CS 111 Final exam

NGUYEN; BAOLINH KIANNA

TOTAL POINTS

135 / 150

QUESTION 1

1 Scatter/gather I/O 8 / 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

- **1 Point adjustment**

salting does not provide encryption

QUESTION 5

5 Factors 8 / 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 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.

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 **10 / 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

QUESTION 12

12 Fairness and scheduling **8 / 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 **9 / 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.

- 1 Point adjustment

Need a bit more detail for load testing.

Final Exam
CS 111, Principles of Operating Systems
Winter 2018

Name: Baolinh Nguyen

Student ID Number: [redacted]

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.

DMA, mem-mapped I/O

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

Direct Memory Access and memory-mapped I/O in device handling lead to the use of scatter/gather I/O. This is because when using these two mechanisms to write to or read from devices, these reads and writes need to be contiguous. Thus, in order to do so, scatter-gather is used. When writing to the devices, pages of memory need to be "gathered" into contiguous buffers that can be sent to the device. When buffers of data are received from the device, this data needs to be "scattered" onto different pages in memory because we do not want memory to be constrained by a contiguity requirement.

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 is important because metadata is critical for correctness. If the data is written but a crash occurs before metadata is written, the file system will only need to retry the data write, which is idempotent. However, if metadata is written first, and a crash occurs, then the metadata will indicate that data has been written when in fact it is not. This means that data is garbage. Writes can only be considered successful once metadata has been journaled properly because this ensures that in the event of a crash, the journal persists and allows the file system to recover the contents because the data has already been written before the metadata was journaled. The file system only needs to recover the metadata corresponding to this write.

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

A URL resembles a soft link. Just like soft links, URLs can be in existence while the page the URL should link to may not exist. A soft link is a special file that contains a pathname a filesystem can follow - with no guarantees about whether this path is valid or not. It simply is a special type of file that the filesystem recognizes as a path to another inode - it is not another mapping for the inode. Thus, when the file itself is deleted, the symbolic link may still exist - just as the URL will still exist - but the file will not be found, just as a web page may not be found.

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

Password salting is used in the storage of passwords. Encrypted passwords are stored along with a salt in the computer and provide extra protection in the case that an attacker steals the encrypted password file and attempts a dictionary attack. In a dictionary attack, the hacker will hash all entries in a dictionary to compare them with the encrypted passwords. However, if a salt is added, the attacker will need to hash not only all entries in the dictionary but for each of these entries, will need to generate every possible salt in order to compare, which will be expensive.

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

A factor is an element you want to test that can be varied at different levels. You want to vary factors to see how these different levels change your system's performance. Choosing the wrong factors will tell you either incorrect or irrelevant information, depending on what you want to test.

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

A file descriptor acts like a capability in that by opening one, you are provided special privileges concerning the file. You use this file descriptor to continually access a file, meaning that you present the file descriptor data structure to the OS, which then allows you to access the files in certain ways. When a file is opened, the OS presents the process with the file descriptor, which allows the process to access the file itself, in certain ways, e.g. read only, read write, etc.

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)]);   — all can be interrupted here
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.

This is not a correct solution to the dining philosophers problem. There is a chance that all five philosophers call sem-wait(forks[left(p)]); and then each will grab the left fork before any can grab the right fork. This does not work because all philosophers will have a left fork, none will have a right fork so none can compute but they are all waiting. This can happen if each philosopher is somehow interrupted after they call sem-wait(forks[left(p)]); which means each gets a left fork but then is unable to get a right fork because all right forks have been obtained. Thus, they all wait for each other, deadlocked.

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

Using object oriented monitors provides mutual exclusion for an entire class using a semaphore. In a synchronized method, a mutex is used to protect certain methods, not the entire class itself. Because of the differing implementation, monitors will not suffer from starvation due to the queue created by the semaphore while in synchronized methods, there are starvation issues. In monitors, the class itself is protected meaning any interaction within the class or involving the class guarantees mutual exclusion. However, with synchronized methods, only selected methods are protected, not entire classes - you only get mutual exclusion inside some selected methods, not all of them.

