15-410

"My other car is a cdr" -- Unknown

Exam #1
Oct. 20, 2015

Dave Eckhardt

Synchronization

Checkpoint 2 – Wednesday, in 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

Asking for trouble?

- If your code isn't in your 410 AFS space every day, you are asking for trouble
  - Roughly 2/3 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
- If you aren't using source control, that is probably a mistake
- GitHub sometimes goes down!
  - S'13: on P4 hand-in day (really!)

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

Book report!

- **Hey, “Mid-Semester Break” is just around the corner!**

Synchronization

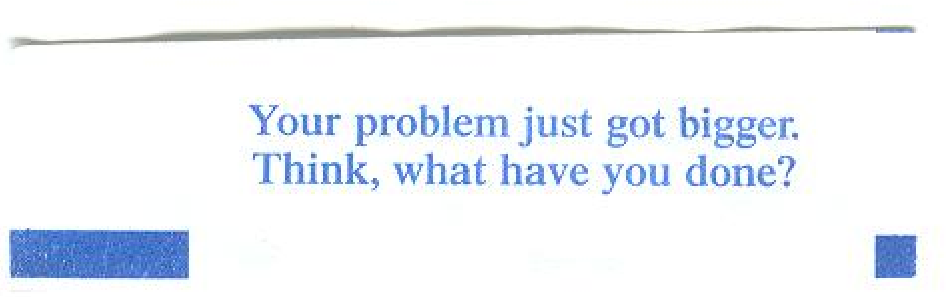
Debugging advice

- Once as I was buying lunch I received a fortune

Synchronization

Debugging advice

- Once as I was buying lunch I received a fortune



Your problem just got bigger.
Think, what have you done?

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 paper 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 – .h file contents

Expectations

- Yes
  - Header guards
  - Public function declarations
  - Public constant declarations
  - Public type declarations
- No
  - File-private data items (static globals)
  - File-private function declarations
  - Code
- Maybe (e.g., “public .h vs. internal .h”)
  - Module-wide data
  - Module-internal functions
  - Module-internal types
  - “Definitions of macros” – what *kind*?
  - “Definitions of functions” – what *kind*?

Q1a – .h file contents

Grading

- Based on clarity of answers
- Lots of high scores

Please keep the key principle in mind

- Visibility: “Who needs to see this?”
  - Things don't belong in places due to shape; they belong in places due to function
  - Some macros are file-private! Who needs to see them?
  - Some functions are file-private! Who needs to see them?

Q1b – “Race-Condition Ingredients”

Expected answer (from slides)

- Concurrent activities
- Different interleavings cause different outcomes
- Interleavings are not controlled

Q1b – “Race-Condition Ingredients”

Expected answer (from slides)

- Concurrent activities
- Different interleavings cause different outcomes
- Interleavings are not controlled

Alternate answer (from some other class?)

- Multiple threads share data items for reading
- At least one thread writes
- The data items aren't protected by a single lock

Q1b – “Race-Condition Ingredients”

Expected answer (from slides)

- Concurrent activities
- Different interleavings cause different outcomes
- Interleavings are not controlled

Alternate answer (from some other class?)

- Multiple threads share data items for reading
- At least one thread writes
- The data items aren't protected by a single lock

The “alternate” answer is ok

- But look!

Q1b – "Race-Condition Ingredients"

Expected	Alternate
Concurrent activities	Multiple threads
Interleavings change outcomes	Somebody writes
Interleavings not controlled	Data not covered by one lock

Observations

- "Alternate answer" is the definition of a "data race"
- Be careful out there!
  - Some solutions described the same ingredient twice but then forgot one (usually: lock)

Q2 – Deadlock

Parts of the problem

- Find the deadlock
- Show a trace
- How to fix it?

Q2 – Deadlock

Parts of the problem

- Find the deadlock
  - Basically everybody found “the” deadlock (some found two)
- Show a trace
  - This part was more trouble
  - This part is important
- How to fix it?
  - More trouble was encountered here
  - ***Especially since an un-detected question bug meant that the elegant expected solution was wrong, thus encouraging “global mutex” solutions. This is not the best exam question to practice on, at least not the “fix it” part.***

Q2 – Deadlock

Common issues

- *Each* deadlock must exhibit *all four* ingredients
    - “I found hold&wait so there's a deadlock!” isn't true
    - Not even “I found circular wait!”

