University of California, Berkeley
College of Engineering
Computer Science Division – EECS

Spring 2016 Anthony D. Joseph

Midterm Exam #2 *Solutions*

April 20, 2016
CS162 Operating Systems

Your Name:	
SID AND 162 Login:	
TA Name:	
Discussion Section Time:	

General Information:
This is a **closed book and one 2-sided handwritten note** examination. You have 80 minutes to answer as many questions as possible. The number in parentheses at the beginning of each question indicates the number of points for that question. You should read **all** of the questions before starting the exam, as some of the questions are substantially more time consuming.

Write all of your answers directly on this paper. ***Make your answers as concise as possible.*** If there is something in a question that you believe is open to interpretation, then please ask us about it!

Good Luck!!

QUESTION	POINTS ASSIGNED	POINTS OBTAINED
1	12	
2	26	
3	24	
4	22	
5	16	
TOTAL	100	

Solutions **NAME: ____**

1. (12 points total) True/False and Why? **CIRCLE YOUR ANSWER AND WRITE AN EXPLANATION (no credit will be given if no explanation is provided).**

a. In addition to greatly reducing the overall failure probability, moving from a single hard drive to using RAID 1 (mirroring) can also increase the throughput of small random reads.

TRUE FALSE

Why?

TRUE*. The small reads need only be sent to a single disk, so in theory, a RAID-1 system can double small random read throughput. The correct answer was worth 2 points and the justification was worth an additional 2 points.*

~~b. The TCP transport protocol provides a reliable, in-order bytestream abstraction.~~

TRUE FALSE

Why?

TRUE*. These are exactly the characteristics of TCP. The correct answer was worth 1 point and the justification was worth an additional 2 points.*

c. The Byzantine failure model allows arbitrary behavior by a faulty system component.

TRUE FALSE

Why?

TRUE*. That is the definition of the model. The correct answer was worth 2 point and the justification was worth an additional 2 points.*

d. Remote Procedure Calls provide identical semantics to local calls.

TRUE FALSE

Why?

FALSE*. RPCs can suffer from network/server failures, so the users need to deal with this difference. The correct answer was worth 2 points and the justification was worth an additional 2 points.*

2. (26 points total) Memory and I/O.

a. (12 points) For each statement, **CIRCLE** the address translation scheme(s) for which the statement is true.

i) (4 points) A process cannot directly read or write another process' memory. Assume there is no `mmap()` operation.

No translation

Base and Bound

Segmentation

Paging

Base and Bound, Segmentation, *and* ***Paging****.*

ii) (4 points) External fragmentation may occur.

No translation

Base and Bound

Segmentation

Paging

Base and Bound *and* ***Segmentation****.*

iii) (4 points) Part of the process' stack could be swapped out to disk while the process continues to execute using the rest of the stack. *Assume the stack is located in a typical location for the type of address translation in use.*

No translation

Base and Bound

Segmentation

Paging

Paging

b. (10 points) Consider a system with the following specifications:

- 46-bit virtual address space
- Page size of 8 KBytes
- Page table entry size of 4 Bytes
- Every page table is required to fit into a single page

How many levels of page tables would be required to map the entire virtual address space?

In the space below, document the format of a virtual address under this translation scheme. Briefly explain your rationale.

*A 1-page page table contains 2,048 or 2^{11} PTEs ($2^2 * 2^{11} = 2^{13}$ bytes), pointing to 2^{11} pages (addressing a total of $2^{11} * 2^{13} = 2^{24}$ bytes). Adding a second level yields another 2^{11} pages of page tables, addressing $2^{11} * 2^{11} * 2^{13} = 2^{35}$ bytes. Adding a third level yields another 2^{11} pages of page tables, addressing $2^{11} * 2^{11} * 2^{11} * 2^{13} = 2^{46}$ bytes. So, we need **3 levels**.*

The correct answer is worth 3 pts. Correct reasoning is worth up to 7 pts (2 pts for identifying that there are 2^{11} PTEs per page, 2 pts for describing how page tables are nested, and 3 pts based upon the quality of the argument):

11 bit page	*11 bit page*	*11 bit page*	*13 bit offset*

c. (4 points) Briefly explain, in four sentences or less, what is DMA ***and*** why DMA enables a system to more efficiently handle I/O devices.

With DMA, the CPU does not have to be involved in the transfer of each byte or word. It is only involved in the set up of the transfer.

Solutions **NAME:** ________________________________

3. (24 points total) Demand Paging.
   a. (18 points) Consider a demand paging system with 3 pages of physical memory. When a page fault occurs, we use a *page replacement algorithm* to make space for the new page by evicting a page from memory. Using the FIFO, Clock Algorithm, and LRU page replacement algorithms, *mark which pages will be in physical memory **after** each page request*. The first 3 rows for each algorithm are pre-filled for you. For the clock algorithm, on a page fault the clock hand advances before performing any checks or actions.

Time	**FIFO**				**Clock**				**LRU**			
– – –	**A**	**B**	**C**	**D**	**A**	**B**	**C**	**D**	**A**	**B**	**C**	**D**
A	X				X				X			
B	X	X			X	X			X	X		
C	X	X	X		X	X	X		X	X	X	
D		*X*	*X*	*X*		*X*	*X*	*X*		*X*	*X*	*X*
B		*X*	*X*	*X*		*X*	*X*	*X*		*X*	*X*	*X*
A	*X*		*X*	*X*	*X*	*X*		*X*	*X*	*X*		*X*
B	*X*	*X*		*X*	*X*	*X*		*X*	*X*	*X*		*X*
A	*X*	*X*		*X*	*X*	*X*		*X*	*X*	*X*		*X*
D	*X*	*X*		*X*	*X*	*X*		*X*	*X*	*X*		*X*
C	*X*	*X*	*X*		*X*	*X*	*X*		*X*		*X*	*X*
Number of page faults	*7*				*6*				*6*			

b. (4 points) When a page fault occurs and a page is evicted, what entries in which data structures must the OS invalidate?

