15-410

"My other car is a cdr" -- Unknown

Exam #1
Mar. 11, 2024

Dave Eckhardt

Synchronization

Checkpoint schedule (NOTE NEW HASH FUNCTION)

- Friday during class time
- Meet in Wean 5207
  - If your group number *ends* with
    - 0-2 try to arrive 10:55-11:00 (5 minutes early)
    - 3-5 arrive at 11:13:17
    - 6-9 arrive at 11:31:19
- Preparation
  - Your kernel should be in mygroup/p3ck2
  - We are expecting everybody (even if not quite done)
    - Unless you notify us by noon on Thursday

Synchronization

Checkpoint 2 - alerts

- Reminder: context switch ≠ timer interrupt!
  - Timer interrupt is a *special case*
  - Some timer interrupts will *not* cause context switch
    - Really!
  - Most context-switch invocations will have nothing to do with the timer
    - Really!
- 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!

- This your approximately-mid-semester reminder about the book report assignment

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 50% 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
  - Using a variety of compilers is likely to expose issues
- 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/
- 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"?

A Word on the Final Exam

Disclaimer

- Past performance is not a guarantee of future results

The class 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 (~85 points, ~6 questions)

Thanks for Avoiding Faint Pencil!

It wasn't a problem on the mid-term

- Let's keep it that way for the final exam!

“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

Q1 – Short Answer

Two parts

- “Three mutex assumptions”
- Register dump

Three Mutex Assumptions

High-level principle

- Different locks for different situations
  - Contention expectation
  - What is being protected
  - Need for waiting/handoff

Mutex is one *specific* kind of lock

- Use in the right situation
- Don't use in other situations

Three Mutex Assumptions

High-level principle

- Different locks for different situations
  - Contention expectation
  - What is being protected
  - Need for waiting/handoff

Mutex is one *specific* kind of lock

- Use in the right situation
- Don't use in other situations

Lost on this question?

- We discussed in *two* lectures (Jan. 31, Feb. 2)
- Plug: maybe review some of the “odd” lectures
  - Debugging, Questions
  - #define, #include
  - We expect you to know *and apply* all of this material

Three Mutex Assumptions

Doctrine

- *Short* sequence
  - Not true of all locks
- Must avoid “interfering executions”
  - This one is true of all locks
- Contention *expected to be rare*
  - Not true of all locks

Who can think of counter-examples?

Three Mutex Assumptions

Doctrine

- *Short* sequence
  - Not true of all locks – rwlock is a counter-example!
- Must avoid “interfering executions”
  - This one is true of all locks
- Contention *expected to be rare*
  - Not true of all locks – “barrier” is a counter-example!
- Be able to say why each matters

Emergency partial credit

- Three critical-section requirements

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

Pretty clear

- One of the registers has a very-wrong value

But...

- Actually, *two* registers have values that are wrong
- What sort of mistake could have *both* negative effects?

Q1b – Register Dump

Common issues

- It is necessary to say *why/how* a wrong register leads to an exception
  - “%xxx should point at Y, not at Z” is not a fault type in this situation
  - “Page fault” is actually fairly *un*likely
  - Some faults not really possible in P2/P3 were claimed

Q1 – Results

Scores

- ~50% of the class scored 7/10 or above (good)
- ~10% of the class scored *below* 6/10 (… … ...)

Q2 – Going Out of Business

What we were testing

- Ability to find comon synchronization problems
- Ability to support a diagnosis with a clear trace

Odd features of the problem

- It was based on code discussed in lecture
- Part of the problem was based on a course-staff bug found by a student (Mohammed Al-Jawaheri)

Almost all traces got full credit

- Thus, prudent to follow up on any point deductions

Q2 – Going Out of Business

Hints

- One bug is fixed by changing *less than one line of code*
- The other bug is fixed with a change that is also very small and straightforward
- Guessing at solutions (without having “sad case” traces) is not likely to be fruitful

Q2 – Going Out of Business

Warnings

- It is unwise to discuss hypothetical probems in code we don't show
  - If we show code and say there is a problem, *we believe there is a problem in the code we are showing*
  - Example: “If no work items are ever enqueued, all threads will be stuck forever”
- Moving signal() inside the mutex *does not* ensure perfect fairness of dequeue() results
  - It is always possible that some unrelated thread got to the mutex first
  - If this is not clear, please see somebody in office hours

Q2 – Going Out of Business

Outcomes

- ~60% had 13/15 or better
- ~12% had 10/15 or below

If you had trouble with Q2...

