15-410

"My other car is a cdr" -- Unknown

Exam #1
Mar. 4, 2019

Dave Eckhardt

Brian Railing

Synchronization

Checkpoint 2 – Wednesday, in Wean 5207 cluster

- Arrival-time hash function will be different

Checkpoint 2 - alerts

- Reminder: context switch ≠ timer interrupt!
  - Timer interrupt is a *special case*
  - Looking ahead to the general case can help you later
- Please read the handout warnings about context switch and mode switch and IRET *very carefully*
  - Each warning is there because of a big mistake which was very painful for previous students

Synchronization

Book report!

- **Hey, “Mid-Semester Break” is just around the corner!**

Synchronization

Asking for trouble?

- If you aren't using source control, that is probably a mistake
- If your code isn't in your 410 AFS space every day, you are asking for trouble
  - GitHub sometimes goes down!
    - » S'13: on P4 hand-in day (really!)
  - Roughly 1/2 of groups have blank REPOSITORY directories...
- If your code isn't built and tested on Andrew Linux every two or three days, you are asking for trouble

Synchronization

Google "Summer of Code"

- http://code.google.com/soc/
- Hack on an open-source project
  - And get paid
  - And quite possibly get recruited
- Projects with CMU connections: Plan 9, OpenAFS (see me)

CMU SCS "Coding in the Summer"?

Synchronization

Debugging advice

- Once as I was buying lunch I received a fortune

Synchronization

Debugging advice

- Once as I was buying lunch I received a fortune

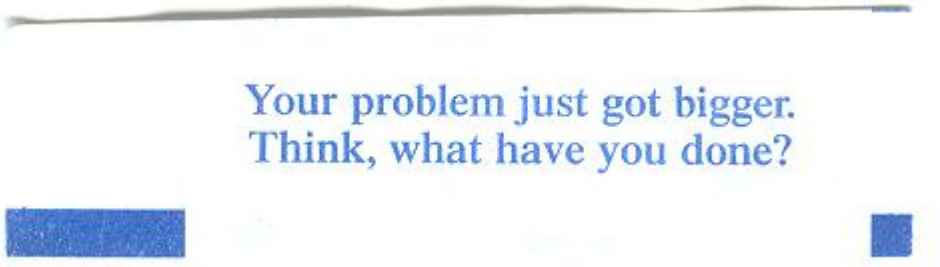

Image credit: Kartik Subramanian

A Word on the Final Exam

Disclaimer

- Past performance is not a guarantee of future results

The course will change

- Up to now: “basics” - What you need for Project 3
- Coming: advanced topics
  - Design issues
  - Things you won't experience via implementation

Examination will change to match

- More design questions
- Some things you won't have implemented (text useful!!)
- Still 3 hours, but could be more stuff (~100 points, ~7 questions)

“See Course Staff”

If your exam says “see course staff”...

- **...you should!**

This generally indicates a serious misconception...

- **...which we fear will seriously harm code you are writing now...**
- **...which we believe requires personal counseling, not just a brief note, to clear up.**

Outline

Question 1

Question 2

Question 3

Question 4

Question 5

Q1a – "The Three Kinds of Error"

Purpose: demonstrate grasp of a design tool

- Hopefully P2 involved deliberate design
- Hopefully P3 is involving deliberate design
- "Robust code is *structurally different* than fragile code"
- P3 requires not just code but *structurally non-fragile code*.

If you were lost on this question...

- We had a lecture on this topic (February 4)
- Other "odd" lectures to possibly review
  - Questions
  - #define, #include

Q1b – Register Dump

Question goal

- Stare at a register dump and form a plausible hypothesis
  - Why? Debugging P3 will require staring at bits to figure out what's wrong... this is a good way to figure out if some practice is needed

Hint

- A register that is definitely a pointer is pointing somewhere definitely wrong

Common issues

- Some seemed to suggest that the processor compares two pointer-like registers and declares a fault based on that
- There were claims that a fairly pointer-like register was pointing to a wrong place (when it was pointing to a very plausible place)

Q1 – Overall

Scores

- **11/59 students (~20%) scored 8/10 or better**
- **10/59 students (~20%) scored 2/10 or worse**

Q2 – Critical-section protocol

What we were testing

- Find a race condition (important skill)
- Write a convincing trace (demonstrates understanding)

Good news

- 52/59 students scored 12/15 (80%) or better

Minor issues

- Trace doesn't have an *exactly-repeating* part
- Trace doesn't *clearly identify* the exactly-repeating part

Alarming issues

- Trace requires a thread to "run at zero speed"
- Trace can't happen

Advice

- Don't "just start writing a trace" (ok on scrap paper)

Q3 – “Mockchain” Deadlock

Question goals

- Diagnose a deadlock situation, based on deadlock principles
- Show a trace
- Design (state) a solution

Q3 – "Mockchain" Deadlock

Question goals

- Diagnose a deadlock situation, based on deadlock principles
- Show a trace
- Design (state) a solution

Observations

- Showing circular wait, by itself, is not enough to show a deadlock
  - In particular, showing two miners in a cycle overlooks that other miners may release them
- Hold&wait isn't about only mutexes/semaphores
  - "Wait" can be for an abstract condition change
- "Global mutex" is an *emergency* solution to deadlock
  - Not a good solution
- Not all "tabular traces" were tabular

Q3 – “Mockchain” Deadlock

Scores

- 27/59 students (~45%) scored 11/15 (73%) or better
- 16/59 students (~25%) scored 5/15 or worse

