CS111 Midterm Exam

Ankith M Uppunda

TOTAL POINTS

84 / 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 3 / 3

✓ **- 0 pts** Correct

QUESTION 2

ABIs and APIs 10 pts

2.1 ABI acronym 2 / 2

✓ **- 0 pts** Application Binary Interface

2.2 ABI definition 2 / 3

✓ **- 1 pts** partial answer

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 4 / 4

✓ **- 0 pts** Correct

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 0 / 2

✓ **- 2 pts** incorrect

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 0 / 2

✓ - 2 pts Incorrect

7.2 non-deterministic 2 / 2

✓ - 0 pts Correct

7.3 race condition 2 / 2

✓ - 0 pts Correct

7.4 critical secstion 1.5 / 2

✓ - 0.5 pts not mentioning 'correctness'

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 2 / 3

- 1 Point adjustment

your first argument does not speak to efficiency

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 1 / 3

✓ - 1 pts semaphore waiter can assume count was positive and has been decremented FOR THAT TASK.

✓ - 1 pts CV waiter only knows that condition was signaled, can make no assumptions about resource state.

semaphores have no query count operation

9.2 why they differ 3 / 3

✓ - 0 pts Correct

9.3 when choose semaphores 0 / 2

✓ - 2 pts You have not explained why semaphores are clearly better

9.4 when choose condition variables 1 / 2

✓ - 1 pts you did not show why CVs were preferred ... not merely less unpreferred

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 2 / 2

✓ - 0 pts Correct

CS111 Midterm 05/03/2018

Name Ankith Uppunda

Student ID # 004782343

Seat Row 5 Seat Col 13 Exam # 84

This is a closed-book, no-notes exam.

All questions are of equal value. Most questions have multiple parts. You must answer every part of every question. Read each question CAREFULLY; Make sure that

you understand EXACTLY what question is being asked

what type of answer is expected

your answer clearly and directly responds to the asked question

Many students lose many points for answering questions other than the one I asked. Misunderstanding a question may be evidence that you have not mastered the underlying concepts.

If you are unsure about what a question is asking for, raise your hand and ask.

Spend more time thinking and less time writing. Short and clear answers get more credit than long, rambling or vague ones.

Write carefully. I do not grade for penmanship, spelling or grammar, but if I cannot read or understand your answer, I can't give you credit for it.

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

Information-Hiding is hiding the details of design implementation from the user. This is done with interface definitions.

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

Information-Hiding allows user to use interface defmitions without a great understanding of how they are implemented. This allows many people to be able to use your shared code in a much faster manner.

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

Modularity is when you break a program into several pieces that are able to be independently tested.

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

If modular pieces are encapsulated or implementation hidden from user, properly and work cohesively together to perform operations that is good modular decomposition.

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

Application Binary Interface

(b) Define the term?

It is the interface the OS uses to talk to the hardware (Instruction Set Architecture) It defines things like register conventions, return adress conventions, stack frames, and allows the OS to be able to talk to ISA.

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

ABI compatibility will allow OS's to be compatible with multiple ISA's, whereas API's won't affect general usability of the OS or computer, just affect that particular programmer or programs that use API.

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

If the Instruction set Architecture of the computer the OS is running on is vastly different from the other.

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

It is much faster when executing since libraries are loaded in at linking time, instead of load time.

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

- Uses stubs to link to particular places in PM (physical Memory) where library is stored (shared library)
- The linker creates these stubs so that they can be properly referenced at load time.

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

- It takes up less memory than ordinary libraries since the library doesn't need to be loaded in memory multiple times (static).
- They can change after link time since the references are fixed at load time for shared.

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

• Conveys / Starvation / Response time

Each process needs time to run, CPU needs to switch between processes to prevent Starvation and make sure they are responded to quickly.

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

• Run process A if priority of A is the greatest, put new processes at top.

• Time Slices, where if the process keeps using them up it lowers its priority. This protects against User abusing blocking to make sure process stays high priority.

• Moving every process to top after Time S, protects against Starvation, Guarantees every process given fair time.

(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 it is lower priority than it should be it might end up being starved for a good amount of time depending on the time slice. This is a problem for response time, if response time is ca

(c2) If it is a higher priority than it should be it could starve another high priority process. This is catastrophic on real time systems that need some processes to run first before others.

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 is the primary problem.

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

Special pools of fixed size buffers would lower Internal Fragmentation, since the pools would be able to allocate exactly the right size for common requests.