- ...Please figure out why, and how to practice.
  - This is core material!

Q3 – Kitchen Robots

Question goals (lots!!)

- Apply various deadlock concepts and skills
- Show a trace
- Pick a (correct) prevention rule
- Describe an avoidance approach
- Compare the prevention approach and the deadlock approach via a design matrix

Q3 – Kitchen Robots

Question goals and and result summaries

- Apply various deadlock concepts and skills
  - Show understanding of *detection* vs. *prevention* vs. *avoidance*
    - “Reboot the system” was not tested
- Show a trace
- Pick a (correct) prevention rule
  - Two were mentioned frequently, but only one prevents deadlock
- Describe an avoidance approach
  - Almost nobody described “textbook” avoidance approach for single-instance resources
  - Many solutions were application-specific and creative
    - Most of those were close but not quite right
- Compare the prevention approach and the deadlock approach via a design matrix

Q3 – Kitchen Robots

Most grade variance came in avoidance

- Concerning avoidance attempts
  - “Ban all cycles”
    - Not really avoidance, also doesn't really work
  - Conceptually unclear text about safe sequences
  - “Ban any recipe using any station more than once”
    - Not really avoidance, also important foods are unavailable
  - Things that clearly deadlock

Design-matrix scores were generally high

  - Non-high scores should be looked into

Q3 – Kitchen Robots

Outcomes

- ~75% of class scored 16/20 or better

Q4 – Trio Matcher

Question goal

- Slight modification of typical “write a synchronization object” exam question
- Neither super-easy nor super-hard
    - Scores below 70% (14/20) are concerning

Q4 – Trio Matcher

Interesting question feature

- There are mutiple good solutions (three or four)
  - Single-mutex vs. multiple-semaphore
  - Queue vs. array vs. fields
  - If you solved it one way, maybe try again a different way?

Things to watch out for

- “Slarching” aka “clobbering”
  - In the example code, after each match a thread is likely to match again immediately; this must work correctly
- “Evil threads” resulting in thread sadness

Q4 – Trio Matcher

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 – Trio Matcher

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
    - If the question provides example traces, it's prudent to check that your code does the right thing for those traces!

Other things to watch out for

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

Q4 – Trio Matcher

Outcomes

- ~20% of the class scored 20/20 (great!)
- ~30% of the class scored 18/20 (“A”)
  - Question is arguably “not super hard”
- ~30% of the class “did not do ok” (under 60%)
  - These outcomes are concerning

Q5 – Nuts & Bolts

Quick (5-point) question

- What's in a P1 “interrupt frame” – and why?

Common issue

- Providing a rationale for %eflags
  - Some things in %eflags change *a lot*, and those values must be correct!

Outcomes

- Many 4/5 and 5/5 scores
- But also many 2/5 scores
  - And some lower!

Breakdown

90% = 63.0	5	students
80% = 56.0	12	students
70% = 49.0	11	students
60% = 42.0	9	students
50% = 35.0	2	students
<50%	2	students

Comparison

- Median score was 54/70 (77%)
  - This is not low

Implications

Score below 52?

- 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

Implications

Score below 52?

- 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 46?

- Something went *noticeably* wrong
  - It's *important* to figure out what!
- Passing the final exam could be a challenge
- *Passing the class may be at risk!*
  - To pass the class you must demonstrate proficiency on exams (not just project grades)
  - We don't know the format of the final exam yet, but a strong grasp of key concepts, especially concurrency, is important

Implications

Score below 46?

- Something went *noticeably* wrong
  - It's *important* to figure out what!
- Passing the final exam could be a challenge
- *Passing the class may be at risk!*
  - To pass the class you must demonstrate proficiency on exams (not just project grades)
  - We don't know the format of the final exam yet, but a strong grasp of key concepts, especially concurrency, is important
- Try to identify causes, draft a plan, see instructor
  - Good news: explicit, actionable plans usually work well

Action plan

Please follow steps in order:

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

Action plan

Please follow steps in order:

1. Identify 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!*
  - The “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!*
  - The "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