Name:________________________________

CSCI 3753: Operating Systems
Fall 2016

Final Exam

12/12/2016 (1:30 – 3:00 PM)

Answer all questions in the space provided

Multiple Choice Questions: Choose one option that answers the question best.
[30 Points]

1. In load time binding,

   - A. CPU only refers to the logical addresses of a process.
   - B. absolute physical addresses are stored in the object code produced by a compiler.
   - C. the compiler needs to know the physical memory address where the process will be loaded.
   - D. a process that was preempted earlier may be loaded anywhere in memory when it is scheduled again.
   - **E. None of the above**

2. Difference between static linking and dynamic linking is

   - A. processes have access to the most recently updated libraries in static linking but not in dynamic linking.
   - **B. dynamically linked processes are typically smaller in size compared to the statically linked processes.**
   - C. multiple statically linked processes may share a single copy of a library, while dynamically linked processes cannot.
   - D. dynamically linked processes run faster compared to the statically linked processes.
   - E. an advantage of dynamically linked libraries is that the library code is position dependent and so more efficient.

3. Which of the following statement is FALSE about authentication?

   - A. OS can block or slow down access after too many failed login attempts
   - B. OS can force users to change their passwords frequently
   - C. OS doesn't store user passwords in clear text
   - D. OS can force users to choose hard-to-guess passwords
   - **E. Authentication mechanisms can control fine-grained access to the files stored in the file system**

4. Which of the following is FALSE about inverted page tables (IPTs)?

    - A. There is a single page table shared by all running processes
    - B. It's hard to implement shared memory pages with IPTs
    - C. IPTs use hash tables for efficient lookup
    - D. Each entry in an IPT includes a process id
    - **E. IPTs use the same amount of memory as single-level page tables**

5. RAID

    - A. provides faster read/write access than single disks
    - B. provides better fault tolerance
    - C. saves disk space by compressing data
    - **D. A and B but not C**
    - E. B and C but not A

6. In memory mapped files

    - A. parts of a file on disk are mapped to pages in the logical address space of a process
    - B. after the first access, all subsequent reads/writes from/to a file are fast
    - C. Multiple processes can map the same file concurrently and share efficiently
    - **D. A, B and C**
    - E. A and B, but not C

7. Advantages of flash memory over magnetic disk include

    - A. lower access latency
    - B. lower power
    - C. more resistant to kinetic shock
    - D. more compact
    - **E. All of the above**

8. Which of the following statements is FALSE about access control lists (ACL)?

    - A. ACLs are used in Linux
    - B. Determining the set of access rights for a given file is easy
    - **C. Determining the set of access rights for a user process is easy**
    - D. Access permissions can be stored in File Control Blocks
    - E. Space needed for ACLs is similar to the space needed to store capability lists

9. Symbolic links

- **A. can create loops in a directory structure**
- B. point to the location in the disk where a file is stored
- C. duplicate files or directories for sharing
- D. rely on file permissions for sharing
- E. None of the above

10. A system-wide open file table stores

- A. read/write pointers of all open files
- B. an open count of the number of processes that have a file open
- C. the file control blocks of open files
- **D. B and C but not A**
- E. A, B and C

Short Answer Questions:
[30 Points]

1. Describe the SSH protocol in detail, i.e. describe the initial handshaking phase as well as the subsequent data messaging phase. When a user supplies their login password, is it encrypted by a public key or a symmetric key, and why? Explain why or why not SSH is resilient to eavesdropping attacks, man-in-the-middle attacks, and/or replay attacks.

1. user asks the server to send its public key
2. server sends its public key
3. users creates a symmetric key, encrypts it using server's public key, and sends the encrypted symmetric key to the server
4. user encrypts its password with the symmetric key and sends it to the server
5. server decrypts the received message to retrieve the password and authenticates the user
6. user and server communicate using the symmetric key for the rest of the session

Note: Step 1 and 2 are executed only the very first time the user uses ssh

SSH is resilient to eavesdropping attack, because all communication is encrypted using the symmetric key. It is resilient to the replay attack, because a different symmetric key is used to encrypt password and other info in different login sessions. However, SSH is vulnerable to man-in-the-middle attack in steps 1 and 2 (only the very first time).

1 point for identifying steps 1 and 2 executed only the very first time
2 points for correctly providing steps 3, 4, 5 and 6
1 point for identifying that SSH is resilient to eavesdropping and replay attacks
1 point for identifying that SSH is vulnerable to man-in-the-middle attack

2. Suppose there are four separate partitions in RAM at present: P1: 100K, P2: 500K, P3: 200K, P4: 300K, and P5: 600K (in order). In what partitions would each of the following algorithms place processes of 212K, 417K, 112K, and 426K (in order)?

