CS 111 Final exam

Mark Andrew Guevara

TOTAL POINTS

123.5 / 150

QUESTION 1

1 Thread synchronization 10 / 10

✓ **- 0 pts** **Correct**

- 10 pts Incorrect/no answer

- 5 pts Answer unclear/needs more detail

- 5 pts On the right track - partial credit

QUESTION 2

2 RPC client stubs 7.5 / 10

- 0 pts Correct

- 5 pts Reason for having client stub incorrect/not present

✓ **- 2.5 pts** **Reason for having client stub unclear/needs more detail**

- 5 pts Functionality of client stub incorrect/not present

- 2.5 pts Functionality of client stub unclear/needs more detail

QUESTION 3

3 Scheduling deadlock avoidance 10 / 10

✓ **- 0 pts** **Correct**

- 3 pts Core 1: P1, then P4; Core 2: P3, then P2 is better

- 5 pts This could deadlock. Don't interrupt P4 to run P1

- 10 pts Not answering the question.

- 3 pts More details on the schedule.

- 1 pts You can schedule P2 in parallel with P1 or P4, though you might not benefit. Or, depending on timing of locks and unlocks, you might benefit.

- 10 pts Solve just with scheduling.

- 2 pts Preemption not necessary

- 5 pts Priority and preemption not necessary. Just schedule to avoid deadlocks.

- 8 pts Must not schedule P1 and P4 in parallel.

- 8 pts This requires lock breaking. You can solve it with pure scheduling.

- 10 pts Won't necessarily prevent deadlock.

- 2 pts Unless P3 takes at least as much time as P1, deadlock possible. Put P1 and P4 serially on one core.

- 5 pts How to do this with scheduling?

- 10 pts No answer

- 2 pts Unless P2 takes at least as much time as P1, deadlock possible. Put P1 and P4 serially on one core.

- 2 pts Unless P3 takes at least as much time as P4, deadlock possible. Put P1 and P4 serially on one core.

- 5 pts Depending on which task gets assigned to which core, this might work for this set of jobs, but it's not the case that any non-preemptive policy will work.

QUESTION 4

4 Santa semaphores 10 / 10

✓ **- 0 pts** **Correct**

- 2 pts Semaphore should be initialized to 0.

- 3 pts Production machine must post on each nutcracker.

- 3 pts Wrappers must use wait operation.

- 2 pts What particular semaphore operations should the machines perform?

- 2 pts Incrementation/decrementation backwards

- 3 pts Use semaphores, not locks.

- 1 pts Explicit yield not required

- 3 pts Initializing semaphore doesn't sleep wrapping machines. They must perform a wait.

- 2 pts Having the producer also wait will cause trouble.

- 2 pts Don't need both a lock and a semaphore.

- **3 pts** wait/post backwards
- **10 pts** No answer
- **10 pts** Answer the question.
- **1 pts** P is the wait operation, V the post operation.
- **2 pts** Basic semaphore operations will maintain the counter correctly.
- **5 pts** This approach blocks the producer, which isn't desirable.
- **2 pts** No description of why this works.
- **5 pts** You only need one semaphore, and you don't need to create/destroy semaphores.
- **5 pts** Only need one semaphore, not two or three.

QUESTION 5

5 Conflicts on a mutex **5 / 10**

- **0 pts** Correct
- **10 pts** incorrect
- **5 pts** No mention of time spent in critical section

✓ - **5 pts** **No mention of number of threads**

- **2.5 pts** Variation on number of threads : effect unclear
- **2.5 pts** Variation on time in critical section: effect unclear

QUESTION 6

6 Device driver interfaces **5 / 10**

- **0 pts** Correct
- **3 pts** Incorrect description of DDI

✓ - **3 pts** **Incorrect description of DKI**

- **2 pts** Wrong/no example of DDI

✓ - **2 pts** **Wrong/no example of DKI**

- **10 pts** No answer
- **8 pts** All drivers potentially use both DDI and DKI. Not proper descriptions or examples of either.

