CS111 Midterm Exam

Raymond Yauhming Kwan

TOTAL POINTS

89.5 / 110

QUESTION 1

Principles 10 pts

1.1 Define Info Hiding 2 / 2

✓ **- 0 pts** Correct

1.2 Value of Info Hiding 1.5 / 3

✓ **- 1.5 pts** miss "give greater flexibility to change the internal details"

1.3 Define Modularity 2 / 2

✓ **- 0 pts** a system is composed of distinct sub-components

1.4 Good Modularity 1 / 3

✓ **- 2 pts** not necessarily a characteristic of good modular decomposition; or it's a basic characteristic

QUESTION 2

ABIs and APIs 10 pts

2.1 ABI acronym 2 / 2

✓ **- 0 pts** Application Binary Interface

2.2 ABI definition 3 / 3

✓ **- 0 pts** a binding of API to ISA

2.3 ABI vs API 0 / 2

✓ **- 2 pts** wrong

2.4 When API over ABI 3 / 3

✓ **- 0 pts** They are using different instruction set architectures.

QUESTION 3

Libraries 10 pts

3.1 Static library - advantages 0 / 3

✓ **- 3 pts** no need to implement or explicitly include the module in the compilation or linkage edit.

This is not different from explicitly included object modules

3.2 Which modules loaded 0 / 3

✓ **- 3 pts** the linkage editor deals with unresolved external references by pulling in the first module (in the specified library search order) that can satisfy it..

3.3 Shared library - advantages 3 / 4

- 1 Point adjustment

vague

QUESTION 4

Multi-Level Queues 10 pts

4.1 What problem they solve 1 / 2

✓ **- 1 pts** Not enough. Should clearly mention that different processes have different behavior or we may want to give different processes different time slices.

4.2 What drives queue changes 4 / 4

✓ **- 0 pts** Correct

4.3 Consequences of wrong queue 4 / 4

✓ **- 0 pts** Correct

QUESTION 5

Fixed Paritition Allocation 10 pts

5.1 problem with it 3 / 3

✓ **- 0 pts** Internal Fragmentation

5.2 effect of special sub-pools 3 / 3

✓ **- 0 pts** Correct

5.3 problem with special sub-pools 0 / 2

✓ **- 2 pts** incorrect

5.4 preventing that problem 1 / 2

✓ **- 1 pts** Incomplete

This should be done dynamically

QUESTION 6

Paging MMU 10 pts

6.1 diagram MMU, translation **5 / 5**

✓ **- 0 pts Correct**

6.2 info in page table entry **3 / 3**

✓ **- 0 pts Correct**

6.3 motivation for TLA buffers **2 / 2**

✓ **- 0 pts Correct**

QUESTION 7

Synchronization Terminology 10 pts

7.1 indeterminate **2 / 2**

✓ **- 0 pts Correct**

7.2 non-deterministic **2 / 2**

✓ **- 0 pts Correct**

7.3 race condition **2 / 2**

✓ **- 0 pts Correct**

7.4 critical secstion **1 / 2**

✓ **- 1 pts partial answer**

race conditions may occur outside of the CS, CS must be serialized

7.5 atomicity **2 / 2**

✓ **- 0 pts Correct**

QUESTION 8

Correct locking criteria 10 pts

8.1 criteria and mechanisms that fails **4 / 4**

✓ **- 0 pts Correct**

8.2 criteria and mechanisms that fails **3 / 3**

✓ **- 0 pts Correct**

8.3 criteria and mechanisms that fails **3 / 3**

✓ **- 0 pts Correct**

QUESTION 9

Asynchronous Completion mechanisms 10 pts

9.1 semaphores vs condition variables **3 / 3**

✓ **- 0 pts Correct**

9.2 why they differ **3 / 3**

✓ **- 0 pts Correct**

9.3 when choose semaphores **2 / 2**

✓ **- 0 pts Correct**

9.4 when choose condition variables **2 / 2**

✓ **- 0 pts Correct**

QUESTION 10

Enforced locking 10 pts

10.1 advantage of enforced **4 / 4**

✓ **- 0 pts Correct**

10.2 when choose advisory **4 / 4**

✓ **- 0 pts Correct**

10.3 requirement for enforced **0 / 2**

✓ **- 2 pts clients cannot directly access the protected object without going through methods that enforce locking**