(a) First fit
212:K → P2; 417K → P5; 112K → P2; 426K cannot be allocated
(b) best fit
212:K → P4; 417K → P2; 112K → P3; 426K → P5
(c) next fit
212:K → P2; 417K → P5; 112K → P5; 426K cannot be allocated
(d) worst fit
212:K → P5; 417K → P2; 112K → P5; 426K cannot be allocated

1.25 Points for each part
-0.5 points if one allocation is incorrect per part

3. What is the working set principle? Explain how it prevents thrashing?

Working set principle is that a process is allowed to run only if its entire working set is loaded in memory.

Working set of a process is the set of pages a process has accessed in the last X units of time, where X is dynamically adjusted to keep the number of page faults (f) of the process between a low and a high threshold. If f goes below the low threshold, X is reduced and if f goes above the high threshold, X is increased.

Thrashing occurs when the CPU spends most of its time handling page faults by swapping pages between memory and disk. By keeping the number of page faults of each process between a low and a high threshold, it is ensured the CPU utilization, i.e. when CPU is doing useful work and not handling page faults, remains high.

1 Points for defining the working set principle correctly
2 Points for explaining what a working set is and how it is constructed
2 Points for explaining thrashing and it is prevented

4. If a disk read takes 10 ms and a memory read takes 10 ns, what would be the average access time if the probability of a page fault is 0.001?

Access time = p * 10 ms + (1-p) * 10 ns, where p is the probability of page faults
= 0.001 * 10 ms + 0.999 * 10 ns
= 10.00999 micro seconds
2 Points for the correct formula
2 Points for getting the correct answer (an answer of 10 micro-seconds is ok)
1 Points for using the correct time units

5. Explain how the enhanced second chance algorithm works?
For each page, 2 bits (reference bit and modify bits) are used. Reference bit (R) of a page is set when that page is referenced, and modify bit (M) is set when that page is modified. In addition, reference bits of all pages are cleared periodically.

Pages are organized in four classes: Class 0: R = 0, M = 0; Class 1: R = 0, M = 1; Class 2: R = 1, M = 0; Class 3: R= 1, M = 1.

All pages are arranged in a logical circle and a clock hand points to one of the pages. When a page fault occurs, the clock hand is rotated until a page in the lowest non-empty class is encountered. That page is then replaced.

2 Points for identifying the two bits and explaining how those bits are updated
2 Points for correctly identifying the four classes
1 Point for explaining the algorithm

6. The following code is executed by 10 different processes that all share the integer variable *counter* whose initial value is 5. Explain through an example how there is a possibility of race condition in this code.

   counter++;

   Assembly code:
   (1) reg1 = counter
   (2) reg1 = reg1 + 1
   (3) counter = reg1

   Process 1 executes (1) and (2) and is preempted
   Process 2 executes (1) (2) and (3) and exits
   Process 1 is re-scheduled and executes (3)

   counter is increased by 1 while two processes have incremented it

   2 Points for correctly getting the assembly code
   2 Points for getting the sequence of steps correct
   1 point for identifying the discrepancy

Problems

1. [9 Points] A disk has 400 cylinders, 0 – 399. Disk access requests arrive in the following order of cylinder numbers: 212 122 190 300 0 41 300 10. Assume that the R/W head is on cylinder number 195 initially moving towards lower numbered cylinders. Calculate the total distance (Number of cylinders) travelled by the R/W head to service these requests if

(a) FCFS algorithm is used

195 → 212 → 122 → 190 → 300 → 0 → 41 → 300 → 10
distance = 17 + 90 + 68 + 120 + 300 + 41 + 259 + 290 = 1185

(b) SSTF algorithm is used

195 → 190 → 212 → 300 → 300 → 122 → 41 → 10 → 0
distance = 5 + 22 + 88 + 0 + 178 + 81 + 31 + 10 = 415

(c) SCAN algorithm is used

195 → 190 → 122 → 41 → 10 → 0 → 212 → 300 → 300
distance = 5 + 68 + 81 + 31 + 10 + 212 + 88 + 0 = 495

For each part:

2 Points for getting the correct sequence (300 must appear twice)
1 Point for getting the right formula for the distance (it is not necessary to calculate the final sum)

2. [9 Points] Assume that you have a primary memory consisting of 4 page frames. Initially, this memory is empty. Consider a page reference string 5, 3, 5, 1, 2, 1, 7, 5, 3, 1. For each of the following page replacement algorithm (a) show the pages loaded in memory after each page reference, and (2) calculate the number of page faults.