+ **2 pts** Some correct elements to DDI description

- **1 pts** DKI description not quite right.
- **2 pts** More detail on DDI description.
- **1 pts** DDI description not quite right
- **1 pts** DKI example vague.
- **0 pts** Click here to replace this description.
- **3 pts** DDI and DKI definitions backwards
- **1 pts** DDI defined by OS, not driver

QUESTION 7

7 FAT and System V file sizes **10 / 10**

✓ - **0 pts** **Correct**

- **10 pts** Incorrect
- **5 pts** Answer on FAT file system incorrect/not present
- **5 pts** Answer on Unix System V file system incorrect/not present
- **2.5 pts** Answer on FAT file system unclear/needs more detail
- **2.5 pts** Answer on Unix System V file system unclear/needs more detail

QUESTION 8

8 File system read-ahead **7 / 10**

- **0 pts** Correct
- **3 pts** Benefits related primarily to lower cost of accessing data already in cache.
- **1 pts** Doesn't require two extra reads.

✓ - **3 pts** **Major problem is poor use of memory cache.**

- **1 pts** Cache write-back not a problem. Not needed if you didn't write the page, unavoidable if you did.
- **8 pts** Incorrect understanding of read-ahead.
- **4 pts** More detail on costs.
- **1 pts** A few more details on cost.
- **1 pts** Read-ahead done on individual files, usually, not on other files in a directory.
- **1 pts** A few more details on benefit.
- **1 pts** You still have to do the correct read, whether you guessed wrong or didn't guess at all. Not necessarily more expensive.
- **1 pts** Not clear what you mean by "expose of not doing random reads."
- **1 pts** File system read-ahead not related to VM.
- **1 pts** Need not delay first read waiting for predicted ones.
- **1 pts** Won't increase working set size, which is defined by program's actual use.
- **5 pts** Nothing to do with fragmentation. Other costs?

- **4 pts** More details on benefits.
- **5 pts** File location not related to read-ahead benefits. Other benefits?
- **1 pts** File system read-ahead not likely to load TLB.

QUESTION 9

9 File update reliability **10 / 10**

✓ **- 0 pts Correct**
- **3 pts** Does not mention problem occurs on crash between writes
- **3 pts** Unclear from answer/Does not mention why data before metadata is not a problem
- **4 pts** Unclear from answer/Does not mention why metadata before data is a problem
- **10 pts** Wrong
- **4 pts** More detail needed
- **1 pts** Does not mention how errors with data first approach are easier to recover from

QUESTION 10

10 Symmetric cryptosystem authentication **10 / 10**

✓ **- 0 pts correct**
- **10 pts** incorrect
- **5 pts** non-repudiation : unclear
- **5 pts** no separation between authentication/encryption: unclear

QUESTION 11

11 Horizontally scalable distributed systems **8 / 10**

- **0 pts** Correct
- **10 pts** Incorrect

✓ **- 2 pts parallelism but no mention of interaction**
- **5 pts** Non interacting subtasks: unclear
- **5 pts** Stateless requests: unclear

QUESTION 12

12 Fixed partition memory management **5 / 10**

✓ **- 0 pts Correct**
- **5 pts** Internal fragmentation is still a problem
- **5 pts** Simple to implement is not a sufficient answer/more detail is required here
- **5 pts** you cannot in general coalesce partitions in this scheme
- **5 pts** You cannot in general allocate multiple partitions to the same program in this scheme. You would have to use overlays, which may not always be possible
- **10 pts** Wrong

- 5 Point adjustment

It is not always possible to know how much memory a process will require, the memory requirement can potentially grow beyond the partition and cause problems. If you could simply refuse to run a process in a fixed partition system you can also do so in dynamic partitions. This is not really an advantage

QUESTION 13

13 Trap table control **10 / 10**

✓ **- 0 pts Correct**
- **10 pts** No answer
- **8 pts** Program could use instruction to alter contents of trap table.
- **5 pts** Not altering trap handler code itself, but altering which code gets called on a trap.
- **7 pts** More details of why.
- **5 pts** Issue isn't value of address, but ability to change it.
- **5 pts** Everyone uses the trap table, but should not be able to change it.
- **4 pts** More details on why
- **2 pts** Just knowing location doesn't allow switch to privileged mode.
- **10 pts** Totally wrong.
- **3 pts** Trap table is in RAM.
- **7 pts** Could be a lot worse than overloading resources.