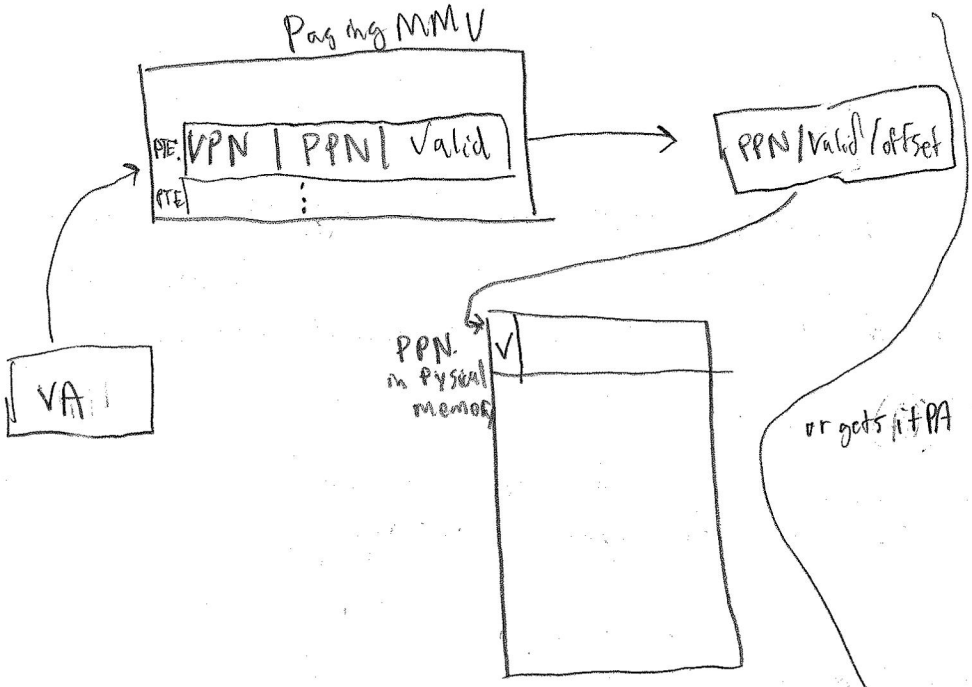
(c) What new problem is likely to arise when we create such pools?

External Fragmentation is likely to arise since you are allocating the right size, which might include small sizes.

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

Paging with Segmentation is one approach to dealing with it. There will still be some internal fragmentation, but limited to a fixed-partition memory allocation. This would solve External Fragmentation though.

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



Explanation

Hardware gets VA goes to paging MMU gets the PPN and extracts offset from VA, It checks if page is valid with valid bit, if not it generates page fault, otherwise gets the PPN, appends offset and goes to that place in physical memory.

or gets it PA

(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

Present/Valid Bit - Whether page is valid or not, if it isn't generate page fault.

- Privileges - Whether it is read or write or execute. Whether its shared
- Dirty Bit - whether page has been written to or not while in physical memory

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

TLB eliminates one check in physical memory. Since if is in the cpu cache it is much faster, whereas with a TLB miss you have to check the PT which is in Physical Memory and then go to the decoded entry in the PT which is also in Physical Memory. Two lookups is much more time wasting then one.

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

(a) "indeterminate"

It is when program produces wrong output from the expected one thread solution.

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

It is when parallel program produces multiple outputs depending on order of execution, but it could some time produce expected output unlike indeterminate.

(c) "race condition"

Its when two or more threads access the same place in memory creating non-deterministic behavior depending on who finishes first.

It is when output depends on order of events of two or more threads.

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

Critical section is a place in code where a race condition can happen, a place where two or more threads being at the same time creates non-deterministic behavior.

(e) "atomicity"

Either the instruction executes or it doesn't, there is no in between.

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) Correctness: Disabling interupts works very poorly on this, since it doesn't prevent Mutual Exclusion in multi core systems.

(b) Efficiency: Spinning locks,
- can't be run on OS thats not preemptable
- waste CPU time on processes that are spinning waiting for the lock.

(c) Fairness: Read/writer locks, since they have potential to Starve writers, based on if readers keep coming in.

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.

The waiter can gage how much of the resource is left or how many threads are waiting on Semaphore unlike a Condition variable.

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

The counter on the Semaphore is the primary difference that makes Semaphores more unique, to condition variables.

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

A multiple count sized buffer, since the wait will be notified when there are resources available, and it can take it from there.

(CV)

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

A ~~length~~ one sized buffer, where it can be empty or full. There is no need to use semaphores for this which are slower than CV.

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

• Provides Security, User has to lock versus Malicious/dumb User forgetting to lock

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

• Advisory there is more freedom since it's up to the user, they can deal with Mutual Exclusion themselves, maybe coming up with more efficient way.

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

A mechanism within the OS, you would have to disable interrupts briefly.

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

It eliminates the problem of having to traverse over everything to find the LRV, every time, In LRV you have to go through entire set every time whereas you don't with clock algorithms.

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

- The clock or pointer to next element to check
- The has recently been used bit on the page.

(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.

I would use postits for used bit, and for storing clock value.

So we start with empty fridge, we progressively fill it. If we use something we mark its used bit or put a used post-it on it one. In a full Fridge, say I want to put a bananay and the clock or pointer is pointed in the back, Then we first check that item, say its been used we set used bit to zero and go to next item. Since it was the last item we check the front. Say that hasn't been used, we take that out and put the bananas

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

It estamates the LRV and make sure to put pages in complexity something that hasn't been used in that clock cycle. It also makes sure to remember where its at, so it doesn't keep removing from front. It reasonable approximates it because of the clock as it makes sure to fairly check every item in list once before it checks the 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.

- Add a long expirary post it → if will expire in < 1 year mark zero
- Go through scan in C, but instead of just checking used bit, we skip over it if it is going to take a long time to expire other wise we exchange if given used bit is zero.
- Have a background process updating long expirary post it.

This would work extremely well, you would probably want a has used within last X clock cycles bit, to check if it actually is used.