CS 111 Final exam

TOTAL POINTS

141 / 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 10 / 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

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 10 / 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 **10 / 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 **8 / 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

QUESTION 11

11 Zombie states 5 / 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

cannot read it clearly

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 8 / 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 10 / 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:

Student ID Number:

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.

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

The two mechanisms that lead to the need for scatter/gather I/O are paging and virtual addressing. Recall that we can use direct memory access to get an I/O device to talk to main memory through the bus without first having to go through the CPU. However, it communicates with physical, not virtual memory. In virtual memory, when a process requests I/O on a physical buffer, it thinks that this is a contiguous region of virtual memory. But due to paging, it is possible that this supposedly contiguous region is actual spread over a number of different pages, which map to pages in physical memory. Thus when a process requests a write to I/O from the contiguous information, it must "gather" the buffers which is spread over physical memory pages to write to the I/O device. When a process requests a read, it must take the bytes and "scatter" them over memory to the pages where the virtual buffer is located.

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?

The order of operations is important because of the possibility of system failure. Consider what would happen if we wrote the journal containing metadata information before we wrote the data blocks, and then the system crashed after we wrote the metadata to journal but before we wrote the actual data blocks. When the system reboots, it will perform the actions in the journal, writing the metadata to disk. This metadata will point to garbage, and the file system will be corrupted. If we write the data blocks before the journal, and the system crashes before writing the journal, we only have written the blocks to data. The write does not appear complete, ~~and since the~~ since the journal has not been written to the device, thus the file system can just try again. Thus it is necessary that we write the data first, as retrying writes is much better than thinking a write succeeded, and then updating the metadata, which now refers to corrupted data.

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

A URL most closely resembles a soft link. A hard link is the association of a name to an inode number. An file cannot be deleted until the link count in its inode is zero, as all hard links refer to that inode. In contrast, a soft link is completely independent of inode of the file it points to. Instead, it contains data to a path name of a file. The inode of the file the symbolic link points to has no idea ~~of~~ how many or which symbolic links it has to it. Thus a file can be deleted without removing all symbolic links, and the symbolic links now point to garbage. A URL is similar to a symbolic link, as a webpage may be deleted without deleting all URLs to it and is more of a path to the website than the actual website itself.

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

The benefit of using password salting is increased security, especially against brute force attacks. When a password is stored, a random value known as a salt is generated and appended to a password, and then the password+salt is encrypted
the hashed and salt are then stored.
using some hashing algorithm! let us say that the attacker found out someone's password by brute force. The attacker could then run the hashing algorithm to obtain the hash of the password since there are few cryptographic hash algorithms. If the attacker has ~~that~~ also obtained a list of the hashed passwords for all users, the attacker now knows the password for any user with the same password. If we salt the passwords, the hashes of these passwords all look different, even though they are the same. Thus if one user has their password compromised, other users with the same password are not compromised.

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

Factors are certain variables of a system that can be changed and have an impact upon the performance of your system. The amount ~~of factors that~~ you change a factor is known as a level. ~~When~~ It is important to think about useful factors and the number of factors you choose when testing your system, since you are effectively running factors × levels tests multiple times. It is also important to ~~choose to~~ consider the variability of factors you choose, and whether or not your factors have a direct rather than indirect effect on the metrics that you are using to measure your performance. For example, the factor of choice of persistent storage (HDD vs SSD) may be important if we are measuring throughput performance in file systems, but not so much if we are testing reliability [illegible] network protocols.

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

A capability is an object that is used to enforce access control. If you present ~~a capability~~ the required capability, then you are granted access. Likewise if you don't have it, then you are not granted access. File descriptors work this way. When a process requests a file descriptor using open, the operating system checks ~~the access~~ a separate access control list to see if the process has the ability to perform requested operations on a file (read, write, exec). If it does, then the OS returns a file descriptor to the process so it doesn't have to check the access control list every time. This file descriptor encapsulates the permissions a process has on a file, and can be presented to gain access to a file, and so is like a capability. It also enforces access control, as if a process does not have relevant permissions for the file, it is not granted the file descriptor and lacks the capability

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, this is not a correct sollution to the Dining philosophers problem.

All four conditions are still present. By design of the problem, mutual exclusion and circular dependency are present. Hold and wait is still present, since there is no guarantee a philosopher can get both forks at once, and can wait for the right fork while holding the left fork. There is still no preemption, so there is no lock breaking mechanism. For example, each philosopher could try to grab the fork on their left, and then be forced into a context switch.

