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

Spring 2018 Anthony D. Joseph and Jonathan Ragan-Kelley

Midterm Exam #3 *Solutions*

April 25, 2018
CS162 Operating Systems

Your Name:	*Mitsuha Miyamizu*
SID AND 162 Login:	*13371337, s042*
TA Name:	*Taki Tachibana*
Discussion Section Time:	*4-5pm Funday*

General Information:
This is a **closed book and TWO 2-sided handwritten notes** examination. You have 110 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	36	
2	22	
3	22	
4	20	
TOTAL	100	

NAME: ____

1. (30 points total) Short answer

a. (12 points) True/False and Why? **CHECK THE BOX FOR YOUR ANSWER.**

i) Pintos uses separate user-kernel mode transfer implementations for interrupts and syscalls.

❑ TRUE ❑ FALSE

Why?
False. *Syscalls in Pintos are registered as an interrupt handler (0x30 to be exact)*
The correct answer was worth 2 point and the justification was worth an additional 1 point.

ii) One only needs a source & destination IP address, source & destination port to uniquely identify a network connection.

❑ TRUE ❑ FALSE

Why?
False. *You need the transport protocol to differentiate between multiple connections between two hosts (IP addr), as you could have multiple connections with different transport protocols between two ports. The correct answer was worth 2 point and the justification was worth an additional 1 point.*

iii) The Two Phase Commit protocol guarantees that all the participants commit or abort at the same time.

❑ TRUE ❑ FALSE

Why?
False. *2PC only guarantees that all the participants commit or abort but does not guarantee the timings. The correct answer was worth 2 point and the justification was worth an additional 1 point.*

iv) Applying a conservative view of the End-to-End Principle to reliable file transfer means you should only implement reliability functionality at the endpoints.

❑ TRUE ❑ FALSE

Why?
True. *The E2E argument says that you need to verify correct delivery at the endpoints. Implementing the functionality in the network is insufficient. The correct answer was worth 2 point and the justification was worth an additional 1 point.*

NAME: ____

b. (14 points) Consider the following 6 I/O operations and their respective cylinder locations on disk. Seek time is 0.1milliseconds *per cylinder traversed*.

Operation	A	B	C	D	E	F
Cylinder	4	10	35	62	69	95

i) Use the Shortest Seek Time First disk scheduling algorithm to schedule these operations. The arm begins at cylinder 33. List the order in which the 6 operations are scheduled and calculate the total seek time. *Ignore rotational and transfer delays.*

Letter:

________, ________,________,________,________,________

C (0.2), B (2.5), A (0.6), D (5.8), E (0.7), F (2.6)

Total seek time: __________

12.4ms

ii) Unhappy with this seek time, we decide to use SCAN as our disk scheduling algorithm instead. Assume SCAN begins by traversing descending cylinder numbers starting with the disk arm at cylinder 33. List the order in which our 6 operations are scheduled and calculate the total time we spend seeking. You may ignore rotational delay in your calculations.

Letter:

________, ________,________,________,________,________

B (2.3ms), A (0.6ms), C (3.1ms), D (2.7ms), E (0.7ms), F(2.6ms)

Total seek time: __________

12.0ms

c. (3 points) Journaling is a technique that can be used to ensure that a file system recovers its metadata to a consistent state after a crash. One type of journaling is called "**data journaling**," because all file system metadata and data is written to the log before being updated in place. Describe the performance of such a journaling file system, as compared to a file system that only logs metadata, for *very large* sequential write workloads. *Roughly* what throughput can you expect and why?

All data gets written twice!
For large sequential writes, most traffic is data
Half the throughput or 2x slower performance

NAME: ____

d. (4 points) What do the four letters in ACID stand for?

A____

C____

I____

D____

Atomicity, Consistency, Isolation, Durability

e. (3 points) Briefly explain how from a programmer's prospective, using RPC differs from using procedure calls.

Procedure calls have shared fate, while for RPC, the remote or local machine may independently fail. RPC is more expensive timewise than local procedure calls.

NAME: ____

2. (22 points total) Revisiting Homework #2.

Your new startup lets users watch streaming anime videos, so you reuse the code you wrote for homework #2 to create a simple video server that accepts client connections, reads in the video title string (e.g. "Kimi no Na wa."), and writes back the raw HD video file to the client.

