CS 111 Final Examination
Spring Quarter, 2023

You have 3 hours (180 minutes) for this examination; the number of points for each question indicates roughly how long we think it will take to answer that question. Make sure you print your name and sign the Honor Code below. During this exam you are allowed to consult three double-sided pages of notes that you have prepared ahead of time, as well as any of the .c and .h files from your solution to Assignment 7. Other than these materials, you may not consult any other sources, including books, notes, your laptop, or phones or other personal devices. You may use the calculator app on your phone. If there is a trivial detail that you need for one of your answers but cannot recall, you may ask the course staff for help.

Note: do not write on the backs of any pages! We will be scanning the exams into Gradescope and won't see anything on the back sides. There are extra pages at the end if you need more space.

I acknowledge and accept the Stanford University Honor Code. I have neither given nor received aid in answering the questions on this examination, and I have not consulted any external information other than three double-sided pages of prepared notes.

_____
(Signature)

_____
(Print your name, legibly!)

_____
(SUNet email id)

Problem	#1	#2	#3	#4	#5	#6	#7	#8	#9	#10	#11	#12	Total
Score													
Max	10	10	16	10	6	9	8	6	15	3	24	40	157

Problem 1 (10 points)

Indicate whether each of the statements below is true or false, and explain your answer briefly.

(a) When a thread is preempted by the scheduler, any locks it holds are automatically released.

(b) When the dispatcher performs a context switch and the new thread is in the same process as the old one, it need not flush the TLB.

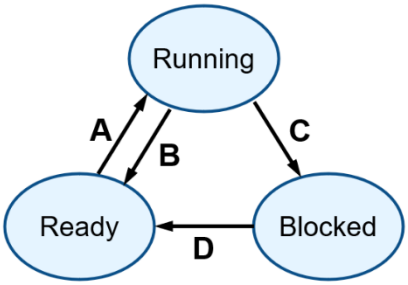
(c) Circular chains of references can cause memory leaks if reference counts are used for storage reclamation.

(e) A base-and-bound address translation mechanism can support shared libraries.

(e) If the block size is increased in a file system, internal fragmentation will also increase.

Problem 2 (10 points)

Consider the following diagram, which illustrates the various scheduling states that a thread can be in, as well as transitions between them:



(a) (2 points) What is the initial state of a new thread?

(b) (2 points) Give an example of an event that would cause a thread to undergo transition A (from Ready to Running).

(c) (2 points) Give an example of an event that would cause a thread to undergo transition B (from Running to Ready).

(d) (2 points) Give an example of an event that would cause a thread to undergo transition C (from Running to Blocked).

(e) (2 points) Give an example of an event that would cause a thread to undergo transition D (from Blocked to Ready).

Problem 3 (16 points)

(a) (4 points) You have been discussing your implementation of the Party class from Assignment 2 with a friend, and they suggest that instead of using a new thread for each incoming guest, it would be better to spawn a new child process for that guest. Would this work? Why or why not?

(b) (4 points) In C++ making a copy of a condition variable is prohibited. For example, code like the following generates a compiler error:

```
std::condition_variable a;
std::condition_variable b;
...
b = a;
```

Why would it be a bad idea to copy a condition variable?

(c) (4 points) Suppose that a collection of threads shares access to a set of locks. One of the threads executes code that may hold any number of locks at once, acquiring them in an unpredictable order. The other threads may acquire any of the locks, but none of these threads ever holds more than one lock at a time. Can deadlock occur over this lock usage? Explain your answer.

(d) (4 points) The journaling mechanism in Assignment 8 logs only metadata such as inodes; it does not log the data of regular files. Suppose you decided to log file data as well as metadata. Could you use the existing log entry types for this? If so, explain how. If not, explain why not and describe one or more new log entry types that would be needed to log file data.

Problem 4 (10 points)

Your disk needs to service this list of requested sectors: 15, 1567, 85, 778. In addition:

- The disk head is currently at sector 800.
- The requests arrived in the order listed (sector 15 arrived first); no additional requests arrive while these are being serviced.
- You can assume that the seek time from one sector to another is proportional to the difference between the sector numbers.
- For the scan algorithm, sectors are serviced in order from lower sector numbers to higher sector numbers.

(a) (6 points) For each disk scheduling algorithm (FIFO, shortest positioning time first, and scan), list the order in which the requests will be serviced.

(b) (4 points) Which disk scheduling algorithm will result in the shortest total seek time? Explain your answer.

Problem 5 (6 points)

(a) (2 points) What is the difference between deadlock and starvation?

(b) (2 points) Describe (briefly) a situation that results in deadlock.

(c) (2 points) Describe (briefly) a situation that results in starvation.

Problem 6 (9 points)

LRU (least recently used) was introduced in class as a potential page replacement policy, but it is not practical to implement in practice. The clock algorithm was then introduced as an easier-to-implement approximation to LRU. Consider each of the following modifications to the clock algorithm: if that modification were made independently, would the clock algorithm behave more or less like LRU? Explain your answers briefly.