~~If this is done before any philosopher~~, The system is deadlocked, as no philosopher can now gain the fork on their right.

Consider the following execution trace

P0 grabs left fork

P1 grabs left fork

P2 grabs left fork

...

P4 grabs left fork

P0-4 block because they cannot get the fork to their right. There is deadlock. We could fix this by enforcing that a philosopher must get both forks, or they will get none of them.

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

The difference is in lock granularity. Object-oriented monitors have a single lock on an object which is locked whenever any method is called on that object. Java synchronized methods also have a lock per object, but only lock the object on methods defined as synchronized, not all of them. We can provide a comparison using the principles of correctness and performance. Object-oriented monitors are correct, but suffer from poor performance, since they serialize any method upon the object. Java synchronized methods depend on the user to implement correctness, but benefit from finer lock granularity, and improve performance because they serialize access to the object less.

9. What is the purpose of a callback in AFSv2?

Callbacks were implemented to eliminate the wasteful polling found in AFSv1. Both polling and callbacks are intended to solve the problem of cache coherency in distributed file systems. In AFSv1, clients would poll the server to see if the version on the server was newer than the one on the client everytime the file is accessed. The problem was that the server was spending the majority of time responding to polling requests. In v2, callbacks were added. ~~These callbacks~~ Callbacks are when a server notifies any client using a file that the version on the server has been updated. This eliminates the unnecessary polling of v1, as instead of clients ~~updated~~ constantly checking whether a file has been updated, clients are notified whenever the file changes through callback.

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?

A certificate operates through public key encryption. We want to securely get the public key of some other party. The problem is that we cannot be sure it is their public key. We could use public key authentication, but not without their public key. If they just sent it directly, some attacker could inject their own public key and pose as the other party. A certificate is the public key of the party we want, signed by the private key of some trusted signing authority. As long as we trust the signing authority, we can trust the public key in the certificate. In addition to the certificate, we also need the public key of the signing authority for decryption already. Otherwise, we have no guarantee that the signing authority is actually a signing authority and not an attacker that signed some other attacker's key. For example, public keys of several trusted signing authorities are usually distributed with web browsers.

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

The purpose of a final state is for crash detection.

Once a process finishes execution, it is put into a zombie state, to be reaped at some later time. We can tell if a program crashed or ran to completion through its final state. This may be important when processes require that they be notified of the failure of dependent processes, and this can be done through the final state.

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

Throughput, or the amount of work we can do in a given amount of time is likely to be damaged. Consider a fair algorithm, like Round Robin, which gives equal time slices to processes and switches in circular order for fairness. This will incur a large amount of preemption, which will cause a large amount of context switches. Context switches are slow because they require transfer of control to the operating system, restoration of another process's state, and the cache is no longer warm. Thus we waste ~~time~~ a lot of time in context switches. This is time we could have used on actual jobs, and thus throughput is diminished.

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?

An advantage of ordering by size is if we want to find available amounts of memory quickly. Since the largest chunks of memory are at the start of a list, we are able to allocate memory quickly, as we are likely to find an available free area early in the list. An advantage to ordering them by address is that it is easy to implement a next fit algorithm. Instead of searching the entire list every time, we could simply start our search at the address we left off at using a guess pointer. We will also reduce external fragmentation, as we concentrate all allocations in sequential order, leaving large chunks of free memory near the end of memory.

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?

A LRU page replacement algorithm operates poorly because by the time a page in the cache will be used again, it will be the least recently used page and will have already have been evicted and the cache misses. Once a page is loaded in the cache, it will not be used again until the next cycle. Another algorithm might be a last in, first out algorithm, or most recently used. This is good because a page will not be referenced until the next cycle, meaning that we want to keep old pages in the cache. This can be accomplished using most recently used, as the newest pages are evicted. When we check a page, it is likely to still be in the cache, and we get a hit.

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

Load testing is ran using a relatively few number of test cases that are relatively simple. the goal of load testing is usually to test how long a system can remain in operation. On the other hand, stress testing is used to test a large number of complex test cases the operate in complicated and perhaps interconnected and unexpected ways. Stress testing is much more rigorous than load testing, and few systems correctly survive stress testing. As a result, stress testing is most likely to be used to find and ~~fix~~ edge-case situations or test the performance of highly critical systems that can not fail. In addition, load generators can also change their load mix and load amount to simulate different loads, and aging, while stress testing is less flexible.