15-410

"My other car is a cdr" -- Unknown

Exam #1
Oct. 18, 2021

Dave Eckhardt

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 11:27:30
    - » 6-9 arrive at 11:44:27
- 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!

- 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 40% 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

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

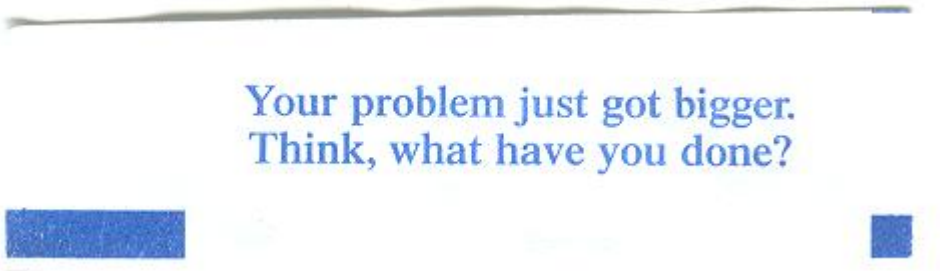

Image credit: Kartik Subramanian

A Note for Posterity

The F'21 mid-term exam occurred during COVID-19

This was *semi*-typical exam

- **Maybe one question shorter than typical**

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 (~85 points, ~6 questions)

Please Avoid Faint Pencil!

Some people wrote using pencil

- Some wrote with *very faint* pencil!
- Please do not do this on the final exam!
  - In any class!

“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

Q1 – CMU-calendar design decision

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 aren't doing design matrices you might not be doing design
  - Submitting a couple with your P3 may improve your P3!

Q1 – CMU-calendar design decision

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 aren't doing design matrices you might not be doing design
  - Submitting a couple with your P3 may improve your P3!

If you were lost on this question...

- We had a lecture on this topic (September 3)
- Other "odd" lectures to possibly review
  - Debugging, Errors
  - #define, #include
  - We expect you to know *and apply* all of this material

Q1 – CMU-calendar design decision

The chart format is your friend

- Without a chart it is too easy to forget to compare the same factors across all proposals
  - "Pros and cons" faces this danger
  - A feature matrix without metric names plus values *that match the names* faces this danger

Q1 – CMU-calendar design decision

The chart format is your friend

- Without a chart it is too easy to forget to compare the same factors across all proposals
  - "Pros and cons" faces this danger
  - A feature matrix without metric names plus values *that match the names* faces this danger

Look for third/fourth options!

- Conflict between desirable factors in two proposals can inspire a new proposal
- "Pros and cons" hides these conflicts

Q1 – CMU-calendar design decision

The chart format is your friend

- Without a chart it is too easy to forget to compare the same factors across all proposals
  - "Pros and cons" faces this danger
  - A feature matrix without metric names plus values *that match the names* faces this danger

Look for third/fourth options!

- Conflict between desirable factors in two proposals can inspire a new proposal
- "Pros and cons" hides these conflicts

Overall, "pros and cons" is an inferior approach

- You may be accustomed to it, but an upgrade may be wise

Q1 – CMU-calendar design decision

Use numbers when possible

- Avoid "pseudo-booleans"
  - Avoid: "good performance" with values "yes" and "no"
  - Prefer: "run time" with values "O(N)" and "O(logN)"

Q1 – CMU-calendar design decision

Use numbers when possible

- Avoid "pseudo-booleans"
  - Avoid: "good performance" with values "yes" and "no"
  - Prefer: "run time" with values "O(N)" and "O(logN)"

Be specific when possible

- Avoid: "freeing of resources"
- Prefer: "freeing of thread control block"

Q1 – CMU-calendar design decision

Use numbers when possible

- Avoid "pseudo-booleans"
  - Avoid: "good performance" with values "yes" and "no"
  - Prefer: "run time" with values "O(N)" and "O(logN)"

Be specific when possible

- Avoid: "freeing of resources"
- Prefer: "freeing of thread control block"

Be wary of "dangerous metrics"

- "Does it work?" / "Fundamental correctness"
  - Documenting non-working proposals can be useful in some situations
  - But two non-working proposals plus one working proposal probably means that design work should continue

Q1 – CMU-calendar design decision

Avoid confusing input variables vs. output variables

- If “14-week semester” and “15-week semester” are options...
  - “Weeks” is not a metric with values “14” and “15”

Q1 – CMU-calendar design decision

Avoid confusing input variables vs. output variables

- If “14-week semester” and “15-week semester” are options...
  - “Weeks” is not a metric with values “14” and “15”

Avoid “essay in each box” syndrome

- What goes in a “value cell” should be the value of an observable variable, not an argument

Q1 – CMU-calendar design decision

Avoid confusing input variables vs. output variables

- If "14-week semester" and "15-week semester" are options...
  - "Weeks" is not a metric with values "14" and "15"

Avoid "essay in each box" syndrome

- What goes in a "value cell" should be the value of an observable variable, not an argument
- Instead of *explanatory* words in a value cell...
  - Try to upgrade the metric description
  - Try to upgrade the option description in the column heading
- Instead of *justification* words in a value cell, put those words in the conclusion/choice area

Q1 – CMU-calendar design decision

Avoid confusing input variables vs. output variables

- If "14-week semester" and "15-week semester" are options...
  - "Weeks" is not a metric with values "14" and "15"

Avoid "essay in each box" syndrome

- What goes in a "value cell" should be the value of an observable variable, not an argument
- Instead of *explanatory* words in a value cell...
  - Try to upgrade the metric description
  - Try to upgrade the option description in the column heading
- Instead of *justification* words in a value cell, put those words in the conclusion/choice area
- The goal for the matrix is quick comparison among options, not explaining options

Q1 – CMU-calendar design decision

Conclusion form

- Avoid
  - Discussing metrics and values in the conclusion that aren't in the matrix
  - We picked X.
  - We picked X because it was the only correct solution.
- Prefer
  - We picked X because value V1 for M1 is unacceptable for the expected workload.
  - We picked X because (M1, V1) is more important than (M2, V2).

