CS 111 Final exam

Ashwin Vivekanandh

TOTAL POINTS

137 / 150

QUESTION 1

1 Scatter/gather I/O 10 / 10

✓ - **0 pts** Correct

- **10 pts** No answer
- **3 pts** Not identifying DMA
- **3 pts** Not identifying non-contiguity of virtual RAM pages
- **2 pts** not identifying data copying as main issue
- **2 pts** Memory mapped I/O is not a motivation
- **2 pts** Not about accumulating I/O operations.
- **2 pts** Files and inodes not relevant.
- **10 pts** Totally wrong
- **2 pts** Scattering and gathering is over RAM, not I/O device.
- **2 pts** Not related to TLB misses.
- **1 pts** Segments are not necessarily contiguous in physical memory, either.
- **2 pts** Memory mapped I/O != paged virtual memory
- **1 pts** Which mechanisms of a VM system?
- **8 pts** DMA and the paging aspect of VM lead to problems without scatter/gather.
- **2 pts** File system issues irrelevant.
- **4 pts** Scatter/gather typically unrelated to demand paging.
- **2 pts** DMA requires physically contiguous memory.
- **3 pts** Defragmentation has nothing to do with scatter/gather.
- **2 pts** Swapping not relevant.
- **2 pts** Double buffering is irrelevant.
- **3 pts** Poor explanation.
- **2 pts** Fragmentation is not directly related to this issue.
- **9 pts** One tiny bit of correct information
- **1 pts** Internal device memory not relevant.

QUESTION 2

2 Metadata journaling 10 / 10

✓ - **0 pts** Correct

- **10 pts** No answer
- **3 pts** Didn't provide enough discussion about what could happen if we write data blocks after metadata/journal is modified.
- **7 pts** Not very correct.

QUESTION 3

3 URLs and links 10 / 10

✓ - **0 pts** Correct

- **10 pts** No answer
- **4 pts** A URL is more like a soft (symbolic) link
- **3 pts** In both cases, the link is a name describing a traversal through a set of linked data items - files and directories in the case of a soft
link, web pages in the case of a URL.
- **3 pts** There is no guarantee in either case that the data
item named by the URL or soft link actually exists.
- **10 pts** wrong answer
- **1 pts** mixed the concept of domain and URL
- **1 pts** do not explain how a URL works

QUESTION 4

4 Password salting 9 / 10

- **0 pts** Correct
- **10 pts** No answer
- **3 pts** Did not correctly explain in detail the definition of salt
- **4 pts** Did not correctly discuss in detail preserving password secrecy in the context of hashes

✓ - **3 pts** **Did not correctly explain dictionary attacks / brute force attacks**

+ **2** **Point adjustment**

QUESTION 5

5 Factors 10 / 10

✓ **- 0 pts** Correct

- 10 pts No answer

- 5 pts A factor is an aspect of the system that you intentionally alter in controlled ways during the evaluation.

- 5 pts Proper choice of factors will allow the experimenter to gain insight into the likely performance outcome of design choices and varying use cases

- 1 pts The reason is not clearly or correctly explained

- 10 pts wrong answer

- 2 pts not proper answer "why"

- 3 pts It's the variables we alter

QUESTION 6

6 File descriptors and capabilities 9 / 10

- 0 pts Correct

- 10 pts No answer

✓ **- 1 pts OS can easily revoke a file descriptor by removing it from the process control block.**

- 3 pts Uniqueness not really a property of either capabilities or file descriptors. Important point is that possession grants access.

- 2 pts Important point is mere possession of each grants access.

- 2 pts Capabilities do not necessarily have any "position" information associated.

- 1 pts Users can also access files by opening them via ACL, so FDs alone don't specify their possible available files.

- 7 pts Both capabilities and file descriptors are about access control, not identification and/or authentication.

- 2 pts Changing the ACL does not invalidate existing file descriptors.

- 2 pts File descriptors are R/W specific.

- 3 pts File descriptors tell us nothing about why someone could access a file, merely that they can.

- 8 pts Insufficient detail.

- 5 pts Important point is that both are access control mechanisms providing security based on mere possession of a data structure.

- 1 pts Capabilities usually do not contain a list. Rather, you have a list of capabilities.

- 7 pts How is a FD like a capability?

- 5 pts Misdefinition of capabilities.

QUESTION 7

7 Dining philosophers 10 / 10

✓ **- 0 pts** Correct

- 10 pts No answer

- 9 pts Wrong answer.

- 3 pts Needs a better explanation. A good example is when all philosophers call getforks() at the same time and all of them get the left fork.

- 3 pts Partial correct.

QUESTION 8

8 Monitors and synchronized methods 8 / 10

- 0 pts Correct

- 10 pts No answer

- 4 pts More detail on granularity.