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

A callback is used in AFS V2 to create an agreement between a client and a server that when a file the client currently has in its cache has been updated, the server will notify the client, who can then invalidate the cached version. This method is interrupt based and improves on NFS's poll-based method where a GETATTR call would be issued to check for cache consistency. The callback is used to ensure cache consistency between the client's cache'd versions and the server's updated versions. Callbacks are established on files that the client has in its cache. If these files are changed according to the server, then the server notifies the client it had previously established a callback for that file with, indicating that the cached version is no longer consistent. This prevents the server from being flooded with requests checking the consistency of a file, like NFS is flooded with GETATTR requests.

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 is created by a certificate authority and is used to distribute public keys of different entities. The certificate contains a digital signature that is a hash of the entity's public key and some other information. The certificate contains ~~a~~ an encrypted hash of this information. The user then uses the certificate authority's public key to decrypt the hash and the user also hashes the same information that the certificate authority hashed, which would be distributed along with the certificate. Then, by comparing these two hashes, the user can verify that the public key given has been validated by the certificate authority. In addition to the certificate itself, we need the public key of the certificate authority in order to decrypt some parts of the certificate and to ensure the certificate was generated by the correct certificate authority. This public key must be obtained through some out of band method, for instance, in a web browser.

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

The final state of a process or the zombie state | child continues after parent dies
occurs when a process has exited but its information still remains on the process control structure of the OS. This is used in the case where a garbage collection policy is enforced. Instead of cleaning out the process, the process is just marked as being in a zombie state, reducing overhead at that point. The system can later return to clean the process control list at a more convenient time. This delays the overhead of cleaning up after an exited process until later to improve performance.

throughput

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

A scheduler algorithm that optimizes fairness would hurt throughput, meaning that the amount of jobs completed would be low. This hurts throughput because by being fair, each thread would be allowed a turn, say, a specific timeslice. However, because it is fair, each thread only gets their turn and is then descheduled for another thread. If this process continues, threads will have a hard time running to completion and completing whatever job they were assigned - harming overall throughput. Jobs will not be completed as the threads will need to yield to ensure fairness.

easy to find good size chunk

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?

coalescing is easier

A memory free list ordered by size makes it easier and faster to find chunks of a suitable size that best fit request needs. It can also make searching faster as it can be ordered such that small, unusable chunks are at the end, meaning they are not searched.

Ordering a memory free list is advantageous in the case where the system wants to be able to coalesce blocks, or join free blocks located near each other to create a larger, more usable block. In the case where it is ordered by address, the system simply needs to look at the blocks and addresses surrounding the block in question and determine if it is possible to coalesce. If it is possible, the system simply needs to move the free list pointers around and change the freelist header to indicate the location and size of the recently coalesced blocks. This would be difficult in a list ordered by size.

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?

random

what Fung to be LRU will eventually be used again

An LRU page replacement algorithm performs poorly in the case where a looping sequential page workload runs sequentially through a set of pages. This is because the LRU will kick out pages that were used least recently. However, because the workload loops back to these pages, the algorithm must page them back in, even though they were LRU-ed out recently. The LRU replacement policy kicks out pages we will be using again consistently if the set of pages is more than the working set. Thus, a better page replacement algorithm would be a random page replacement algorithm, where instead of systematically paging out pages that were used least recently, a random page is chosen, which prevents needed pages from systematically being paged out.

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

[diff loads] [max conditions] systems · load behavior is unpredictable

Load testing means that the system is tested under differing loads, including a heavy load. Stress testing is done in such a way that puts the system through loads and conditions that it may not encounter typically. For instance stress testing includes situations wherein ~~a~~ a large number of clients are on a server or a huge number of requests are being sent. Stress testing tests the limits of the system. Stress testing is used in systems where robustness is important or where the system has a chance of being overloaded by many requests, clients, or may be presented ~~a~~ a very large load.