- Ignore HTTP formatting and headers – assume we read & write raw strings.
- Make sure your code is multithreaded and allows handling of multiple simultaneous client connections.
- Assume very large video files (possibly on the order of gigabytes or more).
- Do not forget to clean up and not leak resources, including freeing memory and closing file descriptors when appropriate.

a) (16 points) Fill in the below code, including all blank lines and freeform boxes. Assume for this part that system calls will succeed when possible (i.e. you do not have to include, for example, checks for thread creation or malloc success). Also assume that the client connection does not drop or hang up prematurely.

```
#define MAX_LEN 20
#define BUFFER_SIZE 4096

void *serve_video(void *);

int main() {
  pthread_t thread; struct sockaddr serv_addr,
                               cli_addr; int cli_len;
   // sockaddr initialization code omitted
   int sockfd = socket(PF_INET, SOCK_STREAM, 0);
   bind(sockfd, (struct sockaddr *) &serv_addr,
                                   sizeof(serv_addr));
   Listen(sockfd, 5);
   while (1) {
        int newsockfd = accept(sockfd, (struct
                    sockaddr *) &cli_addr, &cli_len);
        pthread_create(&thread, NULL, serve_video,
(void*)newsockfd);
   }
   close(sockfd);
   return 0;
}
```

NAME: ______________________________

```
void *serve_video(void *arg) {
  int newsockfd = (int) arg;
  // expect both "request" and "path" < 20 chars
  char request[MAX_LEN];
  char path[MAX_LEN];
  int bytes_read = read(newsockfd, request, MAX_LEN);
  int success = request_to_video_path(request, path);
  // request_to_video_path(): takes "request" string,
  // populates "path" string with file path to video
  if (success == -1) { //request not found or invalid
     close(newsockfd);
     return NULL;
  }
  int fp = open(path, O_RDONLY);
  char* buffer = malloc(BUFFER_SIZE*sizeof(char));
  /* Fill in the following box */

     bytes_read = read(fp, buffer, BUFFER_SIZE);
     while (bytes_read > 0) {
        int bytes_written = 0;
        while (bytes_written < bytes_read)
           bytes_written += write(newsockfd, buffer +
     bytes_written, bytes_read – bytes_written)
        bytes_read = read(fp, buffer, BUFFER_SIZE);
     }
     free(buffer);
     close(fp);
     close(newsockfd);

  return NULL;
}
```

NAME: ______________________________________

b) (3 points) Assume you coded part a) correctly. Now suppose that malloc() failed in one the threads running `serve_video()`. Briefly, in 1-2 sentences, explain what would happen and why.

The entire process would crash, including the server and the other handlers, because threads share address space. (actually it wouldn't b/c read just returns -1 and sets EFAULT and so nothing would happen, but we didn't expect you to know this)

c) (3 points) Suppose you forgot to close any of your file descriptors in the above code and you run it indefinitely. Briefly, in 1-2 sentences, explain what could happen to your video server and why.

`accept()` would fail when you hit the file descriptor limit.

NAME: ____

3. (22 points total) Filesystems.

The Berkeley Fast File System (FFS) in BSD 4.2 was one of the first file systems that treated the disk like a disk and thus improved performance through spatial locality

a. (3 points) What on-disk structures does FFS use to track allocation of inodes and data blocks?

Bitmaps or Bitvectors

b. (3 points) Why was the innovation in (3a) an improvement over the prior Unix file system in BSD 4.1?

BSD 4.1 used linked lists which caused fragmentation and made contiguous allocation difficult.

c. (4 points) Recall that FFS tries to spread large files across disk by splitting them into chunks and putting each chunk in a different part of a block group. Given a hard drive with a peak transfer rate of 200Megabytes/s and that seek and rotation combined take on average 16 milliseconds. How big should each chunk of the file be so that we can achieve 2/3 of the peak transfer rate for large files when they are accessed sequentially? *Show work outside of the box for partial credit.*

6.4 MBytes

*16 ms + 2x/3 = x, so x = 48ms. Transfer size is thus 2/3 * 48 * 10^-3 * 200 * 10^-6 = 6.4 MBytes*

d. (12 points) List the set of disk blocks that must be read into memory in order to read the file `/cs162/bar/foo.txt` in its entirely from a UNIX BSD 4.2 filesystem. Assume that the file is 12,300 Bytes long and that the disk blocks are 1 Kilobytes long. You can assume that each directory fits into a single disk block.

