CoEn 177 Fall 2012 Final Examination - Section 01

Tuesday 4th December 2012

E-Mail:	@scu.edu

Engineering Honor Code: "All students taking courses in the School of Engineering agree, individually and collectively, that they will not give or receive unpermitted aid in examinations or other coursework that is to be used by the instructor as the basis of grading"

____
Student Signature

- This exam is closed-everything (this includes notes, PDAs, and phones), please note and acknowledge the honor code above.

- You may use calculators, but you do not need to simplify numerical answers, so please do not use the calculator apps on phones.

- Attempt all questions, and show partial work.

- If you are unsure of the meaning of a question, feel free to ask the instructor for clarification, or if you are still uncertain … please be sure to state any assumptions you need to make.

- Explicitly state any assumptions you make when answering a question.

- Good Luck!

Login Name:

Question 1 (5 pts): What is the benefit of employing C-SCAN instead of SCAN for disk head scheduling?

```
For this question, we are looking for answers that explain why
satisfying requests while traveling in only one of the two possible
directions is a good thing. Specifically, this would be due to the
increased fairness (or evenness of waiting time) for visits to tracks
at the edges compares to tracks at the center of disks.

(recall the analogy of a looping bus route compared to a back-and-
forth bus route)
```

Question 2 (10 pts): Assume that you have a disk with 16 tracks, and the head is parked on track 4. It takes 1 time unit to move the head from a track to an adjacent track. At time *t=0* we have a queue of requests for blocks on tracks 2, 6, 6, 9, 7, 11, 5, 3, 4. All these tracks need to be visited, and no new requests arrive. How much time will it take to visit all the tracks requested above using each of the following algorithms (when you need to, assume activity is only performed as the arm moves towards higher-numbered tracks):

(the answers below were corrected, as they did not take into account the need to visit "11", and in part (b) the erroneous answer also assumed C-SCAN instead of SCAN)

a) SSTF

~~Visit order: 4, 3, 2, 5, 6, 7, 9 (time = 10)~~
~~or alternatively (& equally correct): 4, 5, 6, 7, 9, 3, 2 (time = 12)~~
Visit order: 4, 3, 2, 5, 6, 7, 9, 11 (time = 11)
or alternatively: 4, 5, 6, 7, 9, 11, 3, 2 (time = 16)

b) SCAN

~~Visit order: 4, 5, 6, 7, 9, 15, 0, 2, 3 (time = 29)~~
~~or alternatively (& equally correct): 4, 3, 2, 0, 5, 6, 7 (time = 11)~~
Visit order: 4, 5, 6, 7, 9, 11, 15, 3, 2 (time = 24)
or alternatively: 4, 3, 2, 0, 5, 6, 7, 11 (time = 15)

c) C-LOOK

~~Visit order: 4, 5, 6, 7, 9, 2, 3 (time = 13)~~
Visit order: 4, 5, 6, 7, 9, 11, 2, 3 (time = 17)

Login Name:

Question 3 (12 pts): Given the following page access sequence, count the page faults for each of the following memory capacities and replacement algorithms. Access sequence: `0, 1, 2, 3, 4, 0, 2, 3, 0, 0, 1, 1, 1, 2, 3, 3, 1, 7, 8, 8, 7, 7, 6, 5, 6`

a) using LRU Page Replacement, and a 3-page memory
b) using LRU Page Replacement, and a 24-page memory
c) using Second Chance Page Replacement, and a 3-page memory

```
There are no real shortcuts to parts a and c in this question, I fear
they must be spelled out in a manner similar to this (with appropriate
updating of the lists of memory contents at each step):

a)

0: [0,-,-] (page fault)
1: [1,0,-] (page fault)
2: [2,1,0] (page fault)
3: [3,2,1] (page fault)
4: [4,3,2] (page fault)
.
.
.

b) Here we have some good news, as the memory capacity is greater than
the number of unique pages visited. So we can immediately say that the
total number of page faults = the number of unique pages visited
(i.e., 9 page faults).
```

Login Name:

Question 4 (12 pts): The block size of a file system is 8K bytes (a character occupies one byte). Each disk can have no more than 16 Exa (16 x 2^{60}) blocks (not bytes … that's the total number of **blocks**). Assume that the file system described above uses *i-nodes*, and that each is a single block, and has 2-K bytes of attribute/header information (the remaining space in the block is used for index pointers). What is the maximum file size you can represent ***on a single disk*** (you need to calculate the size of a block pointer for such disks) if these pointers are:

```
The block pointers are 64-bits, i.e. 8 bytes, in length (since each
disk can theoretically have 16 exa blocks)
```

a) all direct pointers?

```
6KB / 8 bytes = (6/8) K pointers to blocks. Each block is 8KB of data,
so the maximum file size is (6/8)K * 8KB = 6MB
```

b) all single-indirect pointers?

```
In this case each pointer (of the 6/8 K pointers) will point to a
block holding 8KB worth of pointers. That is 8KB/8B = 1K pointers.

So maximum file size is 6MB * 1K = 6GB.
```

c) all triple-indirect pointers?

```
Same as (b), but now each pointer in the index node points to a block
of double indirect pointers (1K), each pointing to a block of single
indirect pointers (1K), each pointing to a block of direct pointers
(1K), each pointing to 8KB of data.

i.e. (6/8)K * 1K * 1K * 1K * 8KB = 6PB
```

d) Finally, what is the maximum file size if an allocation table is used instead of index nodes? (Hint: the pointers in the allocation table, i.e., its entries, are each the same size as pointers that you would use with an index node).

```
Short imperfect answer: size of the disk.
Short perfect answer: size of the disk - size of the allocation table
itself.
```

Login Name:

Question 5 (4 pts): A disk head is at track 10, and an lseek() system call is made. The file system uses a RAM-resident allocation table, and has a single directory that is always cached in RAM. The new current position of the file can be located on a sector on track 1010. If a disk arm can move 100 tracks per millisecond, how long is the **minimum** delay for returning from this lseek() system call? from a subsequent read() system call?

Returning from the lseek() will take little longer than the context switches, since no disk movement is required to update a variable (current position) in memory.

Returning from the read() is dependent on whether the data is found in a cache or not. If it is cached, then the read will not take much longer than the lseek() [since we are assured that the allocation table is entirely in RAM, the only disk access needed is to retrieve the data blocks if they are not cached]. So the minimum is negligible if the data is cached, but the minimum is 10 milliseconds if it is not (1010-10/100 ... i.e., distance/speed = duration).

Question 6 (5 pts): LRU is impractical for virtual memory page replacement, but practical for storage caches. Why is this true?

Here you are expected to point out the cost of implementing LRU (the need to update an in-memory data structure **with every access to a new page in memory**).

Question 7 (6 pts): PGP uses public key encryption, secret key encryption, and one way secure hash algorithms. Why does it need the secret key algorithm (i.e., the symmetric key algorithm)?

In a word: performance,
or in a couple of more words: efficiency of implementation (since secret key encryption is faster, and in PGP we need only use public key encryption for the sharing of the secret key).

Login Name:

Question 8 (8 pts): ***TRUE/FALSE***

a) **[2pts]** We can store file attributes in an index node, but not in a typical allocation table. **`True`**

b) **[2pts]** DMA-driven I/O results in less load on the main CPU than programmed I/O. **`True`**

c) **[2pts]** An operating systems designer claims that his operating system has an authentication mechanism that cannot be thwarted or bypassed in any way. It turns out that his claim is true, and so we should believe that this operating system is therefore secure. **`False (as secure means more than simply authenticated correctly. Good OS authentication does not guarantee, for example, that all applications running as that individual user are secure and cannot be subverted, or that the system is immune from tampering with its I/O or being subjected to a denial of service attack).`**

d) **[2pts]** You can create a secure operating system using encryption. And by "secure" we mean an operating system guaranteed to be free of security vulnerabilities. **`False (for the same reason as question c).`**

Question	Possible	Grade
1	5	
2	10	
3	10	
4	12	
5	4	
6	5	
7	6	
8	8	
	60	

Login Name: