Operating Systems (CS422)

Spring 2004

Yale University

May 10, 2004

Name: ______________________ **Netid:** ______________

- Write your answers directly on the examination paper, including any work that you wish to be considered for partial credit. Use the back side of each sheet if you require more space to answer a question.
- Each question is marked with the number of points assigned to that problem. The total number of points is 50.
- The examination is open-notes and open-book.
- **Be brief in your answers.**

Problem	Topic	Points
1	File Systems	
2	Security	
3	Networking	
4	Dynamic Linking	
5	Concurrency	
6	Synchronization	
TOTAL		

Problem 1 *File Systems [8 points]*

When an operating system crashes, it often leaves the data that it stores on the disk in an inconsistent state. By inconsistent state, we mean that file system data structures do not correctly describe what should be in the files, directories, descriptors, and free lists. This can happen because it takes several disk reads and writes to update a file, its descriptors, and the free list. When the system comes up, it is necessary to validate that the file system and its data structures are all right (or figure out how to fix them).

With an allocation scheme based on block pointers and a free block list (such as that used in 4.1 BSD), is it possible to tell, after a crash, if a given block that is on the free list should not be? If so, how? If not, why not?

With an allocation scheme based on block groups and a free block bit map (such as that used in 4.2 BSD), is it possible to tell, after a crash, if a block that is marked "free" in the bit-map should not be? If so, how? If not, why not?

Suppose we know that a disk block appears both in the free list/map and in a file. Is there some action we could safely take? (I.e., we do not want to make matters any worse, and would like to make them better). If so, what? If not, why not?

Suppose we know that a disk block appears in two files at the same time. Is there some action we could safely take? If so, what? If not, why not?

Problem 2 *Security [8 points]*

Consider a distributed authentication protocol where a client A wants to communicate with the server B. The client initially contacts an authentication server AS, which manufactures a session key *CK* that A can use to communicate with B. Both the client and the server have a registered secret key with the authentication server. Let K_a be A's secret key that is known only to A and AS, and let K_b be B's secret key that is known only to B and AS. Consider the following authentication mechanism that has a few details missing.

Your task will be to fill in the missing pieces to make the system work. Notation: m^k means that the message m is encrypted with the key k.

1) A sends to AS the following message at time $t1$:

$$(A\ \ B\ \ t_1)$$

2) AS responds to A at time t_2:

$$(CK\ \ x\ \ (t_2\ \ t_3\ \ y)^u)^v$$

where x and y refer to message fragments and u and v refer to encryption keys. x and y could be the concatenation of one or more pieces of information. t_3 is the time until which the common key CK will be valid.

3) A sends to B the following message at time t_4:

$$(A\ \ request\ \ t_4)^w\ (t_2\ \ t_3\ \ y)^u$$

where $request$ is the command that A requires B to perform and w is an encryption key.

4) B responds to A with the message:

$$(reply\ \ t_4)^w$$

A. Fill in values for the following unspecified variables to design a secure authentication protocol:

- x:
- y:
- u:
- v:
- w:

B. If x is just an empty string, what forms of attack would the system be vulnerable to?

Problem 3 *Networking [6 points]*

A. Consider two nodes A and B communicating using a TCP stream. A is sending a large file to B using the TCP connection. Assume that the file is large enough for the TCP protocol to operate at the peak sending rate that the system can support. Provide a good upper-bound on the bandwidth of the TCP connection at steady state given the following properties of the system:

- The round-trip latency between A and B is l *secs*.
- The bottleneck bandwidth of some link between A and B is b *bytes/sec*.
- The amount of buffer space available at B is m *bytes*.
- The receiving program at B issues a system call to read n bytes every t seconds.
- The sending program at A can operate as fast as possible to push data through the connection.

Problem 4 *Dynamic Linking [8 points]*

Assume that you are using the Multics VM system (or a system similar to it) and that you are executing a process instantiated from the following program. (The process is created by invoking Prog:main.)

```
[Module: Prog.c]
void main()
{
    int x = Math.pi;         // access the pi variable from the math module
    int y = Math.sin(x);     // invoke the sin function
    Helper.help();           // invoke the help function located in Helper module
}
```

The Helper module looks like:

```
[Module: Helper.c]
void help()
{
    int z1 = Math.cos(0);    // invoke the cos function
    int z2 = Math.cos(z1);   // invoke the cos function
    /* Program point p */

    int z = Math.sin(z1 + z2);
    if (z > 0.5)
        Xwin.display(z);     // invoke a display method defined in Xwin module
}
```

The Math module looks like:

```
[Module: Math.c]
double pi = 3.14;
double sin(int ignore)  { return 0.0; }
double cos(int ignore)  { return 1.0; }
double tan(int ignore)  { return 0.0; }
```

Assume that the program is in the middle of executing the help() method and is currently at program point p. Answer the following questions regarding the virtual memory image of this process.