QUESTION 14

14 Throughput and response time

behaviors 6 / 10

- **- 0 pts** Correct
- **- 8 pts** Time increases exponentially to infinity due to limited queue size (other pertinent analysis regarding increasing queue size is okay for partial credit) (response time increasing purely due to context switching is not a good answer)
- **- 2 pts** Mention that throughput decreases somewhat due to overhead of queuing (explain why throughput doesn't decrease too much)
- ✓ **- 4 pts** **More detail is required in discussing the discrepancy between ideal and typical response time (explicitly mention dropped requests)**
- **- 8 pts** Increasing time due to conflicts is not a good answer
- **- 8 pts** Decreasing functionality is not a good answer
- **- 10 pts** Wrong

QUESTION 15

15 Soft real time scheduling 10 / 10

- ✓ **- 0 pts** **Correct**
- **- 10 pts** Turnaround time is not helpful, since it says nothing about whether deadlines were hit. It just tells you how long it took.
- **- 10 pts** Throughput is not helpful, since it says nothing about whether deadlines were hit. It just tells us how much work was performed.
- **- 10 pts** How do you determine correctness? Throughput is not helpful, since it says nothing about whether deadlines were hit.
- **- 10 pts** Fairness is not helpful, since it says nothing about whether deadlines were hit. Soft real time systems are usually oriented towards a single goal, anyway, so fairness is not a meaningful concept for them.
- **- 10 pts** Total time to completion is not helpful, since it says nothing about whether deadlines were hit. What if the total time to completion is OK, but 2/3s of the individual deadlines were missed? Average time to completion is no better.
- **- 10 pts** Response time is not helpful, since it says nothing about whether deadlines were hit. It just tells you how long it took to get some response.
- **- 3 pts** No example application.
- **- 10 pts** Clock cycles/instruction is not useful. The number of clock cycles per instruction is characteristic of the chip, not the scheduler or even the workload.
- **- 10 pts** Soft real time does not require pre-computation of a schedule. Somehow combining response time, latency, and throughput still tells you nothing about whether you missed deadlines, or by how much.
- **- 10 pts** "Performance time" is not useful, since it says nothing about whether deadlines were hit.
- **- 10 pts** I asked for a metric, not a scheduling discipline.
- **- 10 pts** You gave me three metrics, but none of them are good for soft real time, since none of them say anything about how many deadlines were missed or by how much.
- **- 10 pts** Average wait time is not useful since it says nothing about how many deadlines were missed or by how much. Your example is not a soft real time application.
- **- 10 pts** Totally wrong. DMA has nothing to do with soft real time scheduling, and is not a metric.
- **- 7 pts** Only useful if throughput is defined in terms of jobs that met their deadline, which isn't an ordinary definition of throughput. It also assumes that meeting a deadline for a very long job is great, even if it means missing deadlines for many short jobs, which might not be good. Your example isn't a soft real time job.
- **- 3 pts** File system writes aren't a soft real time application.
- **- 10 pts** This metric does not address how many deadlines were missed or by how much, which is what matters in soft real time. Your application is not a soft real time application.
- **- 10 pts** Throughput is not a good metric for soft real time, since it says nothing about how many deadlines were missed or by how much. Your example is not a

soft real time application, since I/O across a bus absolutely MUST complete, or you have a serious hardware error to recover from.

- 10 pts No answer

- 10 pts Your English is nearly incomprehensible in this answer. I do not know what "The real time scheduler will let the processor continue to run that will bad response time for other processes read to run" means.

- 8 pts Knowing what percentage of jobs must be completed on time for a particular application is not a metric. It's not clear, from your answer, precisely which metric you would use. The metric of "how clear the video is" is subjective and not useful at the OS level.

- 10 pts Turnaround time is not helpful, since it says nothing about whether deadlines were hit. It just tells you how long it took. Your example is not a soft real time application.

- 10 pts Seconds per instruction will depend on hardware characteristics of the system, not on the job mix or any scheduling issue. Your example has nothing to do with soft real time.

- 10 pts Your answer makes no sense. What do you mean by Most Commonly Used in the sense of scheduling? This isn't cache management.

