Final

CS 414 Operating Systems and Systems Competency Exam, Spring 2007
May 16th, 2007
Prof. Hakim Weatherspoon

Name (or Magic Number): _____NetId/Email:_____

Read all of the following information before starting the exam:
If you are a CS 414 student, write down your name and NetId/email NOW. Otherwise, if you are taking this as your Systems Competency Exam, write down your *Magic Number* (do ***NOT*** write your name, NetId, or Email).

This is a **closed book and notes** examination. You have 150 minutes (2 ½ hours) 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 a question is unclear, please simply answer the question and state your assumptions clearly. If you believe a question is open to interpretation, then please ask us about it!

Good Luck!!

Problem	Possible	Score
1	20	
2	24	
3	15	
4	21	
5	20	
Total	**100**	

1. (20 points) Synchronization/Concurrency Control

For the following implementations of the "H20" problem, say whether it either (i) works, (ii) doesn't work, or (iii) is dangerous -- that is, sometimes works and sometimes doesn't. If the implementation does not work or is dangerous, explain why and show how to fix it so it does work.

Here is the original problem description: You have just been hired by Mother Nature to help her out with the chemical reaction to form water, which she does not seem to be able to get right due to synchronization problems. The trick is to get two H atoms and one O atom all together at the same time. The atoms are threads. Each H atom invokes a procedure *hReady* when it is ready to react, and each O atom invokes a procedure *oReady* when it is ready. For this problem, you are to write the code for *hReady* and *oReady*. The procedures must delay until there are at least two H atoms and one O atom present, and then one of the threads must call the procedure *makeWater* (which just prints out a debug message that water was made). After the *makeWater* call, two instances of *hReady* and one instance of *oReady* should return. Your solution should avoid starvation and busy-waiting.

You may assume that the semaphore implementation enforces FIFO order for wakeups—the thread waiting longest in P() is always the next thread woken up by a call to V().

(a) (10 points) Here is a proposed solution to the "H20" problem:

```
Semaphore h_wait = 0;
Semaphore o_wait = 0;
int count = 0;

hReady()
{
    count++;
    if(count %2 == 1) {
        P(h_wait);
    } else {
        V(o_wait);
        P(h_wait);
    }

    return;
}
```

```
oReady()
{
    P(o_wait);
    makeWater();
    V(h_wait);
    V(h_wait);

    return;
}
```

(b) (10 points) Another proposed solution to the "H20" problem:

```
Semaphore h_wait = 0;
Semaphore o_wait = 0;

hReady()
{
    V(o_wait)
    P(h_wait)

    return;
}
```

```
oReady()
{
   P(o_wait);
   P(o_wait);
   makeWater();
   V(h_wait);
   V(h_wait);

   return;
}
```

2. (24 points) Synchronization via Monitors

Some monkeys are trying to cross a ravine. A single rope traverses the ravine, and monkeys can cross hand-over-hand. Up to five monkeys can hang on the rope at any one time. If there are more than five, then the rope will break and they will all fall to their end. Also, if eastward-moving monkeys encounter westward-moving monkeys, all will fall to their end. (This is the same problem setup from an earlier assignment, but the synchronization mechanism differs).

Assume that monkeys are processes.
(a) (18 points) Write a *monitor* with two methods WaitUntilSafeToCross(Destionation dst) and DoneWithCrossing(Destination dst). Where Destination is an enumerator with value EAST=0 or WEST=1.
(b) (6 points) Does your solution suffer from starvation? If so, briefly explain (e.g. give a sequence). Otherwise, simply state *starvation-free*. In either, case state your assumptions, if any.

3. (15 points) Virtual Memory and Paging

a. (5 points) Suppose that we have a two-level page translation scheme with 4K-byte pages and 4-byte page table entries (includes a valid bit, a couple permission bits, and a pointer to another page/table entry). What is the format of a 32-bit virtual address? Sketch the paging architecture required to translate a 32-bit virtual address (ignore the TLB).

b. (10 points) Assume that a process has referenced a memory address not resident in physical memory (perhaps due to demand paging), walk us through the steps that the operating system will perform in order to handle the page fault.

Note: We are only interested in operations that the OS performs and data structures modified by the OS.

