Final Exam

11 May 2011

1. For each of the statements below, fill in the blank with the *best* term from the following list. Some terms might be used more than once, some might not be used at all.

   • authentication • authorization • deadlock • hard link • inode • mount
   • random access • semaphore • sequential access • symbolic link

   (a) A(n) _____ is a synchronization tool with two operations: *wait* and *signal.*

   (b) _____ refers to a state in which a set of processes cannot make *progress,* because they're each waiting on a resource that another one holds.

   (c) _____ is the command to graft a storage device onto an existing file system at some specified location.

   (d) _____ refers to reading or writing parts of a file in no particular order.

2. In CPU scheduling, what is the meaning of *starvation?* Is it possible for a process to starve in the *first-come first-served* scheduling strategy? What about the *shortest-job first* strategy?

3. Below is a fragment of code to determine the number of times each letter of the alphabet appears in a message.

```
1  // Shared variables
2  const int MAX = 26;
3  int current = 0;
4  struct { char letter; int count; } frequencies[MAX];
5
6  // Function to be called with different letters from
7  // different threads.
8  void count_and_update(char message[], char letter)
9  {
10     // Determine number of occurrences (n)
11     // by looping through message.
12     int i, n = 0;
13     for(i = 0; message[i] != '\0'; i++)
14     {
15         if(message[i] == letter)
16         {
17             n++;
18         }
19     }
20     // Update the shared data structure
21     if(n > 0)
22     {
23         frequencies[current].letter = letter;
24         frequencies[current].count = n;
25         current++;
26     }
27 }
```

Given the message "ABSTRACT ALGEBRA," the first part of the `frequencies` table might look like this:

	`letter`	`count`
`frequencies[0]`	`'A'`	4
`frequencies[1]`	`'B'`	2
`frequencies[2]`	`'C'`	1
`frequencies[3]`	`'E'`	1
`frequencies[4]`	`'G'`	1
`frequencies[5]`	`'L'`	1
`frequencies[6]`	—	—
`frequencies[7]`	—	—
`frequencies[8]`	—	—
…	…	…

where `current` has the value 6 (the index of the next unused slot in `frequencies`).

The entries mean that the letter `'A'` appears four times in the message, `'B'` appears twice, and the other letters shown appear once each.

Now suppose that two threads are concurrently working on the letters `'R'` and `'S'`. They should find that there are two occurrences of `'R'` and one of `'S'`, but the results may depend on the way instructions from both processes are interleaved!

(a) Mark each sequence below as valid or invalid, meaning: will it produce a coherent result in the `frequencies` table?

i. `R21 R23 R24 R25 S21 S23 S24 S25` _____

ii. `S21 R21 S23 S24 S25 R23 R24 R25` _____

iii. `S21 S23 R21 R23 S24 S25 R24 R25` _____

iv. `R21 R23 R24 S21 R25 S23 S24 S25` _____

(b) Which lines are part of the *critical section* within this function?

(c) Fix the concurrency bug by adding a *semaphore* to the code. How will the semaphore be initialized? Where will you place the calls to *wait* and *signal?*

4. This question is about page replacement strategies in memory management. Suppose a process needs to access the following sequence of logical pages, but there are only *three* physical frames:

X 0	X 1	X 2	3	0	4	0	1	0	2	0	1	0	4	2
0	0	0												
	1	1												
		2												

(a) Using *least-recently used* page replacement, show the contents of the three frames beneath each page request. The first three (demand-paging phase) are done for you. Mark *page faults* with an **X** – how many are there?

(b) Does the LRU strategy make any poor decisions, leading to more page faults than necessary? Using foresight of what page requests are coming in the future, show a set of replacements that result in fewer page faults.

X 0	X 1	X 2	3	0	4	0	1	0	2	0	1	0	4	2
0	0	0												
	1	1												
		2												

(c) Increasing the amount of physical memory can also decrease the page faults. Do the LRU strategy again, but this time with *four* physical frames.

X 0	X 1	X 2	X 3	0	4	0	1	0	2	0	1	0	4	2
0	0	0	0											
	1	1	1											
		2	2											
			3											

5. This question refers to an *inverted index,* the data structure we explored in the last assignment for building a search engine.

   Suppose the user types the following query into the search box:

   ```
   colorless green ideas
   ```

   Looking in our inverted index, each word produces the following matches. (In this case, each match is a pair of the document ID and the location *within* the document.)

   colorless: (doc 3, line 30); (doc 3, line 56); (doc 5, line 2); (doc 8, line 80)

   green: (doc 4, line 9); (doc 5, line 2); (doc 8, line 6); (doc 9, line 10); (doc 9, line 13)

   ideas: (doc 2, line 1); (doc 2, line 24); (doc 2, line 30); (doc 4, line 10); (doc 5, line 3); (doc 8, line 80); (doc 9, line 4)

   (a) Which documents would the search engine return as matches, using the data above?

   (b) Suggest a way to *rank* the documents (most relevant to least relevant) using only the data above. Which match is most relevant?

6. This question describes a simple file system. On this device, data is stored in *blocks* of $2^8 = 256$ bytes each. Pointers to blocks are two-byte (16-bit) values, so it supports devices up to 2^{16} blocks, or $2^{16} \times 2^8 = 2^{24}$ bytes.

   The first $2^5 = 32$ blocks, however, are reserved as fixed *directory blocks*, called the *file allocation table* (FAT). These contain a series of directory entries of the following format:

File name	12 bytes
Direct pointer	2 bytes
Indirect pointer	2 bytes

   The direct pointer is the number of the first data block (where all 256 bytes are used to store the file contents). The indirect pointer is the number of an *index block,* which contains no data, but just pointers to other data blocks.

   (a) What is the size, in bytes, of the largest file that can be created on this file system?

   (b) What is the maximum number of files that can be stored in the FAT?

   (c) Suppose the FAT contains the following entry:

File name	Direct	Indirect
`HOMEWORK.TXT`	416	820

   and that the index block 820 contains the following pointers:

720
719
417
821
…

   What block(s) would we need to read to find the content of offset 528 in the file?