EECE.4810/EECE.5730: Operating Systems

Spring 2018

Exam 3

May 11, 2018

Name: ____

Section: EECE.4810 (undergraduate) EECE.5730 (graduate)

For this exam, you may use one 8.5" x 11" double-sided page of notes. All electronic devices (e.g., cell phones, calculators, laptops) are prohibited. If you have a cell phone, please turn it off prior to the start of the exam to avoid distracting other students.

The exam contains 4 questions for a total of 100 points. Please answer the questions in the spaces provided. If you need additional space, use the back of the page on which the question is written and clearly indicate that you have done so.

Please note that students enrolled in EECE.5730 must complete two extra problems, which are worth a total of 15 points:

- Question 2c, on page 5
- Question 3d, on page 8

You will have three hours to complete this exam.

Q1: Segmentation	/ 18
Q2: Paging	/ 36 + 7
Q3: File systems	/ 31 + 7
Q4: Protection and security	/ 15
TOTAL SCORE	/ 100 + 15

1. (18 points) ***Segmentation***

Write the function below, which could be used in a program simulating segmentation as a memory management method. The function uses this structure for each segment table entry:

```
typedef struct {
   unsigned char valid;        // Valid bit—either 0 or 1
   unsigned int base;          // Segment base address
   unsigned int bound;         // Segment upper bound (or size)
} segTableEntry;
```

The three function arguments represent the segment table (the array `ST`), the segment number (`segNum`), and the offset into the segment (`segOff`). The function should use the segment number to index into the segment table and return the physical address. If an error would occur in the address translation, the function should return 0. You may assume the segment number is valid and that it does not go outside the array boundaries.

The comments in the function outline how it should behave.

```
unsigned int translate(segTableEntry ST[], unsigned int segNum,
                       unsigned int segOff) {
   // Variable declarations





   // Access correct segment table entry & check error conditions
   // If no errors, calculate & return physical address










   // If an error would occur in the translation, return 0






}
```

2. (36 <u>+ 7</u> points) ***Paging***

a. (18 points) Explain what conditions would create the slowest possible translation for each of the page table organizations we discussed. (For example, if a tree-based page table existed, the slowest possible translation would happen if the tree was imbalanced such that every node had only one child, and all nodes had to be searched to find the right page table entry.)

You should not give a numeric answer, but may provide an example to explain your answer.

i. Two-level page table

ii. Hashed page table using chaining

iii. Inverted page table

2 (continued)

b. (18 points) The clock algorithm for page replacement uses a circular linked list (a linked list in which the last node points to the head of the list) to track currently used pages and to find a replacement candidate. Assume the following structure represents each node in the list:

```
typedef struct n {
   unsigned char ref;          // Reference bit—either 1 or 0
   unsigned int pageNo;        // Page number
   struct n *next;             // Pointer to next node
} clockNode;
```

Complete the function below, which takes as an argument the current "clock hand" (a pointer to the first node to consider for replacement) and returns the address of the node representing the page to be replaced. The comments in the function outline how it should behave.

```
clockNode *findRepl(clockNode *clockhand) {

   // Variable declarations





   // Start at current "clock hand" and loop until node found
   //   with ref == 0

   while (____________________________________________________) {

      // Clear reference bit for current node, then move to next
      //   node in list










   }

   // When loop ends, should have pointer to node with reference
   //   bit == 0, so return that pointer



}
```

2 (continued)

c. (7 points, ***<u>EECE.5730 only</u>***) In paged memory management, is the operating system involved in every address translation? If so, explain how; if not, explain what the role of the operating system is in a system using paged memory management.

3. (31 **<u>+ 8</u>** points) ***<u>File systems</u>***

a. (17 points) Given a system using FFS as the file system, if the block size is 2 KB and the address size is 64 bits, what is the maximum possible file size?

Remember, each FFS inode contains 12 direct pointers to disk blocks, as well as three other pointers—an indirect pointer, a doubly indirect pointer, and a triply indirect pointer.

<u>You do **NOT** need to give an exact numeric result</u>—you can express your result as a sum of powers of 2 (for example, 2^{16} bytes) or numbers of KB, MB, GB, or TB (for example, 64 KB), since the exact value would be difficult to find without a calculator.

<u>However, you must show all of your work to earn full credit.</u>

3 (continued)

b. (6 points) Why is the FAT file system typically only used on smaller disks?

c. (8 points) In a log-based file system, explain (i) what gets written to the log, and (ii) how logging allows a file system to tolerate crashes and ensure changes to the file system are completed properly.

3 (continued)

d. (8 points, ***EECE.5730 only***) Explain (i) how file systems that use a linked list to manage free space can make the list faster to search, and (ii) what aspect of free space management becomes slower if you make this change to make list searching faster.

4. (15 points) ***Protection and security***

a. (9 points) Given the access matrix below, answer questions (i)-(iii).

Domains	**Objects**						
	F_1	F_2	F_3	D_1	D_2	D_3	D_4
D_1	read	owner					
D_2	write* execute	read*	owner	control			
D_3			write				control
D_4	owner	write*					

i. What privileges can a process in domain D_4 grant to a process in domain D_2 for object F_2? Explain your answer.

ii. Can a process in domain D_1 revoke the write privileges of a process in domain D_2 for object F_1? Explain why or why not.

iii. Can a process in domain D_2 grant read privileges for object F_3 to a process in domain D_1? Explain why or why not.

4 (continued)

b. (6 points) Explain one benefit of symmetric encryption over asymmetric encryption, and one benefit of asymmetric encryption over symmetric encryption.