Q4 – Double-condition variables

Question goal

- Variant of typical “write a synchronization object” exam question
- This was was probably “hard” (not “easy”, nor “killer”)

Some workable architectures

- One explicit queue, with search
  - This doesn't perform super-well, and doesn't scale well past double-cond to triple-cond etc.
- Two explicit reference queues
- Mathematics plus two blocking objects
  - Also: mathematics plus three blocking objects
  - Blocking objects are implicit queues

Q4 – Double-condition variables

Common issues

- Losing signals when multiple waiters are present
- "Seriously non-FIFO" solutions (many can starve)

Various other issues

- Buffers signals
- Loses signals
- Requires signals in 0-then-1 order
- "Blends signals"
  - sig(0) can awaken somebody who needs #1
- Deadlocks
- Various races

Q4 – Double-condition variables

Alarming

- Knowing how cvars work is very important!
  - World mutex is released and later re-acquired
  - Signals are not buffered (*due to semantics*)
  - Leveraging the world mutex for internal use generally goes wrong (e.g., threads get lost)
  - Signalling should not block (this is a deadlock factory)
- Each "multi-threaded field" needs *some* "lock plan"!

Q4 – Double-condition variables

General conceptual problems

- "x() takes a pointer" does *not* mean "x() must call malloc()"
- Assigning to a function parameter changes the *local copy*
  - It has no effect on the calling function's value
  - C isn't C++ or Pascal (luckily!)
- init() functions should not randomly refuse to initialize certain areas of memory
- See course staff about any general conceptual problems revealed by this specific exam question

Q4 – Double-condition variables

Synchronization problems

- Spinning is *not ok*
- Yield loops are "arguably less wrong" than spinning
  - Motto: "When a thread can't do anything useful for a while, it should block; when a thread is unblocked, there should be a high likelihood it can do something useful."
  - Special case: mutexes should not be held for genuinely indefinite periods of time
- Blocking should use an underlying primitive (cvar, semaphore) rather than implementing one manually

Q4 – Double-condition variables

Sample cases to try

- **W01 / W10**
- **01W (how many wake up?)**
- **WW0101 / WW1010**
- **WW0011 / WW1100**
- **W00W11 (how many wake up?)**

Q4 – Double-condition variables

Important general advice!

- It's a good idea to trace through your code and make sure that at least the simplest cases work without races or threads getting stuck
- Maybe figure out which operation is “the hard one” and pseudo-code that one before coding the easy ones?

Other things to watch out for

- Memory leaks
- Memory allocation / pointer mistakes
- Forgetting to shut down underlying primitives
- Parallel arrays (use structs instead)

Q4 – Double-condition variables

Outcome

- 15/59 students (~25%) scored 14/20 (70%) or better
- 23/59 students (~40%) scored 7/20 (35%) or worse
  - "Severe tire damage" is typically ~30%

Implications

- Being able to write this kind of code shows understanding of primitives and also hazards
- Life in P3 (and after) may involve embodying special-purpose synchronization patterns in code

Q5 – Scheduler states

Question goals

- Primary: test understanding of blocked vs. runnable
- Secondary: test understanding of trap vs. interrupt

Observations

- Parts A & C should be “easy to just write down an answer”
- Part B may require more thought
- Part D may require genuine thought

Q5 – Scheduler states

Outcome

- 29/59 students (~50%) scored 7/10 or better
- 13/59 students (~20%) scored 3/10 or worse

Implications

- Blocked/running/runnable is a core concept
- Trap/exception/interrupt is a core concept

Breakdown

```
90% = 63.0    6 students
80% = 56.0    6 students
70% = 49.0    8 students
60% = 42.0   13 students
50% = 35.0   16 students (rounded 34 up)
<50%         10 students
```

Comparison

- Median grade was 61%, so this wasn't an easy exam
  - But: last semester's median was 61% too

Implications

Some “curving” seems likely

- Details TBD

Score below 47?

- Form a “theory of what happened”
  - Not enough textbook time?
  - Not enough reading of partner's code?
  - Lecture examples “read” but not grasped?
  - Sample exams “scanned” but not solved?
- It is important to do better on the final exam
  - Historically, an explicit plan works a lot better than “I'll try harder”
  - *Strong suggestion*:
    - » Identify causes, draft a plan, see instructor

Implications

Score below 34?

- Something went *dangerously* wrong
  - It's *important* to figure out what!
- Beware of "triple whammy"
  - Low score on *all three* "middle" questions
    - » Those questions are the "core material"
    - » Strong scores on Q1+Q5 don't make up for serious trouble with core material
- Passing the final exam may be a *serious* challenge
- *Passing the class may not be possible!*
  - To pass the class you must demonstrate proficiency on exams (not just project grades)
- Identify causes, draft a plan, see instructor

Implications

"Special anti-course-passing syndrome":

- **Only "mercy points" received on several questions**
- **Extreme case: *no* question was convincingly answered**
  - **It is *not possible to pass the class* if both exams show no evidence that the core topics were mastered!**

Action plan

Please follow steps in order:

1. Identity causes
2. Draft a plan
3. See instructor

Please do not:

- "I am worried about my exam, what should I do?"
  - *Each person should do something different!*
  - Thus "identify causes" and "draft a plan" steps are individual and depend on some things I don't know

General plea

- Please check to see whether there is something we strongly recommend that you have been skipping because you never needed to do that thing before
  - This class is different