“My other car is a cdr” -- Unknown

Exam #1
Mar. 2, 2020

Dave Eckhardt
Brian Railing
Dave O'Hallaron
Synchronization

Checkpoint schedule

- Wednesday during class time
- Meet in Wean 5207
  - If your group number ends with
    - 0-2 try to arrive 5 minutes early
    - 3-5 arrive at 10:42:30
    - 6-9 arrive at 10:59:27

- Preparation
  - Your kernel should be in mygroup/p3ck1
  - It should load one program, enter user space, gettid()
    - Ideally lprintf() the result of gettid()
  - We will ask you to load & run a test program we will name
  - Explain which parts are “real”, which are “demo quality”
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
  - Don't forget about CC=clang / CC=clangalyzer
- Running your code on the crash box may be useful
  - But if you aren't doing it fairly regularly, the first “release” may take a long time
Synchronization

Google “Summer of Code”
- [http://code.google.com/soc/](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.

...though it might instead indicate a complex subtlety...
  - ...which we believe will benefit from personal counseling, not just a brief note, to clear up.

“See Instructor”...
  - ...means it is probably a good idea to see an instructor...
  - ...it does not imply disaster.
“Low Exam-Score Syndrome”

What if my score is really low????

- It is frequently possible to do *dramatically* better on the final exam
- Specific suggestions later
Outline

Question 1
Question 2
Question 3
Question 4
Question 5
Q1a – “I would like to assume…”

Basic idea: cost-benefit analysis

- What might you *gain* by assuming X?
  - Is it really a noticeable gain?
- What might you *lose* by assuming X?
  - If \(^{!}X\) is wildly unlikely and easy to detect, then maybe the loss is “once in a long while I need to apologize and nobody will be mad”
  - If \(^{!}X\) is plausible and would lead to disaster, then assuming X will plausibly lead to disaster

As system designers:

- You will need to “bake assumptions into your design”
- You should give real thought to which assumptions to “bake in”
- This pattern represents the most-basic “real thought”
Q1b – M:N Threading

Question goal
- Check your familiarity with what it is
- Check your design sense of when it might be useful

Common issues
- General vagueness (e.g., “More efficient”)
- Insufficient differentiation from N:1 (or from 1:1)
- Frequently missed
  - M:N allows parallelism (N:1 doesn't)
Q1 – Overall

Scores
  - ~2/3 of the class scored 8/10 or better
Q2 – Critical-section protocol

What we were testing
- Find a race condition (important skill)
- Write a convincing trace (demonstrates understanding)

Good news
- ~70% scored 8/10 or better

Minor issues
- Being unclear about initial value of avail
- Omitting too many lines of trace (e.g., conditional checks)

Noticeable
- Confusing bounded waiting with progress

Alarming issues
- Misconceptions about how cvars work
- Trace can't happen
Q3 – “Dead rock”

Question goals

- Diagnose a deadlock situation, based on deadlock principles
- Show a trace
- Evaluate a solution
Q3 – “Dead rock”

Observations

- Staring at code (or a description) and tracing through random paths *may* find a deadlock, but it is often very time-consuming.
- It is usually quicker to find a deadlock by focusing on parts of the code that embody deadlock ingredients:
  - Hold&wait is a good thing to look for
    - If you find a couple, maybe there is a cycle
  - If you can't find hold&wait, find waits; check each for possible holding
    - Holding an object
    - Or holding a condition: “When you increment your counter, I will increment mine!”
- Once you have the *end* of the trace it is often easy to write the beginning.
Q3 – “Dead rock”

Observations

- Showing circular wait, by itself, is not enough to show a deadlock
  - Some other thread may be pre-ordained to release key resources
Q3 – “Dead rock”

Part A
- Some people missed a sequence
  - Including somebody with a username containing '0'

Part B
- Does the described protocol allow one thread to hold some X's while wanting some Y's, and also allow another thread to hold some Y's while wanting some X's?
  - If not, circular waiting might be impossible
  - If so, you might be half-way to a trace

Part C
- Imposing a total order is not likely to remove hold&wait
  - It is much more likely to remove circular waiting
Q3 – “Dead rock”

Specific issues

- Missing process/resource graphs
- Traces with long extraneous parts
- Knowing ingredients but not finding a trace

Scores

- ~40% scored 13/15 (86%) or better
- ~40% scored below 9/15 (60%) or worse
- So there was a diversity of scores
Q4 – Testing cvars

Question goal

- Atypical variant of typical “write a synchronization object” exam question
  - Make sure you can block and unblock threads without things going wrong due to race conditions
- This was a hard question
  - Eckhardt's rush-job solution scored 75% when the TAs got hold of it
  - Even with more time, breaking 90% wasn't going to happen

Hint

- If cvars are broken, there are many ways
  - cond_wait() { unlock(); lock(); return; }
  - cond_wait() { unlock(); sleep(33); lock(); return; }
  - cond_wait() { unlock(); while(1) continue; }
- There are also less-deterministic ways to be broken
Q4 – Testing cvars

Decoder ring (aka detailed hints)

- “Actual block” = tester verified that scheduler believes the threads are actually really truly blocked (this was hard)
- “Early 1st signal” = tester didn't take time to be pretty sure that both wait() have begun before first signal() starts
- “False start” = tester doesn't detect if wait() doesn't actually wait before sending the first signal()
- “Misses double wakeup” = tester doesn't detect if one signal() wakes two threads
- “Liveness” = tester doesn't check both threads run after second signal()
- “Disorder” = tester doesn't check threads ran in the right order
- “Hang” = test can hang without printing a verdict

Points may be -2 or -1
Q4 – Testing cvars

Outcome

- ~8% scored 16/20 (80%) or better
- ~20% scored 14/20 (70%) or better
- ~36% scored 10/20 (50%) or worse
  - “Severe tire damage” group is typically ~30% of class

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 – Stack Picture

Outcome

- ~40% scored 9/10 or better
- ~30% scored below 6/10
## Breakdown

<table>
<thead>
<tr>
<th>Percentage</th>
<th>Score</th>
<th>Students</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>90%</td>
<td>58.5</td>
<td>6</td>
<td>(57.0 and up)</td>
</tr>
<tr>
<td>80%</td>
<td>52.0</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>70%</td>
<td>45.5</td>
<td>24</td>
<td>(45.0 and up)</td>
</tr>
<tr>
<td>60%</td>
<td>39.0</td>
<td>14</td>
<td></td>
</tr>
<tr>
<td>50%</td>
<td>32.5</td>
<td>16</td>
<td>(29.0 and up)</td>
</tr>
<tr>
<td>40%</td>
<td>26.0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>&lt;40%</td>
<td></td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

### Comparison/calibration

- These scores are low – maybe 5% too low?
- A bit of adjustment is plausible/likely
Implications

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:
    » Identify causes, draft a plan, see instructor
Implications

Score below 39?

- Something went *noticeably* wrong
  - It's *important* to figure out what!
- Beware of “triple whammy”
  - Low score on *three* questions
    - Generally Q3, Q4, Q5
- Passing the final exam could be a challenge
- *Passing the class may not be possible!*
  - To pass the class you must demonstrate proficiency on exams (not just project grades)
- Try to identify causes, draft a plan, see instructor
Action plan

Please follow steps in order:

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

Please follow steps in order:

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

Please avoid:

- “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 not known by us
Action plan

Please follow steps in order:

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

Please avoid:

- “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 not known by us

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