15-410

"My other car is a cdr" -- Unknown

Exam #1
Oct. 17, 2018

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

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

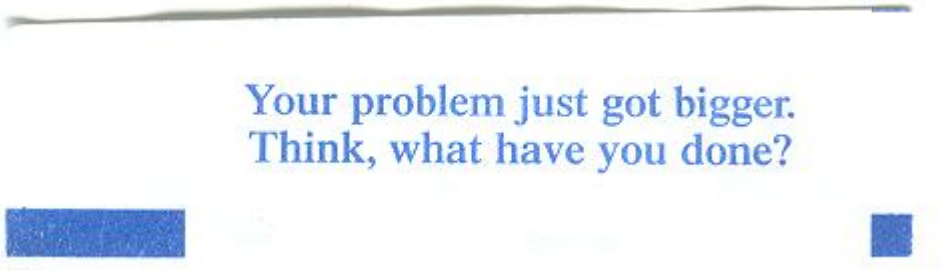

Image credit: Kartik Subramanian

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 – Decision Table

Purpose: demonstrate grasp of a design tool

- Hopefully P2 involved deliberate design
- Hopefully P3 is involving deliberate design
- When you leave here, hopefully you practice deliberate design and record deliberations sometimes

Common issue: core isn't "metrics & values"

- "Pros and cons"
  - It's just too easy to leave things out
- "Evaluate the common case and the rare case"
  - These are ok metrics in some cases, but not the overall approach

Common issue: no "because" step

- It is almost always necessary to resolve a conflict

Q1a – Decision Table

Other issues

- Missing values
- No example decision

Possible 1-point claw-back

- "Try to find a third approach"
  - Good job catching the buried premise!

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

Good news

- Most people identified the suspicious register

Common issues

- Some people didn't explain how that kind of value in that register would lead to trouble
  - Some seemed to suggest that the processor compares two registers and declares a fault based on that
- Sometimes there were issues with reproduction code

Q2 – “Uplock” Starvation

What we were testing

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

Good news

- 2/3 of the class got 7/10 or better

Other news

- 1/3 of the class got 2/10 or below

Largest common issues

- Trace doesn't demonstrate starvation
- Trace can't happen

Others

- Explanation problems, confusing trace, ...
- Repetition isn't made clear

Q3 – Parallel-sort Deadlock

Question goals

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

Good news / bad news

- A/B: ~50% of class
  - Deadlock was fairly simple
- Below C: ~45% of class

Alarming

- Some submissions demonstrated misunderstanding of cvars
  - Allowing this to persist would be unwise

Q3 – Parallel-sort Deadlock

Notes

- The code won't let two threads deadlock (hmm...)
- Some ingredients were mis-attributed
  - "Mutual exclusion" does exist, but not because the code contains `mutex_lock()`/`mutex_unlock()`
  - Other mis-attributions were observed
- A simple fix does exist

Q4 – Targetable condition variables

Question goal

- Slight modification of typical "write a synchronization object" exam question
- This was neither "easy" nor "killer"

Somewhat alarming

- Holding a mutex across `cond_wait()` is "at least quite dubious in general"
  - It was also a fertile source of deadlock in this problem
- The sample trace had two threads...
  - Solutions that solved exactly the two-thread case were somewhat alarming (see also Q3)

Less alarming but common

- Excessive use of the "world mutex" (passed into the tcv) can result in thread loss

Q4 – Targetable 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!)
- See course staff about any general conceptual problems revealed by this specific exam question

Q4 – Targetable 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!)
- See course staff about any general conceptual problems revealed by this specific exam question

Alarming things

- 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 – Targetable 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 – Targetable condition variables

Outcome

- ~30% of the class "did ok": scored 70% or better
- ~45% of the class scored 50% or below

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 – `Nuts & Bolts`: Stack Copying

Question goals

- Test understanding of x86-32 Linux/Pebbles stack
- Test higher-level implications of stack contents
  - This is relevant to P3! Be careful out there!

A frequent conceptual issue

- Return address is address of instruction *after* `CALL`
  - True across architectures (even with fixed-size instructions)

Breakdown

```
90% = 63.0    2 students
80% = 56.0    3 students
70% = 49.0    3 students
60% = 42.0    6 students
50% = 35.0    7 students
<50%          4 students
```

Comparison

- Median grade was 61%, so this wasn't an easy exam

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 plan, see instructor

Implications

Score below 36?

- 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 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!**