15-410

"My other car is a cdr" -- Unknown

Exam #1
Mar. 9, 2026

Dave Eckhardt

Babu Pillai

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!

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...

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

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

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

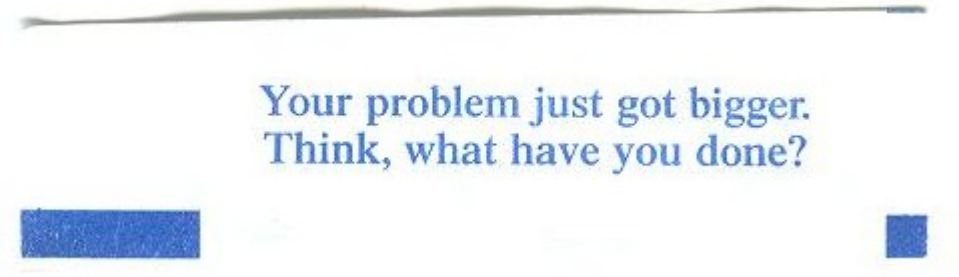

Image credit: Kartik Subramanian

A Word on the Final Exam

Disclaimer

- Past performance is not a guarantee of future results

The 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

- Maybe more design questions
- Some things you won't have implemented (text useful!!)
- Still 3 hours, but may be more stuff (~85 points, ~6 questions)

A Word on the Final Exam

Special note for Spring 2026

- This race question might have been a bit harder than others
- This synch-object question was arguably easier than others
  - So it's possible that the final exam might have one that's a bit harder

Thanks for Avoiding Faint Pencil!

It wasn't a problem on the mid-term

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

New Request for Final Exam

Please don't augment your answers with red pen

- **Green would be ok**

“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 necessarily 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
  - Please execute those instructions in order

Outline

Question 1

Question 2

Question 3

Question 4

Question 5

Q1 – Short Answer

Three parts

- Three kinds of error
- Two kinds of thread cancellation
- Starvation

“And Now, A Word From Our Sponsor...”

Three Kinds of Error

An actionable *robustness practice*

- Hopefully P2 involved careful error handling
- Hopefully P3 will involve careful error handling
- "Robust code is *structurally different* than fragile code"
- P3 requires not just code but *structurally non-fragile code*.

If you were lost on this question...

- We had a lecture on this topic (February 4)
- Other "odd" lectures to possibly review
  - Debugging, Questions
  - #define, #include
  - We expect you to know *and apply* all of this material

Three Kinds of Error

Official trichotomy

- Resolvable – so resolve it
- Reportable – so report it
- "Rebootable"(?)
  - Involve the developer, because the program is *broken*
  - Stop the program before propagating lies

Not exactly in the same space

- "Rewritable"(??) - "I shouldn't have written this code, so I need to re-design and rewrite"
- That was generally accepted anyway

Q1a/b – Three Kinds of Error

Purpose

- Demonstrate understanding of the three kinds of error in 410 orthodoxy

Selected Issues

- Tying *frequency of occurrence* to *type of error*
  - We *hope* that worse errors are rarer...
  - But what makes a "really bad" error really bad is not whether or not it's rare
- Troublesome descriptions

Q1b – Thread Cancellation

Key idea: two kinds, both are troublesome

- Many people did reasonably well

Common issues

- Describing thread exit due to something the thread did
  - That's basically the opposite of cancellation
- Describing something about threads that isn't exiting or cancellation

Q1c – Starvation

Key elements

- Unbounded time
- Thread can't get the resources it needs
- Because other threads are getting the resources they need instead

Q1 – Results

Most people did well on (a) and (c)

Scores

- **~85% of the class scored 7/10 or above (good)**
- **~5% of the class scored *below* 5/10**

Q2 – Rendezvous

What we were testing

- **Ability to find *and show* race conditions**

Q2 – Rendezvous

What we were testing

- Ability to find *and show* race conditions

Conceptual

- `cond_wait()` drops *and reacquires* the world mutex
- Assuming that "the problem" is bad behavior inside invisible cvar code contradicts the problem statement

Trace issues

- It is imprudent to start a trace in the middle of a scenario
- Traces should show all important steps (e.g., synch operations)
- `cond_signal()` doesn't awaken a thread waiting on a different cvar
- `cond_wait()` inside if() isn't *always* "Paradise Lost"

Q2 – Results

Scores

- 68% of the class got 14/15 or 15/15 (good!)
- ~20% of the class scored *below* 10/15 (10/15 == 60%)

Q3 – Ride Synchronization

Question goal

- Slight modification of typical "write a synchronization object" exam question

Interesting question features

- Can be done with or without semaphores
  - Maybe try it the other way!
- Some typical disaster causes are missing (phase errors)
  - This arguably was not a particularly-difficult question