QUESTION 11

Clock Algorithms 10 pts

11.1 what problem they solve **2 / 2**

✓ **- 0 pts Correct**

11.2 elements of clock algorithm **2 / 2**

✓ **- 0 pts Correct**

11.3 refrigerator LRU algorithm **2 / 2**

✓ **- 0 pts Correct**

11.4 approximation of LRU **2 / 2**

✓ **- 0 pts Correct**

11.5 regrigerator working set algorithm **1 / 2**

✓ **- 1 pts must describe (1) progressive scan, (2) relative age computation, and (3) comparison w/target, all in the context of a refrigerator**

1: (a) Define "Information-Hiding" (in the context of s/w design):

Information-hiding means abstracting away or keeping the specific details of implementation under the covers so that users do not need or have access or knowledge of such details to use the program.

(b) Briefly explain why Information-Hiding is a good thing:

Information hiding allows for interfaces to be more well-suited and intuitive for the clients, rather than bogging them down with complexity and details of implementation. This is an example of appropriate abstraction.

(c) Define "Modularity" (without using the word "module"):

Separate larger programs into smaller sections that exhibit closely related functionality and together, coherently combine individual purposes to make up the whole program.

(d) Briefly describe a (covered in this course) characteristic of good modular decomposition:

Good modular decomposition should isolate failures such that if one module fails, it does not affect the success of another individual module, and any problem or bug can be identified to belonging to a single module.

2: (a) What does the acronym "ABI" stand for?

Application Binary Interface

(b) Define the term?

ABI binds the API to a particular ISA and specifies data formats, load modules, and linkage conventions among other things.

(c) Why is ABI compatibility preferable to API compatibility?

APIs are more easily evolved and changed than ABIs.

An incompatible ABI may require the software vendor to reprogram the entire system to fit the ABI.

(d) When might it be necessary or reasonable for two OSs that support the same APIs to not support the same ABIs?

If the two OSes are designed for different purposes and have different ISAs.

3: (a) Give one advantage of static (non-shared) libraries over user-supplied object modules.

Automatically get newest updated library on compilation.

(b) What determines WHICH object modules FROM WHICH libraries become incorporated into a load module?

For static libraries, the library specified is copied in.

For DLL's, a linkage editor looks to resolve references in the stub module in the Procedure Linkage Table. These are references to the specific routines that are asked for.

(c) Briefly list two advantages of shared libraries over ordinary libraries?

Shared libraries have deferred binding to link time and allow for code sharing.

4: (a) What fundamental problem (or truth about processes) motivates the use of multi-level feedback queues for process scheduling?

It is not often reasonable to know beforehand how long a process needs to run or what behaviors it has.

(b) State TWO DISTINCT ways a process might find its way onto the right queue.

One way is if a process is constantly preempted, requiring many time slices. This process may be moved to a queue that has longer time slices.

Another way is if a process constantly blocks for I/O or yields for resources, this process may move to a queue that favors response time and has smaller time slices.

(c) What would be the negative consequences of a process being on the wrong queue (provide an answer for wrong in each direction)?

(c1) If a process is on a queue that has too short time slices (process takes long time to run), then throughput will suffer as gratuitous context switches will make the process take much longer to complete.

(c2) If a process has too long time slices, then CPU utilization becomes inefficient as there are times when the CPU is available but not used.

5: (a) What is the primary problem associated with fixed-partition memory allocation (returning fixed sized regions that may be larger than the requested size)?

Internal fragmentation - there are unused spaces within memory blocks.

(b) Briefly explain how special pools of fixed-size buffers affect this problem?