- 10 pts Round Robin is a scheduling discipline, not a metric. As a scheduling discipline, it's terrible for soft real time, anyway.

- 7 pts Not too useful for adjusting your OS's scheduling behavior. Also, very expensive to obtain and too variable to work with.

- 10 pts This metric does not address missing deadlines, nor by how much. No application provided.

- 10 pts Wait time is not helpful, since it says nothing about whether deadlines were hit.

- 8 pts Not clear how you would do this for a soft real time system, in practice. Your example is certainly not a soft real time application.

Final Exam
CS 111, Principles of Operating Systems
Fall 2018

Name: Mark Guevara

Student ID Number: 704962920

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. Consider the following code. Will it correctly handle parent/child thread synchronization? Why or why not?

```
int done = 0;
pthread_mutex_t m = PTHREAD_MUTEX_INITIALIZER;
 pthread_cond_t c = PTHREAD_COND_INITIALIZER;

void thr_exit() {
        Pthread_mutex_lock(&m);
        Pthread_cond_signal(&c);
        Pthread_mutex_unlock(&m);
}

void thr_join() {
        Pthread_mutex_lock(&m);
        Pthread_cond_wait(&c, &m);
        Pthread_mutex_unlock(&m);
}

void *child(void *arg) {
        printf("child\n");
        thr_exit();
        return NULL;
}

int main(int argc, char *argv[]) {
         printf("parent: begin\n");
         Pthread_t p;
        Pthread_create(&p, NULL, child, NULL);
        thr_join();
        printf("parent: end\n");
        return 0;
}
```

The code will not handle thread synchronization.
It is possible that when the child process is created, it does not immediately run, and the parent process runs thr_join(). This causes it to grab the lock and wait. When the child process runs, it is unable to grab the lock in thr_exit(), so it cannot wake up the parent process, so the parent sleeps indefinitely.
It is also possible that the child runs first, grabs the lock, signals, releases, and finishes, at which point parent will run and wait for a signal the child cannot ever send again.

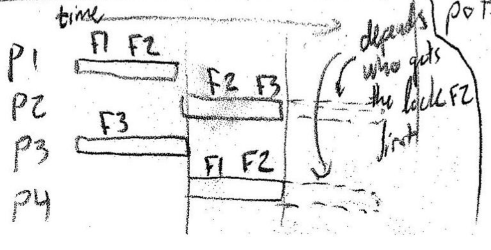
2. What is the reason for having a client stub in an RPC system? What functionality does it have?

An RPC, or remote procedure call system, operates by recieving procedure call requests from clients and sending back the results. The client stub is the interface a client process uses to send these requests and recieve the data back. It limits what can be sent to the server.

3. Assume you have 2 CPU cores (C1 and C2) and 4 processes (P1, P2, P3, and P4) to run. The processes all take around the same amount of time to execute, but must acquire locks on certain resources to run. P1 requires locks on F1 and F2. P2 requires locks on F2 and F3. P3 requires a lock on F3. P4 requires locks on F1 and F2. How could you use scheduling alone to ensure that no processes in this set deadlock, while making maximum use of the cores? Use of a technique other than scheduling to solve the problem will get no credit.

	P1	P2	P3	P4
F1	x			x
F2	x	x		x
F3		x	x	

P1 and P3 should be run first simultaneously, as they share no locks and can be scheduled with no conflict. P2 and P4 can then be scheduled simultaneously, despite their conflict over F2. One of the two processes will grab it first, run to completion, then release it, allowing the other process to run afterwards (assuming they are all doing something like spin-waiting or signaling so they can immediately run when the lock is freed). P1 and P4 should never run simultaneously, as they require the same locks and could potentially deadlock.



If they do no waiting, P2 and P4 should each be scheduled independently.

4. Like many other industries, Santa Claus has turned to automation to produce a sufficient quantity of toys to meet demand. He has built a machine that automatically produces nutcrackers, and purchased several slower machines that, on command, can grab a nutcracker and wrap it as a present. Unfortunately, the wrapping machines go haywire and attempt to wrap up the nearest elf if they try to wrap a nutcracker when none is available, so it's important not to order them into action if no finished nutcrackers are still unwrapped. Santa plans to use a semaphore to control the process, essentially using it to notify wrapping machines when a nutcracker is available. What operation on the semaphore should the nutcracker-producing machine perform? What operation on the semaphore should the wrapping machines perform? What should the semaphore be initialized to? Briefly describe how this synchronization process will work.