Q3 – Ride Synchronization

Things to watch out for

- Both `enter()` and `leave()` have a "simple" scenario and a "less simple" scenario
  - Accounting is probably different for the two scenarios!
  - So there were two popular "wrong accounting" races
- Mutexes in particular should not span expensive operations
- Solutions should not impose arbitrary limits on the number of priority levels (see problem statement)
- Some solutions got the priority levels reversed

Q3 – Ride Synchronization

General note on blocking

- Threads that can't do productive work should *stop running*
- Once stopped, a thread should remain stopped *until there is a reasonable likelihood that it can do productive work*

Q3 – Ride Synchronization

General note on blocking

- Threads that can't do productive work should *stop running*
- Once stopped, a thread should remain stopped *until there is a reasonable likelihood that it can do productive work*

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

Q3 – Ride Synchronization

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 a 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)

Q3 – Ride Synchronization

Outcomes

- ~10% of the class got 20/20
  - This may be unusually high?
- ~30% of the class scored 16/20 or better (80%+)
  - This band is arguably smaller than usual?
- ~60% of the class "did not do ok" (under 60%)
  - Some answers had multiple serious issues
  - Some answers did not suggest a clear understanding of thread synchronization

Other questions in this category can be harder

- Perhaps a final-exam question might be harder

Q4 – M:N Threads

Concepts being tested

- M:N threads
- What thread-library `thr_xxx()` functions do
  - (What some thread-library *non-*`thr_xxx()` functions do)

Q4 – M:N Threads

Concepts being tested

- M:N threads
- What thread-library `thr_xxx()` functions do
  - (What some thread-library *non-*`thr_xxx()` functions do)

Bad news

- Descriptions of categories resulted in some confusion
  - C#2: *should* switch in some situations but *should not* switch in other situations: circumstantial
  - C#3: could legitimately switch/not-switch by flipping a coin: policy

Q4 – M:N Threads

Concepts being tested

- M:N threads
- What thread-library `thr_xxx()` functions do
  - (What some thread-library *non-*`thr_xxx()` functions do)

Bad news

- Descriptions of categories resulted in some confusion
  - C#2: *should* switch in some situations but *should not* switch in other situations: circumstantial
  - C#3: could legitimately switch/not-switch by flipping a coin: policy

Good news

- There was enough confusion that grading was based mostly on *explanation* for a thr_xxx() function
- Mostly not based on which category it was placed in

Q4 – M:N Threads

Common issues

- `thr_join()` has two *very different* cases!
  - Assertions that it "must block" were problematic
- Suggestions that yield loops are ok where blocking is needed were problematic
- Some structurally-wrong claims were made
  - `thr_init()` is a special case
  - `thr_exit()` is a different special case

Q4 – M:N Threads

Results

- ~40% of the class got 14/15 or 15/15
- ~40% of the class got 12/15 or 13/15
- Due to confusion, grading was fairly relaxed
- Scores below 12 suggest revisiting the concepts

Q5 – Stack Diagram

Concepts tested

- x86-32 stack discipline (as experienced in P0, P1, P2)
- C arrays & pointers & strings
- Where things live

Selected issues

- Missing strings, missing argv[] array
- Confusing argv pointer with argv[] array
- argv[] missing something at the start or end
- argv[] strings in impossible parts of the address space
  - Warning: "All strings are located in _____" is not a rule
- Problems with invocation of `_main()`

Q5 – Results

Overall outcomes

- **~25% got 8/10 or better**
- **Scores under 6/10 (40% of class) suggest misconceptions that are important to quickly address**

Breakdown

```
90% = 63.0    6 students
80% = 56.0    7 students
70% = 49.0   12 students
60% = 42.0   10 students
50% = 35.0    4 students (34 and up)
<50%          2 students
```

Comparison

- Median score was 50/70 (71%)

Implications

Score below 57?

- 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 57?

- 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 *much* better than “I'll try harder”
  - *Strong suggestion*:
    - » Identify causes, draft a plan, see instructor

Implications

Score below 48?

- Something went *noticeably* wrong
  - It's *important* to figure out what!
- Some exams had low Q3+Q5
  - Thus trouble with "high-level" *and* "low-level" material
  - Looking into this is probably prudent
- 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)

Implications

Score below 48?

- Something went *noticeably* wrong
  - It's *important* to figure out what!
- Some exams had low Q3+Q5
  - Thus trouble with "high-level" *and* "low-level" material
  - Looking into this is probably prudent
- 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)
- 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. 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

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