4. (21 points) File Systems

Adding Links to a File System. This design question asks you to consider adding links to a file system that does not have any linking mechanism. ***This is a design question; you should not write code implementation unless stated explicitly.***

a. (12 points) The first set of questions is about adding **hard** links to the file system.

i. (3 points) What changes would you make to the internals (directory, fileheader, free map, etc.) of the filesystem to support hard links?

ii. (3 points) How do the semantics of the Remove system call change, and how do you implement that change to Remove and any other affected system calls?

iii. (3 points) What new system calls, if any, must be added to the system? Give the C language-style signature of any new calls, e.g., int Open(char *file).
List the signature and give a one sentence definition of each arg and return value.

iv. (3 points) Use the Remove system call and answer to question part iii above to show the implementation of the Rename system call.

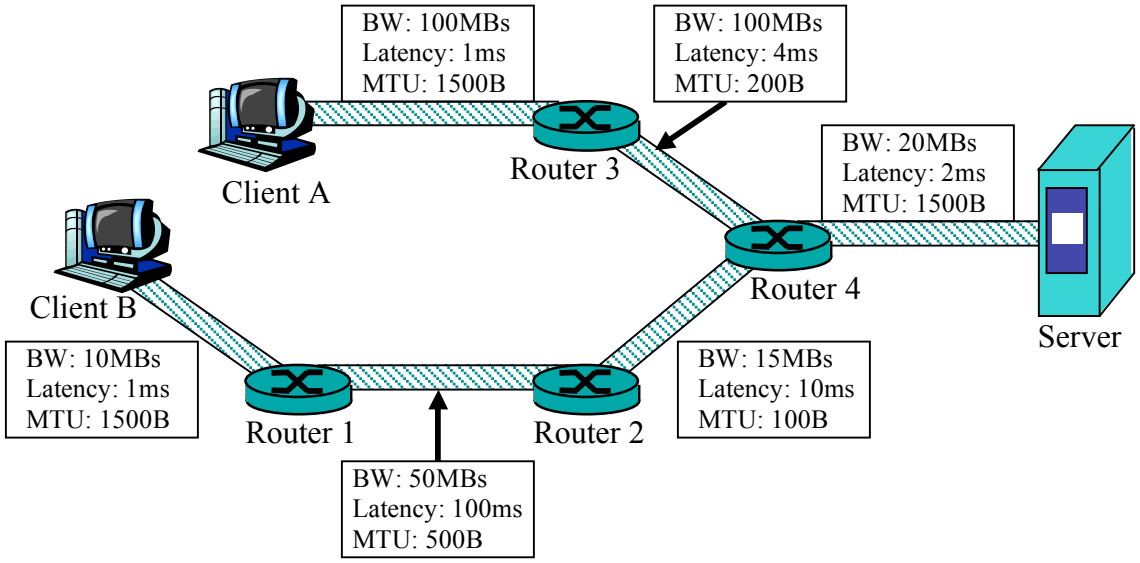
b. (9 points) This set of questions is about adding **soft** links to the file system. Soft links are also called symbolic links.

i. (3 points) What changes would you make to the internals (directory, fileheader, free map, etc.) of the filesystem to support soft links?

ii. (3 points) What existing system calls have to be modified to support soft links? How do the system calls need to be modified?

iii. (3 points) What new system calls, if any, must be added to the system? Give the C language-style signature of any new calls, e.g., int Open(char *file).
List the signature and give a one sentence definition of each arg and return value.

5. (20 points) Networking



The above figure illustrates a network in which two clients (Client A and Client B) route packets through the network to the server. Each link is characterized by its Bandwidth (BW), one-way Latency, and Maximum Transfer Unit (MTU). All links are full-duplex (can handle traffic in both directions at full bandwidth).

a. (4 points) Under ideal circumstances, what is the maximum bandwidth that Client A can send data to the server without causing packets to be dropped (Assuming that the headers are of zero length)? How about Client B? Explain.

b. (4 points) Keeping in mind that TCP/IP involves a total header size of 40 bytes (for TCP + IP), what is the maximum data bandwidth that Client A could send to the server through TCP/IP? Explain.

c. (4 points) Assume that Client A sends a continuous stream of packets to the server (and no other clients are talking to the server). How big should the send window be so that the TCP/IP algorithm will achieve maximum bandwidth without dropping packets? Explain. Hint: don't forget to account for the 40 bytes of header.

d. (4 points) Assume that Clients A and B both send a continuous stream of packets to the server simultaneously. Assume that Bandwidth is shared equally on shared links. What must the sizes of their send windows be so that packets are not dropped? Explain.

e. (4 points) Consider the situation of (5d). If the server is sending data back to Clients A and B at full rate, do the acknowledgements headed to the clients decrease the bandwidth in the forward direction (clients→server)? State your assumptions and explain your answer.