When a page fault occurs, the OS must invalidate the TLB entry and page table entry corresponding to the evicted page.

c. (2 points) In the Nth chance page replacement algorithm, explain the purpose of the parameter *N*. Consider the effects of a small or large *N*.
N enables the OS to approximate the behavior of an LRU algorithm by tracking approximately how long a page it is since a page was last referenced. A larger value of N will enable a more finer grain approximation of LRU.

4. (22 points total) Disks and File Systems.

   a. (6 points) Name two performance-improving features that are provided by modern disk controllers. For each of the features you name, briefly explain the benefits and consequences of a disk controller that does not provide that feature.

   i) Feature #1 – Benefits and consequences if it is not provided:

   We decided to grade this question more leniently by interpreting 'performance' to be techniques/features that improve the performance of disk operations OR reduce the complexity of the OS. Note that we did not accept SMR as an answer because it somewhat reduces the performance of reads (due to the high degree of ECC required) and it significantly reduces the performance of writes (due to the need to write an entire shingled region when writing a single block).

   - *Track buffer – Reads an entire track when a sector is requested. Pro: makes sequential reads faster. Con: without it OS has to implement read ahead. Note that we did not accept answers that confused the OS buffer cache with a disk track buffer.*
   - *TRIM for SSD. Pro: improves performance of writes. Con: slow writes due to need to erase before writing*
   - *ECC for HDDs. Pro: can recover data from failing sector. Con: data loss*
   - *Disk scheduling. Pro: higher performance due to reduced seek/rotational latencies. Con: more seek/rotational latency overhead*
   - *DMA support. Pro: higher performance due to reduced CPU overhead. Con: higher CPU overhead.*
   - *Sector interleaving/spanning or track skewing. Pro: higher performance for sequential access due to reduced rotational latencies. Con: higher rotational latencies.*
   - *Remapping of defective sectors/slip sparing. Pro: enables Logical Block Addressing so the OS can view the disk as a linear sequence of blocks. Con: without it OS, would have to manually keep track of failed sectors and the locations of alternate sectors.*
   - *Accelerometer for protection against drops. Pro: helps reduce damage by moving head to spare track upon detection of a fall. Con: Damage to HDD.*
   - *RAID. Pro: helps protect against disk failure. Con: disk failure causes data loss*
   - *Disk buffers for writes. Pro: faster writes. Con: slower writes*

   ii) Feature #2 – Benefits and consequences if it is not provided:

   See above.

b. (5 points) If a disk subsystem receives an average of 10 requests per second and each request has an average response time of 500ms, how many requests are in the subsystem on average? *Explain your answer.*

5. We can apply Little's Law: $L = \lambda W$, where λ is the average arrival rate (10 requests per second) and W is the average response time (1/2 a second)

c. (4 points) The FAT file system eliminates external fragmentation by allocating disk space in block-sized units. Considering that fact, briefly explain why defragmentation is still sometimes necessary.

The disk blocks allocated to a file may be scattered across the disk.

d. (3 points) In the UNIX 4.2 BSD FFS, inodes have direct pointers, a singly indirect pointer, a doubly indirect pointer, and a triply indirect pointer. The maximum file size supported by this inode type is approximately the same as the maximum file size supported by an inode with only a triply indirect pointer. Briefly explain one disadvantage of an inode design that only uses a triply indirect pointer instead of a combination of pointers (*other than the slightly reduced maximum file size*).

Every file access would require four additional disk reads.

e. (4 points) Briefly, *in one to three sentences*, explain the purpose of the journal in a journaling file system?

The journal is used to store the intentions of the OS when making changes to the file system structures (and the data in a full journaling file system). During boot, the journal is used to redo(or complete) any incomplete operations (operations that were in-progress at the time of an OS crash or power failure).

5. (16 points total) Pintos coding question: Verifying user pointers.
Your project group is tasked with implementing a new syscall for Pintos. This syscall takes a linked list as one of its arguments, and your responsibility is to create a function that checks the validity of the linked list, so the syscall handler can safely dereference it. The value of head may be NULL, if the list is empty. This is function signature of your syscall:

```
void foo(struct node *head);
```

We have also defined the structure of the `node` struct, as well as `exit_thread_if_invalid()`, a function that will verify the validity of user-specified pointers.

```
struct node
  {
    int number;
    char buffer[16];
    struct node *next;
    // If this is the last node, next = NULL
  };

/* Checks that P to P+SIZE-1 is a valid user buffer.
   Kills the current thread if it is invalid. */
void exit_thread_if_invalid(void *p, size_t size);

uint32_t SYS_FOO = 60;  // The syscall number for foo()
```

Fill in the `syscall_handler()` function ***on the next page*** so that it *safely* handles the `SYS_FOO` syscall.

```
static void
syscall_handler (struct intr_frame *f)
{
  uint32_t* args = ((uint32_t*) f->esp);
  exit_thread_if_invalid(args, 4);





  // Check if this is SYS_FOO
  if (_args[0] == SYS_FOO_____________________) {
    exit_thread_if_invalid(&args[1], 4);
    struct node *head = args[1];
    while (head != NULL) {
      exit_thread_if_invalid(head,
                             sizeof(struct node));
      head = head->next;
    // This will actually handle SYS_FOO,
    //   assuming arguments are valid
    handle_foo(args, &f->eax);
  } else {
    // Code for ALL other syscalls will go here.
  }
}
```