Special pools of fixed-size buffers offer buffers of popular sizes so that the memory regions more closely match the requested size.

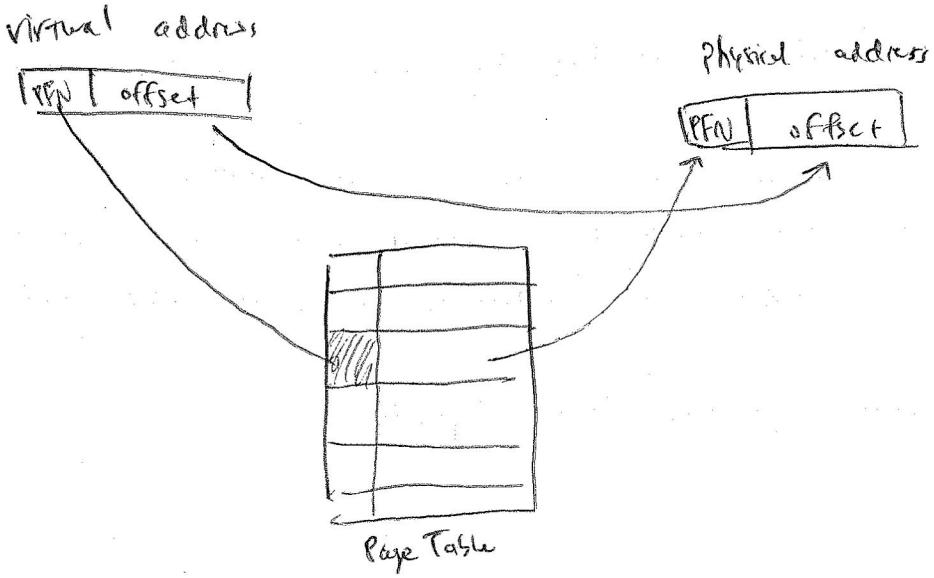
(c) What new problem is likely to arise when we create such pools?

Convoys may form to wait for a certain sized block.

(d) Briefly describe an approach for dealing with that problem?

Coalescing smaller pools into larger ones could serve requests that asked for the larger size buffers.

6: (a) Draw a diagram of a paging MMU, and illustrating how it translates a virtual address into a physical address.



(b) List (and briefly describe) two key pieces of information (other than the physical page frame number) that one might find in a page table entry

Valid bit, whether or not the page is in memory or not.

Dirty bit, whether this page has been written to, which would require writing out to disk before swapping out.

(c) Given the (relative simplicity) of paged virtual address translation, why has it been necessary to create Translation Look-Aside Buffers?

It takes too much time to go into memory to access the page table. TLBs give a cache of frequently used translations right on the CPU for faster lookup time.

7: Define (and distinguish the differences between) the following terms:

(a) "indeterminate"

When a program cannot say what the result will be, for example dividing by zero, this is an indeterminate result.

(b) "non-deterministic" ... distinguish from "indeterminate"

Non-deterministic means that given the same input, correctness and outputs could change, perhaps as a result of execution order.

Difference is that non-deterministic refers to the process while indeterminate describes a result.

(c) "race condition"

The execution order is timing dependent.

(d) "critical section" ... distinguish from "race condition"

A critical section is a section of code whose correctness depends on execution order and it involves a shared resource usually.

Difference is that critical sections are areas where correctness may be affected while race conditions are one of the reasons why the critical section gives different result.

(e) "atomicity"

Atomicity means for an operation to be able to complete without interruption, all-or-nothing.

8: The text gave three criteria in terms of which lock mechanisms should be evaluated. In class this list was expanded to four criteria. List and briefly describe three of those criteria AND provide an example of a real locking mechanism that does poorly on each criteria ... briefly explaining why it does poorly.

(a) Mutual Exclusion correctness

- Interrupt disabling does poorly on multi-processor parallelism because the interrupts are disabled only on the single CPU.