✓ **- 2 pts All synchronized methods in an object share one lock.**

- 2 pts OO monitors provided by language, not OS.

- 6 pts Monitors lock entire object for any method, synchronized methods only lock on specified methods.

- 6 pts Sync methods more fine grained than object monitors, since the latter locks object on ANY method.

- 10 pts Totally wrong.

- 3 pts Monitors do not prevent inter-object deadlocks.

- 2 pts Monitors lock a class instance, not an entire class.

- 1 pts Java sync methods require identification of the methods. They don't try to determine if the object is modified.

- 3 pts With synchronized methods, non-synchronized methods can be used in parallel.

- **1 pts** Java synchronized methods provide enforced locking.
- **3 pts** Object oriented monitors are often provided in the language, and need not be implemented by the programmer.

QUESTION 9

9 Callbacks in AFSv2 **10 / 10**

✓ **- 0 pts** Correct

- **10 pts** No answer
- **2 pts** Callbacks occur when a file is updated, not to check if the cached copy is still OK.
- **10 pts** Not the purpose of an AFS v3 callback. It's for cache consistency.
- **5 pts** Callbacks go from server to caching clients when a file is updated.
- **8 pts** More detail required.
- **10 pts** AFS is a file system.
- **5 pts** Callback is to notify caching client of updates at other sites, not to validate that data has been received.
- **5 pts** Why does this have to happen?
- **2 pts** Not just for directories.
- **2 pts** Why would a file's status change without the client knowing about it?

QUESTION 10

10 PK certificates **9 / 10**

✓ **- 0 pts** Correct

- **10 pts** No answer
- **2 pts** Did not mention public key of issuer in certificate.
- **2 pts** Did not mention digital signature of trusted 3rd party in certificate
- **2 pts** Did not say that a mutually trusted third party is needed to sign the digital signature
- **4 pts** Did not correctly say that the trusted 3rd party's public key, which matches the 3rd party's private key used to sign the digital signature, is needed to decrypt the digital signature

- 1 Point adjustment

The second item is not really another signature, it is just the public key of the party.

QUESTION 11

11 Zombie states **8 / 10**

- **0 pts** Correct
- **10 pts** No answer
- **5 pts** A final state indicates that a process has finished executing all of its code. However, it has not yet been cleaned up.
- **5 pts** It allows the parent process to check its exit status and possibly perform other cleanup tasks.
- **10 pts** wrong answer
- **2 pts** all of the memory and resources associated with a zombie process are deallocated

✓ **- 2 pts The parent process checks the exit status**

- **5 pts** Parent process waits for child process

QUESTION 12

12 Fairness and scheduling **10 / 10**

✓ **- 0 pts** Correct

- **10 pts** No answer
- **5 pts** Performance is a vague term. What precisely do you mean? Your example is unclear.
- **1 pts** Precisely what do you mean by performance here? Fairness itself is one aspect of performance.
- **10 pts** That's not a property.
- **5 pts** Why is continuity desirable?
- **2 pts** Even a fair scheduler would not insist on a blocked process getting an equal time slice.
- **2 pts** Need better description of why.
- **3 pts** Fairness and preemption aren't the same thing. Unfair algorithms can also use preemption.
- **1 pts** You're talking about turnaround time, not response time.
- **2 pts** Your description does not say why throughput is damaged.
- **2 pts** Disk latency not really relevant here.
- **2 pts** That's not throughput. Throughput is the amount of useful work completed in a unit time. You're talking about turnaround time.

QUESTION 13

13 Free list ordering **10 / 10**

✓ **- 0 pts** Correct

- 10 pts No answer

- 8 pts Incorrect understanding of memory free list.

- 2 pts Missing details or not a very good explanation for ordering by size.

- 2 pts Missing details or not a very good explanation for ordering by address.

- 4 pts Wrong answer for ordering by size.

- 4 pts Wrong answer for ordering by address.

QUESTION 14

14 Page replacement for looping sequential workloads **10 / 10**

✓ **- 0 pts** Correct

- 10 pts No answer

- 3 pts More specifics on the alternate algorithm.

- 4 pts Clock algorithms approximate LRU, so they aren't likely to do well.

- 1 pts How could we know this?

- 5 pts What other algorithm to use?

- 2 pts How to practically implement your chosen algorithm?

- 3 pts How will you do lookahead at the end of the loop area? How can you know?

- 1 pts How to practically order the pages?

- 3 pts How to choose which chunks to replace? Bad if you choose the LRU chunks.

- 2 pts How do you know when you've reached the end of the loop and need to move to the head?

- 5 pts Problem is vast number of page misses.

- 5 pts This algorithm is no better than LRU, since it guarantees maximum paging.

