15-410

"My other car is a cdr" -- Unknown

Exam #1
Oct. 17, 2017

Dave Eckhardt

Dave O'Hallaron

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

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

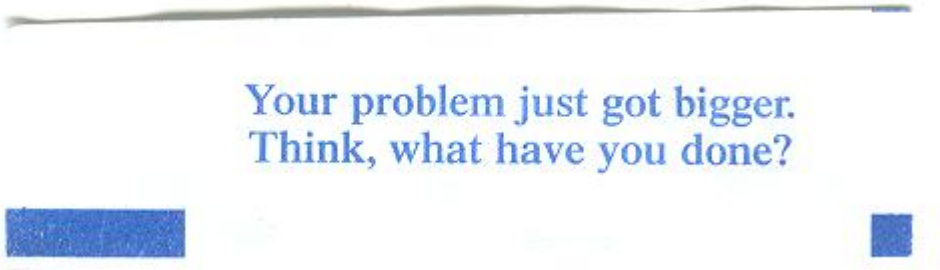

Image credit: Kartik Subramanian

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 – Deadlock Ingredients

Purpose: demonstrate familiarity with key mental tools for design

- Deadlock can be painful to fix in a large/complex code base
- Being conscious of hazards and options is important

Outcomes

- Generally reasonable answers
- Some people confused prevention vs. avoidance
- Some people skipped parts of the question

Q1b – Interrupt Acknowledgment

Purpose: Demonstrate understanding of the interrupt "life cycle"

- Key points
  - Who dismisses interrupts?
  - Who handles dismissal?
    - » What (precisely) does dismissal imply/enable?
  - What is the dismissal mechanism?

Outcomes

- Answers generally good
- Occasional alarming answers
  - "The dismissal is received by the IDT"

Q2 – Scheduling Transitions

What we were testing

- Key concepts: running, runnable, blocked
- Also: which Pebbles events cause transitions

Good news

- Half the class got 8/10, lots of people got 7/10

Other news

- One quarter of the class got 9/10 or 10/10... not a lot

Common issues

- Arc were labelled with non-Pebbles events
- sleep() and “SLEEPING” were not connected (!!)
- As hinted, we were expecting some single-node arcs

Be careful!

- “Blocked” is a core concept; precision here is wise

Q3 – “Nemo's Algorithm II”

What we were testing

- Primarily: ability to find and show race conditions
- Also: knowledge of what a c.s. algorithm should do

Good news

- Many people got a perfect score (60% of the class)

Bad news

- Several students alleged repetition but did not show it well
  - This is an important thing to get right
  - HW1 solution contained very explicit advice
- 20% of class did “emergency bounded waiting” trace
  - Please compare HW1 Q2 vs. exam Q3
  - Try to say how the algorithm change causes the behavior change

Q4 – “Multi-lock”

Question goals

- Diagnose a deadlock situation
- Design a solution, based on deadlock principles
- Slight modification of typical “write a synchronization object” exam question

Q4 – "Multi-lock"

Question goals

- Diagnose a deadlock situation
  - This part was easier than most deadlock questions
- Design a solution, based on deadlock principles
  - This part was harder than most deadlock questions
    - » The trace was consistent with multiple designs, of varying difficulty to implement
    - » Also, some people pursued a design not suggested by the trace
- Slight modification of typical "write a synchronization object" exam question
  - This wasn't too bad *for one design*
  - The problem can be solved with two short loops in lock() and one short loop in unlock_all()

Q4 – “Multi-lock”

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!)
- See course staff about any general conceptual problems revealed by this specific exam question

Q4 – "Multi-lock"

General synchronization calamities

- Deadlock (always a problem, deeply ironic here)
- Progress failures (e.g., losing threads)
  - Unlocking not-held locks
- Mutual exclusion failures
- 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

Q4 – "Multi-lock"

Things to watch out for

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

Other general advice

- It's a good idea to trace through your code and make sure that at least the simplest cases work without threads getting stuck
  - Simplest case: one thread locks and unlocks
  - Second-simplest case: one thread locks, a second thread tries, the first thread unlocks
  - Also any trace provided in the problem statement

Q4 – “Multi-lock”

Outcome

- ~15% of the class had a feasible approach and reasonable code
- ~20% more “numerically passed”
- ~30% “suffered severe damage”

Q4 – “Multi-lock”

Outcome

- ~15% of the class had a feasible approach and reasonable code
- ~20% more “numerically passed”
- ~30% “suffered severe damage”
  - Interestingly, 70% of the “severe damage” category did very well on Q3

Q5 – Nuts & Bolts: Register Dumps

Question goals

- 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

Part A

- This really should jump out at you
- If not, try to figure out why it didn't
  - There were some “not so great” loop solutions and one “really alarming” loop solution

Part B

- “The problem” involves *comparing* registers

Q5 – Nuts & Bolts: Register Dumps

Outcomes

- Around 75% of class got 8/10 or better
- Scores under 7 suggest a debugging chat with an instructor

Breakdown

```
90% = 58.5    4 students (57.0 and up)
80% = 52.0    4 students (51.5 and up)
70% = 45.5   17 students (45.0 and up)
60% = 39.0    5 students
50% = 32.5    4 students (31.0 and up)
40% = 26.0    0 students
<40%          2 students
```

Comparison/calibration

- These scores are low – maybe 5% too low?
- Some adjustment is likely

Implications

Special note for F'17 exam

- Look at score for Q3 + Q4
  - If it was above 25/35 that is better than if not
  - If it was below 20/35 that is concerning

Score below 45?

- 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: draft plan, see instructor

Implications

Score below 35?

- 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)
- 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!**