receiving the data, the controller generates one or more packets and sends them.

The packets travel one hop to router RTR, which will in turn forward the packets to their final destination. *Assume that no packets will be lost, and assume that the processing time at the router is negligible, and a router must wait for the entire packet to arrive before forwarding it.*

The receiving network controller uses DMA to copy the data it receives to memory, and when the last bit arrives and is transferred, it interrupts the CPU. The OS copies the data to the P2P application's buffer in user space. After the copy completes, the OS sends back a one-byte acknowledgement to the sending OS. *For simplicity, you may treat acknowledgements as if they are zero bytes long. Assume that a sender uses a send window-based acknowledgement protocol.*



System Parameters:

- Time to process an interrupt, T_{int} = 1.5 millisecond
- CPU cycle time, T_{CPU} = 10 nanoseconds
- Packet size, N_{pkt} = 1,000 bytes (ignore headers)
- Time to copy or DMA one byte, T_{copy} = 1 microsecond
- Latency of a link, T_{link} = 3 milliseconds
- Bandwidth of links, B = 8,000,000 bit/sec
- Send window N_{win} = 1 packet
- Retransmission timeout, T_{retran} = 400 milliseconds

a. (7 points) Compute T_{pkt}, the time to reliably send a single packet from a P2P application on computer 2 to a P2P application on computer 1 over a link with no losses.

b. (3 points) What is the maximum or peak rate at which one process can reliably send data to another process?

c. (4 points) In terms of T_{pkt}, how long does it take to reliably send an .mp3 file from computer 2 to computer 1 if the file is 4,000,000 bytes in size (assuming the links have no losses)?

d. (5 points) If we use a send window of 40 outstanding packets instead, what is the maximum rate at which one process can reliably send data to another process (assuming the links have no losses)?

e. (4 points) If we use a send window of 40 outstanding packets instead, how long does it take to reliably send an .mp3 file from computer 2 to computer 1 if the file is 4,000,000 bytes in size (assuming the links have no losses)?

4. (22 points total) Read-only Page Sharing.

To more efficiently use the physical memory allocated to processes, modern operating systems share read-only pages of different running instances of the same application (e.g., multiple Emacs processes). Your task is to implement this functionality for Nachos – sharing of read-only COFF pages in Nachos memory. Since we don't have actual inodes in Nachos, it's hard to have a real unique file handle, so it is sufficient if your answer provides a file name.

a. (6 points) List the data structures that you would need to add to the kernel:

b. (4 points) Name all the place where would you have to update these data structures:

c. (12 points) Design the function `load_coff_page()` that takes an executable name and virtual page number, and returns a physical page number if that COFF page is already in memory, or -1 if it must be loaded from disk.

```
int load_coff_page( String exec_name, int vpn ) {
```

This page intentionally left blank