(a) Optimal

Frame	5	3	5	1	2	1	7	5	3	1
0	5	5	5	5	5	5	5	5	5	5
1		3	3	3	3	3	3	3	3	3
2				1	1	1	1	1	1	1
3					2	2	7	7	7	7
	*	*		*	*		*			

Five page faults

(b) Least Recently Used

Frame	5	3	5	1	2	1	7	5	3	1
0	5	5	5	5	5	5	5	5	5	5
1		3	3	3	3	3	7	7	7	7
2				1	1	1	1	1	1	1
3					2	2	2	2	3	3
	*	*		*	*		*		*	

Six page faults

(c) FIFO

Frame	5	3	5	1	2	1	7	5	3	1
0	5	5	5	5	5	5	7	7	7	7
1		3	3	3	3	3	3	5	5	5
2				1	1	1	1	1	3	3
3					2	2	2	2	2	1
	*	*		*	*		*	*	*	*

Eight page faults

For each part:
2 Points for the pages loaded in memory after each page reference
1 Point for identifying all page faults correctly

3. Assume that a system has 512 KB of RAM for user processes with frame size of 2 KB. A process consists of 16 pages, and the page mapping table and TLB are as shown below:

Page Table:

Frame #	**Valid bit**	**R/W bit**	**Modify bit**
20	1	0	1
7	1	0	0
16	0	0	0
100	1	1	1
4	0	1	1
64	1	0	1
151	0	1	1
11	1	1	0
55	1	0	0
7	0	0	1
32	0	1	0
77	1	0	1
4	1	1	0
25	1	1	0
0	1	0	0
1	1	0	0

TLB:

Page #	Frame #
7	11
14	0
8	55
13	25
0	20

(a) **[2 Points]** What are the maximum and minimum numbers of frames this process needs to run? Ignore the space needed to store process state or page table.

Maximum number of frames: ______16____

Minimum number of frames: _______1______

1 Point for each part

(b) **[6 Points]** The following virtual addresses are referenced. Compute the corresponding physical addresses. Write these addresses in binary format.

A1: 100010110110011 Physical address = 110111 10110110011 (page#: 8)

A2: 1111110101000 Physical address = 1 11110101000 (page#: 3)

A3: 1000111100110 Physical address = (page#: 2) Page fault

A4: 10000001111000 Physical address = (page#: 4) Page fault

A5: 110111000111010 Physical address = 11001 11000111010 (page#: 13)

A6: 100111000011010 Physical address = (page#: 9) Page fault

(c) **[4 Points]** Assuming that there are no other user processes running, provide a ranking of the six addresses in (b) from fastest to slowest in terms of total address translation time. Explain your rankings.

Fastest to slowest: page# in TLB, page# not in TLB and no page fault, page fault
Since this is the only process running, no page has to be replaced to handle page faults, so cost of all page faults is same (loading the new page in).

Fastest: A1, A5
Medium: A2
Slowest: A3, A4 and A6

2 Points for getting the rankings correct
2 Points for correct explanation

(d) **[2 Points]** How does your answer to (c) might change if there are several other user processes running in the system at the same time? You do not need to provide any new rankings here.

With more processes running, a page fault may require some of the pages in memory to be swapped to the disk. This time can vary depending on whether the victim page has been modified or not. So, the rankings in slowest category, A3, A4 and A6 will change with respect to one another. They will still be slower than the fastest and medium category though.

1 Point for identifying that rankings only in the slowest category will change
1 Point for correct explanation

4. You are asked to allocate a file according to either a File Allocation Table (FAT) or multi-level indexed allocation (UNIX inode - triply indirect). Assume that the file is 132 MB long, there are 2 KB per disk block, each pointer in FAT occupies 4 bytes, and each index block entry takes 4 bytes.

(a) **[2 Points]** How many bytes are used to lay out the file when using a FAT file system?

Total blocks needed = 132 MB/2 KB = 66 * 2^{10} = 67584

FAT: Need to allocate 66 * 2^{10} pointers for this file = 66 * 2^{12} bytes = 270336 bytes
However, note that all these pointers, except the pointer to the first block are stored in the file allocation table, which is always allocated. So the extra space overhead for a file is just one pointer, i.e. 4 bytes

0.5 Point for calculating the total number of blocks
1 Point for correct formula plus calculation
0.5 Points for making the observation about the extra space overhead

(b) **[2 Points]** How many bytes are used to lay out the file when using a UNIX-style file system?
UNIX: First 12 blocks – 12 direct pointers = 48 bytes
Next 512 (29) blocks – single indirect pointer = 4 + 2^{11} bytes
Next 67060 blocks – double indirect pointer = 4 + 2^{11} + 131 * 2^{11} bytes
0 blocks – triple indirect pointer = 4 bytes
Total: 60 + 133 * 2^{11} bytes = 272,444 bytes

1 Point for identifying direct blocks, single, double and triple indirect blocks
1 Point for correct calculation

Now suppose that you wish to read the 66,000'th block of the file. Assume that each of the following counts as one search operation: moving from one element to the next in a linked list; indexing into an index block; moving from index block to the next.

(c) **[2 Points]** How many searches are needed to read block 66,000 when using the FAT file system?

Need to traverse 66,000 pointers in the linked list. So, 66,000 searches in total.

(d). **[2 Points]** How many searches are needed to read block 66,000 when using the UNIX-style file system?

This block is in the doubly indirect index.

Total searches = 1 into FCB + 1 to move to first level index block + 1 to index into the first level index block + 1 to move to second level index block + 1 to index into the second level index block = 5 searches in total

1 Point for identifying that the block is in doubly indirect index
1 Point for correct calculation