Number of Blocks:

21 blocks

Blocks:

1. Read in the file header for root (always at a fixed spot on disk)
2. Read in the first data block for root (/)
3. Read in file header for cs162
4. Read in first data block for cs162
5. Read in file header for bar
6. Read in first data block for bar
7. Read in file header for foo.txt
8. Read in first data block for foo.txt
9 - 19. Read in second through 12th data blocks for foo.txt
20. Read in indirect block pointed to by the 11th entry in foo.txt
21. Read in 13th foo.txt data blocks. The 13th block is partially full

NAME: ____

4. (20 points total) Security and Two-Phase Commit.

Suppose that No Such Agency headquarters (coordinator) in **Washington** (W) communicates with branch offices (workers) in **Moscow** (M) and **Beijing** (B) using symmetric-key-encrypted, reliable messages. Suppose furthermore that the encryption key used is changed each day; every evening the new encryption key that is to be used the next day is sent from Washington to the two branch offices. Two-Phase Commit is used in order to ensure that, despite crashes, either (a) everyone eventually switches to the new encryption key, NEWKEY, or (b) no one switches to the new encryption key and everyone continues using OLDKEY. Assume that a machine takes a *long* time to reboot and recover after a crash, and timeouts are 1 second.

The steps involved in implementing two-phased commit are listed below, in time order:

1. Washington: write "begin transaction" to its log
2. Washington → Moscow: "New key is NEWKEY." (→ means W sends msg to M)
3. M: write "New key is NEWKEY." to its log
4. M → W: "Prepared to commit"
5. W → B: "New key is NEWKEY."
6. B: write "New key is NEWKEY." to its log
7. B → W: "Prepared to commit"
8. W: write "New key is NEWKEY." to its log
9. W: write "commit" to its log
10. W → M: "commit"
11. M: write "got commit" to its log
12. M: Key = NEWKEY
13. M → W: "ok"
14. W → B: "commit"
15. B: write "got commit" to its log
16. B: Key = NEWKEY
17. B → W: "ok"
18. W: Key = NEWKEY
19. W: write "done" to its log

a. (4 points) If W crashes after step 9 and no one else fails, what key will everyone end up using, once W reboots and recovers? ***Give the reason why.***

Key: ________________
NEWKEY

Why:
Step 9 commits the transaction.

NAME: ____

b. (4 points) If M crashes after step 4 and no one else fails, what key will everyone end up using, once M reboots and recovers? ***Give the reason why.***

Key: __________________
NEWKEY

Why:
M has informed W that it is prepared to commit (or abort). W continues and commits the transaction in step 9. When it tries to tell M to commit in step 10, it will notice that M has crashed and will have to wait for M to recover.

c. (4 points) If B crashes after step 4 and no one else fails, what key will everyone end up using, once B reboots and recovers? ***Give the reason why.***

Key: __________________
OLDKEY

Why:
W will time out on trying to communicate with B and will abort the transaction.
Note*: Since message communications take on the order of milliseconds to execute and crashes take* ***much*** *longer to recover from (think of how long it takes to reboot your laptop), it's not reasonable to assume that B will recover before step 5, so that W will continue as if nothing had happened. Therefore W will always notice that B crashed and will abort the transaction.*

NAME: ________________________________

d. (4 points) If M crashes after step 10, what recovery steps must it take after it reboots in order to achieve the correct global state with respect to which encryption key to use?

M will look at its log and see that there is an entry for a transaction for which it is "prepared to commit" but hasn't heard a final resolution of commit or abort yet. It sends a message to W asking whether to commit or abort the transaction. W will tell it to commit the transaction, implying that M will do steps 11, 12, and 13, and will end up using the new key.

e. (4 points) If W crashes after step 11, what recovery steps must it take after it reboots in order to achieve the correct global state with respect to which encryption key to use?

W will look at its log and see that there is an entry for a transaction that it has committed but not finished. (If it were finished, the "front" pointer for the log would have been moved past it.) W will contact both M and B and tell them to commit the transaction and then finish doing the transaction itself. It has to tell both M and B because it doesn't whom it told to commit before it crashed -- it only knows that it wrote "commit" to its log. Put another way, W "restarts" the transaction at step 10 since the only thing it is sure of during recovery is that it executed step 9, which wrote "commit" to the log.

NAME: ____

5. (0 points total) That's All, Folks!

   a. What does the "r" in "jrk" stand for? (You should really know this one)

   b. Favorite part of CS162?

   c. Any other thoughts on how CS162 staff is doing or to improve the course?

Hope you enjoyed CS162 and operating systems, have a fun summer!