University of Illinois at Urbana-Champaign
Department of Computer Science

Final Examination

CS 241 System Programming
Fall 2005

8:00-11:00 am Thursday December 15
1304 Siebel Center

Name:
NetID:

- Print your name and netid neatly in the space provided above; print your netid at the upper right corner of every page.
- This is a **closed book**, **closed notes** exam. No electronic aids are allowed, either.
- Eleven short answer, one implementation, one design essay; do all parts of every question.
- This booklet should include this title page, plus pages 1-12. Do your work inside this booklet, using the backs of pages if needed.
- Problems are of varying difficulty, hence if you don't know the answer immediately, progress to the following problem and come back to the unsolved problem later.
- Perfect syntax is not required, but any code you write must be understood by the graders. A few reference functions are included on the last page.

Problem(s)	Points	Score
1-3	15	
4-5	10	
6	15	
7-9	10	
10	10	
11	20	
12	10	
13	30	
Total	120	

1. (5pts) What is the difference between a physical memory address and a virtual memory address?

2. (5pts) Explain the difference between a library function and a system call.

3. (5pts) Why is it generally bad practice to acquire a lock in a signal handler?

4. (5pts) Explain Best Fit, First Fit, and Worst Fit. Argue for the exclusive use of one in a system.

5. (5pts) Compare and contrast linked list allocation and indexed allocation in a file system.

6. On a particular 64bit system with a 60gb hard disk, I-nodes are 512 bytes and disk blocks are 8kb. The I-node status information takes up 232 bytes. There is an address for a single indirect block, an address for a double indirect block, and an address for a triple indirect block; the rest of the space is for direct address pointers.

   (a) (2pts) How many addresses are needed to address a 64 kilobyte (2^{16}) file?

   (b) (4pts) How large of a file can be addressed by the double indirect block alone?

   (c) (4pts) Does a 4 megabyte (2^{22}) file use all of the addresses on its I-node? Any addresses in the single indirect block? double indirect? triple indirect?

   (d) (5pts) How much disk space (including overhead) is used to store the addresses required to access a 4 megabyte file (the answer involves more than one calculation)?

7. (4pts) Give more than one situation where use of a FIFO would be more desirable than a pipe.

8. (3pts) What C function is used to redirect standard output to a file?

9. (3pts) When is it proper to use select()?

10. (a) (2pts) How would the implementation of a web server using threads differ from one using processes?

    (b) (2pts) A web server is started listening on port 8080, receives 5 incoming connections, then properly and completely is shut down. How many close() system calls are there on a threaded implementation? How many on a multiprocessed implementation?

    (c) (6pts) Discuss how a multithreaded web server running on a single processor system could be optimized using the process scheduling methods discussed in class? Which do you recommend?

11. (20pts) Oh no! Andrew just accidentally deleted restart.h and restart.c! He is in desperate need of the copyfile() method. Complete the implementation below without using any restart library function calls. Add any helper functions you want. Informal descriptions will earn only partial credit.

```
#define BLKSIZE 4096
// int copyfile(int fromfd, int tofd)
//   Attempts to copy all readable bytes from file descriptor fromfd to file
//    descriptor tofd.
//   File descriptors are assumed opened before the call to copyfile(), and
//    shall not be closed by the function.
//   if successful returns the number of bytes copied from fromfd to tofd
//   in the event of an error other than EINTR (automatic restart on EINTR),
//    errno is set and -1 is returned
int copyfile(int fromfd, int tofd)
{
```

11. (continued.)

12. (10pts) For each set of threads below, determine if a deadlock state can be reached. If so, fix the deadlock, if not, give sufficient proof the deadlock cannot occur. Assume that `mutex_a` should be acquired before read and write on variable a, `mutex_b` for variable b, etc.

(a) **Thread 1**

```
lock(mutex_a)
lock(mutex_b)
b = a
unlock(mutex_a)
lock(mutex_c)
b = b + c
unlock(mutex_c)
unlock(mutex_b)
```

Thread 2

```
lock(mutex_b)
lock(mutex_c)
c = b
unlock(mutex_b)
lock(mutex_a)
a = c * 2
unlock(mutex_a)
unlock(mutex_c)
```

(b) **Thread 1**

```
lock(mutex_d)
lock(mutex_e)
e = d * 2
unlock(mutex_e)
unlock(mutex_d)
```

Thread 2

```
lock(mutex_e)
lock(mutex_f)
f = f + e
unlock(mutex_f)
unlock(mutex_e)
```

Thread 3

```
lock(mutex_f)
lock(mutex_d)
d = d * f
unlock(mutex_d)
unlock(mutex_f)
```

(c) **Thread 1**

```
lock(mutex_h)
lock(mutex_i)
lock(mutex_j)
j = i + h
unlock(mutex_j)
unlock(mutex_h)
lock(mutex_g)
g = g + i
unlock(mutex_g)
unlock(mutex_i)
```

Thread 2

```
lock(mutex_g)
lock(mutex_i)
i = g + g
unlock(mutex_i)
unlock(mutex_g)
```

13. (30pts) CS 241 Industries wishes to design a network memory server. Our programmers aren't very good, but if we give them the algorithm to follow, we believe they can create a good implementation from it.

    We seek to have a server with a lot of fast memory (4 gigabytes), that is running a simple **memoryServer** program. Clients should be able to connect to it over a very fast network connection, and then request memory that they can use to store and load data. The clients have only 64kb of local memory to use, and no disk from which to swap virtual memory. The client will have an instantiation of a **memoryClient** class which can be used to allocate more memory as needed, or transfer data to and from the server.

    Give all the details you can (network protocols, how and when to connect, if the applications are multithreaded, virtual memory sizes, how much memory to swap across the network for a "mapoffset()" call, etc) of how the algorithm should be structured to allow the server to service numerous connecting clients. **Be explicit and verbose in your response.**

    Also be sure to address these 2 questions:

    - What kind of function calls will the server need to implement?
    - What accessor methods will the client need and how will it run those on the remote machine?

13. (continued.)

(extra paper.)

Manual page

```
ssize_t read(int fd, void *buf, size_t count);
read() attempts to read up to count bytes from file descriptor fd into the
buffer starting at buf.  If count is zero, read() returns zero and has no
other results.
On success, the number of bytes read is returned (zero indicates end of file).
It is not an error if this number is smaller than the number of bytes
requested.  On error, -1 is returned, and errno is set appropriately.

ssize_t write(int fd, const void *buf, size_t count);
write() writes up to count bytes to the file referenced by the file descriptor
fd from the buffer starting at buf.
On success, the number of bytes written are returned (zero indicates nothing
was written).  On error, -1 is returned, and errno is set appropriately.
```