1) How many entries are contained in the linkage section for the module Helper.c? For what variables/functions do these entries serve as indirections?

2) How many link-traps have occurred since the beginning of program execution to program point p? What variable-accesses/function-calls caused these link traps? (A link trap occurs when a non-local reference (variable access or function call) is still in symbolic form.)

3) How many link-traps will occur from program point p to the end of program execution? What variable-accesses/function-calls caused these link traps? (Assume that the *display* routine does not cause any link traps.)

Problem 5 *Concurrency [8 points]*

A spanning tree is a subgraph with no cycles that includes all vertices. A spanning tree on n vertices has $n - 1$ edges.

Consider the following setting for the problem. There exists a spanning tree of a graph (as in the figure given below) with some of the edges included in the spanning tree (which are denoted by solid lines) and some not included in the spanning tree (which are denoted by dashed lines). Let there be multiple threads that are trying to update the spanning tree by replacing a link in the spanning tree by one which is currently not in the spanning tree. (This scenario arises in networks where links are replaced by better links as the system evolves.) For instance, a thread could swap in the edge (D, E) to replace the edge (B, E) and still preserve the spanning tree properties (namely, the tree contains no cycles and spans all the nodes in the graph).

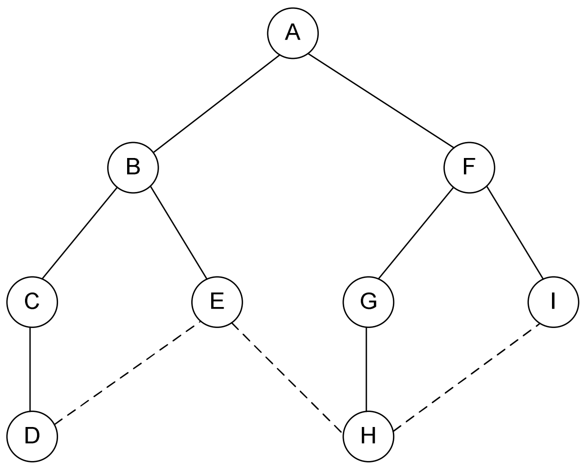

Figure 1: Sample Tree

A. In the figure given above, illustrate two kinds of problems that might arise when multiple threads are performing concurrent link replacements.

B. Give a locking algorithm that allows safe concurrent updates of the tree. Your solution should allow concurrent updates if they are non-interfering. A solution that locks the entire tree before performing an update is not a valid solution. Just provide a pseudo-code (or a high-level description) of the algorithm.

Problem 6 *Synchronization [12 points]*

Due to the Mother's Day rush, you have hired by the local department store to provide synchronization for their revolving door. Your job is to keep people flowing in both directions through the door. Here are the constraints:

1. The door has four (4) compartments. Each compartment can hold, at most, one (1) person. The door only revolves in one direction.
2. People wait in line on either side of the door until an empty compartment is available. A person must exit from a compartment before the next person can enter it.
3. If people are waiting both inside and outside, then one person from each direction should move through the door at each turn.
4. You have available to you (already written) a procedure called TurnDoor(), which rotates the door by 90 degrees. You have no idea how long this procedure takes to execute. If you rotate the door while a person is getting in or out, you will squash them.
5. You also have available to you (already written) procedures called EnterDoor() and LeaveDoor(), which moves a person calling this routine into and out of the revolving door. You have no idea how long these procedures take to execute.
6. When your program starts, assume that the door is completely empty and that open compartments are facing the inside and outside. Also, assume that no one is waiting in line.

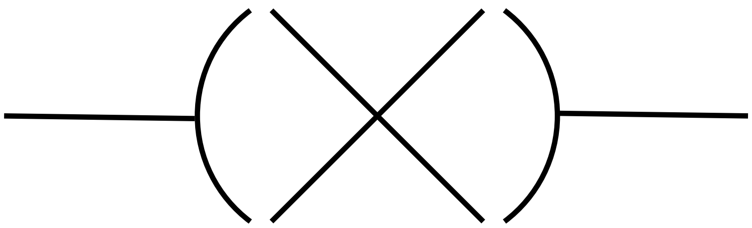

Figure 2: Revolving door with four compartments. One compartment faces out, one faces in, and two are closed.

You are to write the code for EnterStore() and LeaveStore() procedures. These procedures are called by a person (process) when they (it) want to enter or leave the store. These procedures will return when the person (process) has safely entered or exited the store. You also have to write the code for any procedure that EnterStore() and LeaveStore() may need to call (except TurnDoor()).