CS 111 Final exam

PATEL; DEVEN VIMAL

TOTAL POINTS

144 / 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 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 9 / 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 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: Deven Patel

Student ID Number: 104 766 465

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.

is a policy, so the ~~memory~~ virtual → system memory that implements it is what I mean.

paging, DMA

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

One mechanism that leads to the need for scatter-gather IO is paging, and the other is direct memory access. Paging is necessary since if not for paging, the memory ~~Page~~ could be stored contiguously and not need to be scattered. paging causes the need for scattering with the benefit of having non-contiguous memory that can be moved as desired. Moving this memory can have the benefit of easier swapping of working sets to the disk etc. Direct Memory Access leads to the need for scatter gather I/O since the pages, which are scattered, must be gathered to send to the ~~disk~~ device for it to do its task. When the device wants to put memory back, it will have to be scattered again as per the need of the paging system.

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?

First the JFS puts the data in the storage device. This must be first since we want to write metadata only after the data has been written or else The data may not have been written and ~~we crash~~ if we crash and see the metadata we may think the write went through and have corrupted data ~~space~~ really. Then the metadata is written and committed. ~~Once all of~~ → we checkpoint and It is committed only after all of the data is written. Once committed, we throw on the final identifier signalling a completed journal system write. Then we can tell the process the write is done since even if we crash ~~the JFS~~ and we can recover the write is already in the disk and the metadata is journalled. If we had told process earlier, ~~the~~ then crashed, the recovered process might later think the data is written when it was not.

here we commit then checkpoint ~~and write the~~

In this diagram we show the start identifier (Ts) and the metadata blocks we write, then a seperator indicating a commit, then a transaction end identifier

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

A URL more closely resembles a soft link. A soft link is much like a path that redirects to a file, but not a documented reference to it like a hard link. When all hard links pointing to a file's inode are gone, the file is gone, but the soft link remains, just in a broken state. Similarly if I have a URL and its site goes down, it will still point there but there is nothing there. The URL paths to the site, but it's not like a reference like a hard link to an inode. If all URLs are gone, the site can still exist.

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

Password salting has a benefit of preventing common dictionary attacks. A salt is a string of characters that can be added to a password. A dictionary attack is basically where a hacker has a bunch of common passwords he has already run through a hash, say, SHA-3. If you use a common password and just send it over a network, and it gets snooped, a hacker might be able to just locate the original password with his table. However, if you add a salt, then hash the password and send it over a network, a hacker most likely will not have it in his table and will have to know the salt and rebuild a table if he wants to crack your password. Doing that is much more painstaking and unlikely, so that is how a salt helps you.

since it's extremely inefficient, so a hacker probably won't do it

metric is the values you want to find of tendency and dispersion
factor is generally what you test/change, levels are actual values you change,

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

A factor is a variable to change essentially in performance evaluations. Some examples include like which file system do I test with, or how much memory do I allocate, or something of the like. This is not to be confused with a level, which tells you ~~the number~~ how much or which one exactly (for example ~~does~~ 10% more memory or EXT-3 file system). The choice of factors are important since you want to ① see which general conditions you can test to make your software robust and responsive and ② see which conditions your program is most hurt by. With these info and similar info you can gain wisdom and insight into your programs behavior, especially after you test with levels and workloads, and you can work to best optimise your program if you have not already.

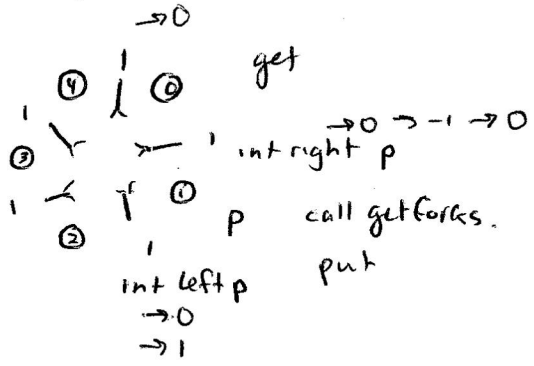
6. In what way is a file descriptor like a capability?

A capability is an access control mechanism which is held by a program (but more frequently the OS to diminish possibility of forgery) which is essentially a "ticket" of access, proving that the program has permissions (authorisation) to access whatever they have the capability for. File descriptor is like a capability because say I open a file, I usually specify read-only or something and get a file descriptor. That descriptor is then stored by the process and is used as a "ticket" to read the file from there on out, much like a capability. It shows that the process has been authorised to read that file, since it has already been assigned the capability (File descriptor).

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.

This is not a correct solution to the dining philosopher problem because of the non-blocking functions. If a philosopher already has two forks and ~~calls sem_wait~~ another calls sem_wait ~~while~~ over a fork already being used and does not block and wait in a queue for it, then he will not get the fork. Sem_wait ~~is~~ involves a caller decrementing to 0 and getting the resource or decrementing below zero and blocking and waiting in a queue for it. Sem_post involves being done with a resource and incrementing and waking the top waiter on a queue. If it is non-blocking that implies a philosopher does not sleep for the fork ~~if there is not the first one~~ in use so he cannot be woken and informed it is ready. Therefore this solution is arbitrarily bad.