- 3 pts Why would you see constant page replacement?

- 3 pts Which pages do you designate for swapping?

QUESTION 15

15 Load and stress testing **4 / 10**

- 0 pts Correct

- 10 pts No answer

✓ **- 4 pts** Did not say that load testing measures system performance under particular loads, usually loads that are expected to occur in actual operation

- 4 pts Did not say that stress testing is used to understand how a system will perform in unusual circumstances.

✓ **- 2 pts** Did not mention that stress testing is most likely to be used in systems that cannot afford to fail.

Final Exam
CS 111, Principles of Operating Systems
Winter 2018

Name: ASHWIN VIVEKANANDH

Student ID Number: 204705339

This is a closed book, closed note test. Answer all questions.

Each question should be answered in 2-5 sentences. DO NOT simply write everything you remember about the topic of the question. Answer the question that was asked. Extraneous information not related to the answer to the question will not improve your grade and may make it difficult to determine if the pertinent part of your answer is correct. Confine your answers to the space directly below each question. Only text in this space will be graded. No question requires a longer answer than the space provided.

memory management

1. What two mechanisms of a modern memory management system lead to the need for scatter/gather I/O? Why do they do so?

Scatter/gather I/O is when I/O requests are gathered from different parts of memory into a buffer, and I/O responses are scattered to the various places in physical memory. This method of I/O is a result of the paging, VM mechanism, which splits memory into small frames, and direct memory access, which allows I/O devices to access memory without CPU at memory speeds. Now, when I/O requests are made in virtual memory, they are scattered due to paging, and can be gathered using DMA into a buffer to perform I/O

2. For a journaling file system that only puts metadata in the journal, the data blocks must be written to the storage device before the journal is written to that device. The process requesting the write is informed of its success once the journal is written to the device. Why is this order of operations important?

This order of operations is important because if there is a crash between the writing of the metadata (inode, bitmap) and the data block, the inode and bitmap, now in storage, will point at garbage. If we write the data before, and then a crash occurs, it's as if the write never happened since the inode and bitmap won't be in storage to point at this data. Upon reboot, they will be written, and we can proceed like normal.

3. Does a URL more closely resemble a hard link or a soft (symbolic) link? Why?

→ A URL resembles a symbolic link because a symbolic link is essentially a file that contains the pathname of the file it is linked to.

→ A URL contains a pathname to a webpage on the web that it identifies. It points to this data in the same way a symbolic link points to a file.

→ when a file is removed its symbolic link is no longer valid, and is dangling. Similarly, when webpage is removed, the URL will no longer be valid and will return an ~~error~~ HTTP error.

4. What is the benefit of using password salting? Why does it provide this benefit?

Password salting is when you "salt" the password by adding a random sequence of characters to the end of a password so that it is harder to guess.

It provides this benefit because passwords are usually a test of what you know, which could be duplicated by a 3rd party with the right brute force. Adding this random string makes it harder to use brute force, and converts this into authentication based on what you have.

5. In performance evaluation of systems software, what is a factor? Why is the choice of factors important in such evaluations?

→ A factor is some system parameter that is varied during the testing of systems software. For example ~~an~~, varying the number of threads in a program would affect the performance of the program in certain ways.

→ Choice of factors is important for several reasons:

1. they can help identify bottlenecks
2. they can identify parts of the system that are irrelevant / don't affect performance
3. they can help you test different cases that might not be possible in a real workload.

6. In what way is a file descriptor like a capability?

→ A file descriptor is like a capability because ownership of it by a process allows the process to read / write to the file

→ A capability is usually in the form of a bit pattern; A file descriptor is in integer form, which is ~~the~~ the same kind of numerical identifier

It is tough to revoke a file descriptor since the ~~file~~ ~~it~~ process itself has to close the file / exit.

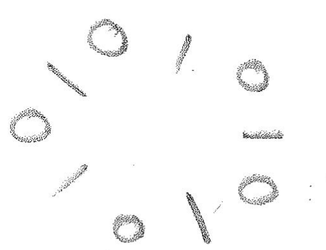
$1 \rightarrow 0$ locked, and inside

7. Consider the following proposed solution to the Dining Philosophers problem. Every of the five philosophers is assigned a number 0-4, which is known to the philosopher. The philosophers are seating at a circular table. There is one fork between each pair of philosophers, and each fork has its own semaphore, initialized to 1. `int left(p)` returns the identity of the fork to the left of philosopher p, while `int right(p)` returns the identity of the fork to the right of philosopher p. These functions are non-blocking, since they simply identify the desired fork. A philosopher calls `getforks()` to obtain both forks when he wants to eat, and calls `putforks()` to release both forks when he is finished eating, as defined below:

```
void getforks() {
sem_wait(forks[left(p)]);
sem_wait(forks[right(p)]);
}

void putforks() {
sem_post(forks[left(p)]);
sem_post(forks[right(p)]);

}
```



Is this a correct solution to the dining philosophers problem? Explain.