(a) (3 points) Eliminate reference bits entirely (i.e. the algorithm behaves as if reference bits are always cleared for every page).

(b) (3 points) Whenever the algorithm reaches physical page 0, it switches direction (first it sweeps pages in increasing order from 0 to N, then it sweeps back from N-1 to 0, then up from 1 to N, and so on).

(c) (3 points) Instead of replacing a page immediately when its reference bit is zero, the algorithm only replaces a page if the reference bit was zero on 3 consecutive scans of the page.

Problem 7 (8 points)

For each of the following pieces of information, which are stored in page map entries, indicate whether the item must be included in TLB entries, and explain why or why not.

(a) The physical page number for the page.

(b) The "present" bit, which indicates whether the page is currently present in physical memory.

(c) The "kernel" bit, which allows access to the page only when in kernel mode.

(d) The "read-only" bit, which either allows or disallows writes to the page.

Problem 8 (6 points)

You are developing a software package that uses another package developed separately by someone else. Your package will contain a file that is actually a link to a file `libnet.a` in the other package and you will use this link when compiling your package. You know that the other developer will occasionally delete `libnet.a` and replace it with a newer version.

(a) Suppose that you want your link to automatically refer to the newest version of `libnet.a` Should you use a hard link or a symbolic link? Explain your answer briefly.

(b) Now suppose that the other developer tends to introduce bugs in new versions of `libnet.a`. You have tested the current version and know that it works, and you want to keep using that same version, even if the developer "upgrades" the file. Should you use a hard link or a symbolic link? Explain your answer briefly.

Problem 9 (15 points)

For each of the following approaches to managing memory or storage, indicate whether the approach suffers from significant performance degradation as memory becomes nearly full. Explain each answer briefly. You may assume that the memory or storage doesn't ever fill completely.

(a) First-fit memory allocation

(b) Memory allocation based on garbage collection

(c) A paged virtual memory system

(d) A file system that uses a bitmap to keep track of free blocks

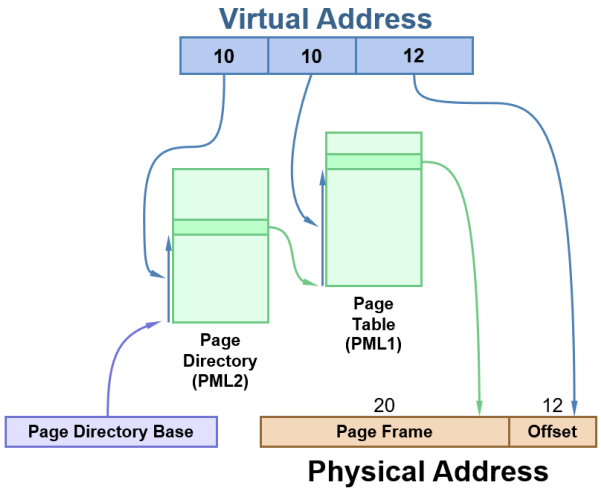
(e) A flash translation layer, which manages storage of a flash memory device

Problem 10 (3 points)

In 1-3 sentences, explain how crash recovery helps substitute some need to trust the file system.

Problem 11 (24 points)

Intel has implemented several different virtual address translation mechanisms for the x86 processor. The x86-64 mechanism discussed in lecture is the latest version. The diagram below illustrates an earlier approach called x86-32:



(a) (3 points) How large are individual page map entries in this scheme (in bytes)? Explain your answer briefly.

(b) (3 points) How large is the Page Directory Base register, in bits? Explain your answer briefly.

(c) (5 points) With this mechanism, how many physical pages will be consumed by a process with one page of code, one page of data, one page of stack? Explain what each page will be used for.

Problem 11, cont'd

(d) (5 points) How large would the stack have to grow before another page table would need to be created? You can express your answer as a power of 2.

(e) (4 points) Describe two advantages of this addressing scheme over the x86-64 scheme discussed in lecture.

(f) (4 points) Describe two advantages of the x86-64 scheme over this one.

Problem 12 (40 points)

Extend your solution for Assignment 7 by implementing the following function:

```
int find_indirect(const struct unixfilesystem *fs, int block_num)
```

The `block_num` argument is the number of a particular sector on disk. This function must determine whether `block_num` is used as an indirect block (but not a doubly-indirect block) in a file in the file system. If so, it will return the inumber of the file containing the block; if not it will return 0. If an error occurs, `find_indirect` will return -1.

Here are some additional details and simplifications:

- Your solution may invoke any of the functions defined for Assignment 7, and you may assume they have been implemented correctly.
- For this problem you do not need to print a message for each error condition; returning -1 is sufficient.

Use this page if you need extra space for any problem on the exam

Use this page if you need extra space for any problem on the exam