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

Object oriented monitors are a whole class instance ~~lock~~ lock, and Java synchronised methods only lock methods within a class, so other methods can be used in parallel. Let us compare them the usual way we do in class. ↑ and not all must be synchronised

	Monitor	Synch Method
Correctness (1)	(1) Correct. ~~Only one thread in class at a time~~, not really any choices for programmer except class.	(1) Correct only one thread in func at a time so LONG as the
Fairness (2)	(2) → Relatively fair as far as that goes	(2) programmer picks right funcs to use on. Not particularly fair, depends on thread priorities
Progress (3)	(3) deadlock can happen with interclass dependencies	(3) deadlock risk
Performance (4)	Coarsley grained, bad	(4) ~~fine~~ finely grained, good!

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

* In initial ASF, there was an ~~Fetch~~ Auth thing to see if a file had changed or not and see if a client's local cache had gone stale (~~Test~~ Auth I think). The issue was ASF was overwhelmed with frequent ~~Fetch~~ Test Auth calls with ~~server~~ clients wanting to check if they had a stale cache for a file. To fix this, since AFS is stateful, in AFS ~~F~~v2 they made a callback. A callback essentially would, once the on server version of a file was changed, notify all the clients who have the file that a new version is ready and their whole file cache is stale (AFS caches whole file not block). This greatly increased scalability of afsv2 since it got ~~fixed~~ less TestAuth calls and allowed clients to know when their ~~file~~ file cache was stale.

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?

Certificate lets us securly get a public key of someone since it is signed by someone we trust. We need the signed, hashed info and everything to recreate the hash so we can be assured we are getting exactly what we intended to retieve. If we can use the trusted signer's public key to get the hash signed by his private key and then create the same hash using the same hashing function on the 3rd party public key the trusted has given us, then we can be sure it is correct since only the trusted could have signed with his private key, and if the public key of third party was changed in transit, we would know by ending up with a different hash than the signed one.

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

Zombie Process is a process that is done running essentially, but has yet to be cleaned up. It is still using memory and resources but has not been closed/removed by its parent process nor the OS. There are various uses for such a process. For one, it might be waiting on a process like its parent before it exits. In situations like this, if the parent does not cause its exit, we might assume something has gone wrong with the parent or the program had a bug and this can allow us to detect program issues. Also, a zombie process might be able to be used for overall program recovery and process reconstruction if something went wrong since it already has a lot of resources it needs. Also a zombie can be used as a process to exec another process from so the overhead of a fork() → exec() is not gone through.

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

If we optimise fairness, we are likely to hurt performance. Think about all of the common examples. Fairness implies that processes should have good response time and get about even time quantums. Algorithms like ~~red~~ round-robin are like that. This is at odds with performance since with algorithms like these there are many, many context switches between processes which implies an inordinate amount of over head, thereby detracting from performance greatly. Algorithms like shortest job first, or otherwise, which remove preemption and thus remove context switching are always more preformant and less fair for this reason that context switching eats precious cycles.

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 that if we use ~~a memory allocation~~ ~~principle like best~~ certain memory fitting algorithms we can quickly find a free section that is say best fit by attempting a binary search instead of linearly probing down looking for a best fit. A sorted optimisation like this can greatly improve the efficiency of various algorithms in various ways like the example above. On the other hand, ordering by address is great for issues of coalescing free space. If we have ordering by address we already know what adjacent free spaces are adjacent and can check and coalesce, but, if by size, we need to ~~search~~ search hard to find adjacent free spaces and coalesce to make a bigger one.

note
Worst fit ~~also~~ also gets better since we can just look at the back now where the largest sizes are.

Of course this is just one example, many page fitting algorithms can be improved like this. Improving best fit, we expect, would greatly reduce internal fragmentation

~~and~~ each algorithm has ~~certain~~ trade offs as we have learned.

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 page replacement, when a page needs to be swapped out of a processes working set into the disk, gets rid of the least recently used page. In other words, it gets rid of the earliest page access that has not been recently used in relation to other tasks. We actually usually use clock algorithms, but I will not get into that. Since we are running through sequentially, we might always get page faults if the fixed size is greater than the working set size since we will always throw away the earliest accessed page to bring in a new page and never reuse it, thus continuing the cycle.

Since you are only asking me to handle it better than LRU, not the best, an idea is to use a Last-in first-out like page replacement alg. this would cache the first number of pages then once working set was full only replace the most recently added page with a new page and this would happen till we got around, at which point, we could use our earliest cached pages which is better than no cached pages.

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

the whole list back to the front

Load testing is essentially under what load conditions my program can run. This involves testing various trial mixes, trial types, trial volume (amount) and trial sequence fidelity. By testing all of these, we can see how the program functions under various "loads" (those terms earlier essentially define load).

Stress testing is essentially running the program and throwing the machine into various levels of stress. This can include giving the program a lot of load, but can also include throwing a lot of errors, filling up the caches and ram, inducing thrashing, and all sorts of things. Very few programs are made to or expected to pass stress testing. We use it to check the robustness of a program and see how it preforms in such conditions. We hope it will preform gracefully, even in states of machine degradation.