No, because if all philosophers call getforks() at the same time, each philosopher will have one chopstick (the one that was to their left) and when they call wait() on the right chopstick, they will all find the semaphore of the right chopstick at 0 and will decrement it and go to sleep.

Now, all philosophers are waiting for the philosopher on their right to relinquish their left chopstick, but this can't happen since none of them will be able to call putforks(). Thus there is a deadlock.

there is

~~If getforks()~~ w

8. What is the difference between synchronization using object-oriented monitors and synchronization using Java synchronized methods?

Monitors perform object level locking on Java objects whereas synchronized methods are methods that are intrinsically locked (only one thread can use the method at a time)

Synchronization with monitors enables multiple objects of the same class to run without race conditions

~~Synchronization with methods~~

9. What is the purpose of a callback in AFSv2? piece of state held by server for each client

A callback is used to notify a client of a change to a file in the server. The client will then know that the local copy on client's machine of the file is out of date, and can read the file from the server. This is better than having to repeatedly check/TestAuth instruction whenever you write/read to the server, to check if your local copy is synchronized with the server. You are given a promise by the callback that you will be updated.

10. Describe how a certificate allows us to securely obtain a public key for some other party. What information, in addition to the certificate itself, must we have to be sure of the certificate's validity? Why?

We decrypt the software using the software distributor's public key. Usually, this key is available in the certificate, along with two digital signatures: one of the distributor and the other being of a <u>mutually trusted party</u>. If any the of these signatures are missing it is not possible to verify the certificate is valid. The first signature is used to verify the public key and the second signature is used to validate the certificate as a whole

11. What is the purpose of a final state (also known as a zombie state) for a process?

The zombie state is the penultimate state of a killed process. ~~The~~ Its purpose is to inform the OS that it no longer requires its resources by having this status visible in the <u>process table</u>. periodically call some daemon that ~~its~~ The OS will issue some privileged instructions to <u>reap the resources</u>, stack, etc. of the process for reuse. When OS is done, the process is officially killed, and is <u>removed from the process table</u>.

response time, throughput, turnaround

12. If we use a scheduler algorithm that optimizes fairness, what other desirable property is likely to be damaged? Why?

→ An algorithm is considered fair if the scheduler does not neglect the running of certain processes for too long in favor of other processes. [illegible]

→ Fairness is implemented through preemption since we have to continuously switch out processes so no process is starved

→ Thus, throughput is damaged because all jobs will complete later than they usually would have due to the scheduler optimizing fairness by doing small part of each job over a longer time.

13. Elements in a memory free list could be ordered by size or could be ordered by their address. What is an advantage of ordering them by size? What is an advantage of ordering them by address?

→ Ordering by size is useful for implementing algorithms that search the free list to minimize external fragmentation. Now we can just search the free list knowing that the next chunk we search will be smaller/bigger than the current

→ i.e if chunks are ordered from small to big, we would go through the list until finding a chunk bigger than the request, this would be the 'best fit' chunk. Worst fit would be last chunk, etc.

Now we know where algorithms will find.

→ Ordering by address is useful for free chunk coalescing

since → when a chunk is returned to the free list, it just has to check its neighbors and if either is free, they can merge into a bigger chunk that is more useful to the process

$\underbrace{1,2,3,}_{1,2,4}$ 4

LRU, random, FIFO

$\underline{1,2,3,4}$

$\underbrace{1,2,3,4,5}$ 1,2,3

14. A looping sequential page workload runs sequentially through a set of pages of some fixed size, cycling back to the first page once it is finished with the last page. Why might an LRU page replacement algorithm handle this workload poorly? What kind of practical page replacement algorithm would handle it better?

LRU is the opposite of the ideal replacement algorithm for this specif workload. In the ideal case, we delay page faults for as long as possible, but LRU replaces the page that will page fault closest in the future. Thus it is a very poor choic here

→ RANDOM replacement is better than LRU as an algorith because it is more likely to replace page required further in the future than the page LRU chooses to replace. It is thus closer to the ideal replacement algorithm, which aims to replace the one referenced furthest in the future.

15. What is the difference between load testing and stress testing? When is stress testing most likely to be used?

Load testing is used to measure how an application/program performs under different types of workloads such as random/sequential, etc. Stress testing is used to test the capacity of a system and how much of this load it can handle without compromising performance.

Stress testing is most likely to be used in larger systems to evaluate scalability. For example, a distributed file system would require stress testing for how many users it could handle, same thing for networks, etc, that will experience heavy loads.