The semaphore should be initialized to zero, indicating there are no available nutcrackers. Whenever one is produced, the machine should signal the semaphore and increase it by 1. Whenever the wrapping machine is ready, it should wait on the semaphore and decrement it by 1. If there is a positive number of nutcrackers, the wrapping machines can immediately wrap. If there are >=0 nutcracker the semaphore will show that, and the wrappers will wait until one becomes available and the producer signals the semaphore.

5. A program uses multiple threads that have a mutex protecting a critical section. What are the main factors that determine the probability that two or more threads will have a conflict on this mutex? For each factor, what is the effect of varying values (high and low) for that factor on the probability of conflict?

1. How often each thread requests the lock
   - High number of requests means more potential conflicts
   - Few requests = fewer conflicts

2. How long the threads hold the lock
   - Long duration means more potential conflicts
   - Short duration means fewer potential conflicts.

6. Operating systems typically have two interfaces related to device drivers: the Device Driver Interface (DDI) and the Device/Kernel Interface (DKI). What is the purpose of each of these interfaces? Describe one kind of functionality you would be likely to find in the DDI and one functionality you would be likely to find in the DKI.

The DDI is a low-level set of instructions unique to every piece of hardware that defines how the hardware interacts with an OS. It is used to directly control what the hardware does when sent a request, e.g. how a specific HDD must rotate and scan to find a particular file.

The DKI is higher-level, and provides a standard interface of commands for interacting with a wide list of unique examples. For example, an HDD and an SSD would both be defined as drives, and as a result any process can interact with them in the same way with commands like read() and write(), regardless of the hardware implementation

7. What factors determine the largest possible size for a file in the FAT file system? What factors determine the largest possible size for a file in the Unix System V file system?

In FAT, the largest possible file would be one that uses every single available entry in the file allocation table. This could be limited by the size of the disk or the size of the pointers to each of the blocks (a smaller wordsize means fewer total blocks to work with + a smaller FAT table)

In Unix System V, a file's maximum size is determined by the number of block pointers in the file's inode, including the direct and indirect pointers. In theory the file system could be configured to maximize the number of blocks by using mostly triply-indirect blocks in each inode.

Unix is affected by the size of blocks in the system. Larger blocks mean more total storage, assuming there are more total blocks than pointers in the inode.

8. What would be the potential benefits of having a file system perform read-ahead automatically based on observed application behavior? What would be the potential costs of doing so?

Read-ahead is done due to the assumption of locality of reference. If a process is reading a large file, the potential savings are huge because the OS will not need to go to disk for every read, some will automatically be in memory for immediate use. This can greatly improve the efficiency of a process, as it will not need to block for I/O for as long. The drawback would be if a process is accessing many small files that aren't localized on the drive. Read-ahead would not provide any benefit, and instead would add additional overhead to the read call.

9. Why, for reliability purposes, is it beneficial to write out the data associated with a file update to persistent storage before writing out the metadata associated with the same update?

If a crash occurs between the data write and the metadata write, there is no issue because there is nothing in the metadata that references the data, so it is essentally treated as if it were not there. The file update can be repeated and there are no problems. If metadata were written first, there are potential issues. If a crash occurs between the metadata and data writes, the metadata will show that there is valid data for the file on the disk, when in reality the metadata points to garbage values that were never updated to the new data.

10. Why aren't symmetric cryptosystems ideal for providing authentication?

Symmetric cryptosystems rely on both parties sharing a common key for encryption and decryption. In order to start an encrypted session, one user must provide the other with this key. Over a network connection, a symmetric cryptosystem has no inherent way of verifying who it is sending the key; the reciever must trust that the sender is the correct party through some other means. A potential attacker could send a reciever their own key, and if the reciever wrongfully thinks it is the right person, they could start sending sensitive data to the wrong person. A system like public key encryption could be added to provide this authentication for sending and recieving the key.