Q1 – Overall

Scores

- 2/3 of the class scored 8/10 or better
- Only one submission received 10/10

Q1 – Overall

Scores

- 2/3 of the class scored 8/10 or better
- Only one submission received 10/10

Sentiment

- 2:1 in favor of 15-week semesters!
- Perhaps let your student-government representatives know your thoughts

Q2 – Barrier Problem

What we were testing

- Find a synchronization botch (important skill)
- Write a convincing trace (demonstrates understanding)

Good news

- Most students (~75%) scored 5/7 (71%) or better on “find problem and write trace”

Less-good news

- 2/3 did ok on fixing the problem

Q2 – Barrier Problem

Minor issues

- Omitting too many lines of trace (e.g., conditional checks)

Noticeable issues

- Not explicitly naming an observed problem
- Not giving a clear and compelling trace
  - Missing state updates or control-flow choices

Semantic issues

- Spinning for a long time is not a good solution!
  - For this question, elsewhere in this class, or elsewhere
- Mutexes are supposed to be used in specific circumstances
- Some proposed solutions don't work
  - Generally, having threads “stall” before the cond_wait() won't work

Q3 – Grading Deadlock

What we were testing

- Find a deadlock (important skill)
- Write a convincing trace (demonstrates understanding)
- Fix a deadlock (and argue that the fix works)

Good news

- ~50% scored 13/15 or better
- ~75% scored 10/15 or better
- So lots of people can identify and trace a fairly typical deadlock

Q3 – Grading Deadlock

Noticeable issues

- Omitting too many lines of trace
  - A very terse trace might summarize very-different executions (one deadlock, one not)
    - » That does not clearly demonstrate understanding
- Trace does not follow assumptions or state
  - Sometimes happens when trace is missing too many details

Q3 – Grading Deadlock

Common issues

- "Global mutex" is an *emergency* solution to deadlock
  - Not a good solution
- Memorizing the four deadlock ingredients probably is a good idea
  - If something is a fix, that thing should clearly ensure the absence of one of the ingredients – it should be easy to say which and how
- Generally, avoid traces with multiple operations in a single row
  - Unless clarity is genuinely improved
- Not all "tabular traces" were tabular
  - A paragraph isn't really a trace

Q4 – “Message Buffer”

Question goals

- Variant of typical “write a synchronization object” exam question
- This one was probably “typical” rather than “easy” or “hard”

Key design areas

- How to block excess senders?
- How to protect buffer slots?
- How to track uncollected messages?

Q4 – “Message Buffer”

Common issues

- Most solutions protected all slots with a single mutex
  - There can be *many* slots! What are the mutex rules?
  - There are other options!

Q4 – “Message Buffer”

Common issues

- Most solutions protected all slots with a single mutex
  - There can be *many* slots! What are the mutex rules?
  - There are other options!
- Whenever possible, consider multiple options
  - cvars aren't the only way to block threads
  - singly-linked lists aren't the only way to achieve sequential access

Q4 – “Message Buffer”

Common issues

- Most solutions protected all slots with a single mutex
  - There can be *many* slots! What are the mutex rules?
  - There are other options!
- Whenever possible, consider multiple options
  - cvars aren't the only way to block threads
  - singly-linked lists aren't the only way to achieve sequential access
- We said “ok to assume malloc() doesn't fail on an exam”
  - That is a structurally-more-reasonable assumption for mb_init() than for mb_anythingelse()!
    - » Please review P2 material on “return values”
    - » P3 faces similar considerations!

Q4 – “Message Buffer”

Common issues

- Most solutions protected all slots with a single mutex
  - There can be *many* slots! What are the mutex rules?
  - There are other options!
- Whenever possible, consider multiple options
  - cvars aren't the only way to block threads
  - singly-linked lists aren't the only way to achieve sequential access
- We said “ok to assume malloc() doesn't fail on an exam”
  - That is a structurally-more-reasonable assumption for mb_init() than for mb_anythingelse()!
    - » Please review P2 material on “return values”
    - » P3 faces similar considerations!
- NULL is a legit void*
  - Sending/receiving it should work

Q4 – “Message Buffer”

General synchronization calamities

- Deadlock
- 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 – “Message Buffer”

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/case 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)

Breakdown

```
90% = 49.5    3 students (49 and up)
80% = 44.0    4 students (44 and up)
70% = 38.5    5 students (39 and up)
60% = 33.0    4 students (36 and up)
50% = 27.5    0 students
<50%          2 students
```

Comparison/calibration

- Scores are a little low for a typical 410 mid-term
  - Low 43%, median 72%, max 92%

Implications

Score below 38?

- 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 38?

- 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 33?

- 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 33?

- 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. 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

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