(b) Performance

- Spinning/spin-locks may actually delay the resource you're waiting for and wastes CPU cycles. Especially bad for many waiters/contention.

(c) Fairness

- ~~sleep-locks with waiting queue can result in higher priority.~~

Spinning and yielding can result in starvation, if another thread gets to the resource first

9: The text discussed both semaphores and condition variables as mechanisms that could be used to implement asynchronous event notification and waiting.

(a) Describe an important difference in what the waiter can assume after resuming after wait on a counting semaphore and on a condition variable.

After resuming after wait on a counting semaphore a waiter can assume that there is a resource there specifically for that waiter.

On a condition variable, a waiter cannot assume this; there may not be a resource there anymore for the waiter if another thread took it.

(b) Briefly explain why semaphores and condition variables are different in this respect.

Semaphores have a powerful mechanism to count the number of completed events, and can implement a waiting queue. Condition variables do not.

(c) Briefly describe a situation where this difference would make semaphores a better choice.

~~If there is one writer but many readers, then semaphores~~

If the resource is consumed exclusively, then semaphores may be better so that once a waiter wakes, it knows it has access to the resource, preventing spurious wakeups.

(d) Briefly describe a situation where this difference would make condition variables a better choice.

When there are many readers, readers can have shared access to the resource so only the last reader has to release the lock and readers after the first one do not have to block or wait in a queue. The readers can check just a condition variable to see if data is available to read.

10: (a) What is the primary advantage of "enforced" (vs advisory) locking?

You are guaranteed that the object will not be unsafely corrupted. (Advisory is unguaranteed)

(b) Describe a problem characteristic that would make "advisory" preferable?

Enforced locking may be too conservative at times. Advisory would then be preferable, giving users the flexibility for their own specific implementation.

(c) What is required to make it possible to "enforce" locking?

A user mode created implementation for accessing the object.

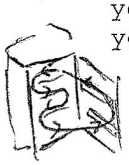
XC: (a) What otherwise difficult/expensive problems (be very specific) do "clock algorithms" address?

It's difficult/expensive to associate an exact "LRU" time with each reference. It'd be difficult to do in the MMU itself or in the CPU.

(b) What are the key elements of a "clock algorithm"?

Keep a circular list of each page and only when you need a page check if a page has been referenced. If it has, then mark it as if it has not been referenced. Swap out the page that has not been referenced in the last clock period. first in the list

(c) Briefly describe an LRU Clock Algorithm for deciding what old thing to remove from your refrigerator to make room for a new thing. I specifically want to understand how you implement your progressive scan, and what your "recently used" test is.



Have a circular list of places (spots) in the refrigerator (for example, the drawing to the left). Have a starting spot.

Every time you take an item out to use it, place a sticker on it when you put it back in. The sticker symbolizes you have used it. (only put max one sticker).

When it becomes time to make room for a new thing, starting from the spot you left off at last time, check if this item has a sticker. If it does, take the sticker off and go to the next one. If it does not have a sticker, this is the "least recently used" item; replace it. The next search will start at the spot after this.

(d) Explain why your progressive clock-scan yields a reasonable approximation of Least Recently Used.

The approximate "least recently used" item is the first item in your search that has not been referenced since the last swap. This is more often than not close to the least recently used item.

(e) Not unlike Global LRU, this seems a clumsy mechanism, in that it imposes a more-or-less constant replace-by age on all items, even though some expire in days while others are good for years. Describe changes to your scan algorithm and "recently used" test to implement "Working Set Clock" replacement. I specifically want to understand your progressive scan, what your "recently used" test is, and how you set the key comparison parameters.

Partition the refrigerator into 4 sections: items that expire in a few days, weeks, months, and years. Each section will have its own circular list. When determining which to replace, first determine which section it should belong to. Do the scan through that section. If the scan goes one full search through its individual section, then do a search through another section (only lower) and steal a spot from the other section. (if it's the days section, or higher)