11. What kinds of workloads will benefit most from distributed systems based on the principle of horizontal scalability? Why?

Horizontal scalability means that a server can easily add additional machines running the same software and distribute client requests between the original and new machines without any change in service for the clients. This is useful for systems with an increasing number of clients and, as a result, requests to the server. If there are too many requests, more servers can be added and the requests can be offloaded and balanced between multiple machines. This is also useful for systems where servers fail often, as it is likely that at least one server will be up at a given time to service requests.

12. What are two advantages of fixed memory partition memory management schemes? Why do such schemes have these advantages?

Fixed memory partition memory management means that when a process is created, it is allocated a specific amount of memory that cannot be changed. One benefit this provides is that all of a process's code, stack, and data is held contiguously in physical memory, so no overhead is added doing paging translations. Another benefit is that the OS will know at a process's creation time wether or not there is enough memory available to create the process. In a dynamically allocated memory partition system, a process could potentially starve out other processes of their allocated memory if it is given too much memory to work with alone, forcing the OS to use other resources like disks. With fixed memory, processes always have their data in memory.

13. Why must the ISA instruction that controls where to find the trap tables in a system be privileged?

If it were not privileged, a potential attacker could locate the table and see the location of every trap command. They could then potentially modify the trap table to point instead to a set of malicious code that the OS will then run with kernel-level permissions. The attacker could also replace the code of one of these commands at the location the trap table points to and achieve the same effect. By making the act of accessing the trap table privileged, these issues can be avoided as no process will ever see that information other than the kernel.

It is also simply bad practice to give processes access to code that should be handled by the kernel, since even not maliciously a process could mess up the state of another process if it knew how to access, for example, the trap for a read() command.

14. Figure 1 below shows a typical throughput vs. offered load curve for a system resource. Figure 2 below shows a typical response time vs. offered load curve for a system resource. As load increases, the response time goes to infinity, while the throughput drops less dramatically. Why does response time show a more dramatic trend than throughput?

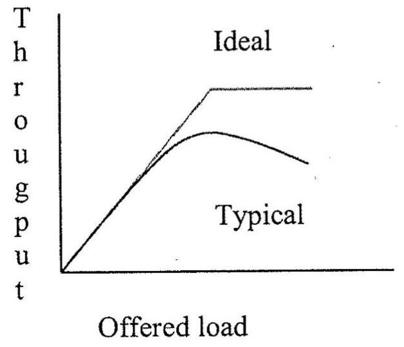


Figure 1

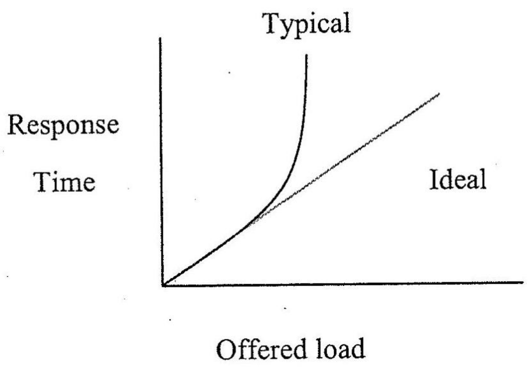


Figure 2

Throughput tends to fall off slowly because it is only really affected by the number of context switches; more swithes means less time spent actually running processes. Response time has a very dramatic increase because the system will eventually hit a point where its scheduler is so backlogged that new processes will essentially never get the chance to run. At some critical point resources like memory will fill, and new processes need to wait (potentially indefinitely) for space to free up for them to run.

15. What metric might you use to determine the effectiveness of a soft real time scheduler? Why would that metric be useful for this purpose? Give an example of an application where this metric would be helpful in determining if the soft real time system was meeting its goals.

Soft real time schedulers are used when a process would like to run at set real time interval, but has some room for failure, or timing t is off by so margin (for example updating a frame on a monitor). A useful metric would be a measure of how frequently these failures occur per a set number of expected executions. A large number of failures could indicate that other processes are using a lot of resources and the scheduler cannot keep up (e.g. with a monitor, how many frames are dropped could show how well the scheduler is performing).