Q2 – Deadlock

Common issues

- *Each* deadlock must exhibit *all four* ingredients
    - "I found hold&wait so there's a deadlock!" isn't true
    - Not even "I found circular wait!"
- "I can solve the deadlock with a single global lock for everything!"
    - Solving a concurrency problem by deleting all the concurrency will make your boss angry.
    - "Global mutex" is kind of like "global variable" – a vigorous defense is required
    - In this problem, "just impose a lock order" usually meant "just impose a global mutex"

Q2 – Deadlock

Common issues

- *Each* deadlock must exhibit *all four* ingredients
  - “I found hold&wait so there's a deadlock!” isn't true
  - Not even “I found circular wait!”
- “I can solve the deadlock with a single global lock for everything!”
  - Solving a concurrency problem by deleting all the concurrency will make your boss angry.
  - “Global mutex” is kind of like “global variable” – a vigorous defense is required
  - In this problem, “just impose a lock order” usually meant “just impose a global mutex”
- “Make X preemptible” is usually not feasible
- “Add an infinite number of X's” is basically never feasible

Q2 – Deadlock

Common issues

- *Each* deadlock must exhibit *all four* ingredients
  - “I found hold&wait so there's a deadlock!” isn't true
  - Not even “I found circular wait!”
- “I can solve the deadlock with a single global lock for everything!”
  - Solving a concurrency problem by deleting all the concurrency will make your boss angry.
  - “Global mutex” is kind of like “global variable” – a vigorous defense is required
  - In this problem, “just impose a lock order” usually meant “just impose a global mutex”
- “Make X preemptible” is usually not feasible
- “Add an infinite number of X's” is basically never feasible
  - “Add a *carefully chosen* number of X's” may be ok

Q2 – Deadlock

Advice

- Being able to show a problem clearly and convincingly is an important skill
- Being able to state a problem precisely (e.g., with a formula) is often a key step toward solving it well

Q2 – Deadlock

Advice

- Being able to show a problem clearly and convincingly is an important skill
- Being able to state a problem precisely (e.g., with a formula) is often a key step toward solving it well

Warnings

- "Global mutex" is *a* solution
  - *Every* concurrency problem can be solved by a global mutex
  - It is never a *high-quality* solution

Q3 – “LIFO condition variables”

Question goal

- Slight modification of typical “write a synchronization object” exam question

General conceptual problems

- “x() takes a pointer” does *not* mean “x() must call malloc()”
- Not all byte arrays are null-terminated
  - strcpy() vs. memcpy()
- 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!)
- Everything must be initialized and destroyed
- See course staff about any general conceptual problems revealed by this specific exam

Q3 – "LIFO condition variables"

"Be careful out there"

- Deadlock scenarios
- Memory leaks
- Busy-wait/spin-loop – use an accepted synch object!
- Waking up threads when it really doesn't make sense
  - Use cond_broadcast() rarely – one "ok case" is when the number of threads to awaken is genuinely uncertain

Question-specific conceptual problems

- In condition variables, the "world mutex" is important!
  - *Must* be dropped and re-acquired
  - Timing of drop & acquire iscritical: too early or too late will generally cause a thread to hang
- "Wrong number of awakens"
  - two signal()s awaken one thread
  - signal() causes a broadcast() to skip some threads

Q3 – “LIFO condition variables”

Question-specific practical issues

- It is *possible* to implement a stack out of queues
  - But it's harder to get that right than implementing a stack out of a stack!
- malloc() is not the only way to allocate memory – especially if the amount is small and fixed and the duration is small and fixed!

Q4 – Faulty Mutex

Problem

- Find the problem in the mutex code

Q4 – Faulty Mutex

Problem

- Find the problem in the mutex code

Good news/bad news

- Good news: there were two problems (progress, mutual exclusion)
- Bad news: they were both subtle

Q4 – Faulty Mutex

Problem

- Find the problem in the mutex code

Solution hints

- Hint for progress
    - "`goahead[i] = 1 – goahead[j]`" is not atomic
- Hint for mutual exclusion
    - One thread can unlock "at a bad time"

Q4 – Faulty Mutex

Problem

- Find the problem in the mutex code

Solution hints

- Hint for progress
  - "`goahead[i] = 1 – goahead[j]`" is not atomic
- Hint for mutual exclusion
  - One thread can unlock "at a bad time"

Conceptual warnings!

- "Pathological scheduler" != bounded-waiting failure
  - If scheduler never runs you, *no* mutex can fix that
- "Can't get the lock right away" != bounded-waiting failure
  - Need to show unbounded entries by other parties
- *Traces that show impossible execution sequences*

Q4 – Faulty Mutex

Advice

- You really do want to figure out the trace on scrap paper instead of launching bravely into exploring the state space (it's big)
- Possible approaches
  - Assume a problem of some type, think about parts of the code that could result in that
  - Recall common failure patterns (TOCTTOU)

Q5 – Interrupt-Handler Wrapper

Question goals

- Test understanding of stack discipline
- Test understanding of interrupts
- Test understanding of assembly language (to some extent)

Q5 – Interrupt-Handler Wrapper

Question goals

- Test understanding of stack discipline
  - Which parts need to be updated/preserved?
- Test understanding of interrupts
  - In particular, “stack discipline” isn't really the standard
- Test understanding of assembly language (to some extent)

Q5 – Interrupt-Handler Wrapper

Scoring adjustment

- Part A/B point split was set to 6/4

Expected solution

- A: Somebody forgot about %edx
- B: A time-window/register-usage argument why this will often not be noticed

Also accepted for Part A

- Timer is (arguably) acknowledged too early
  - Good scenario explanation required
- Timer is (arguably) acknowledged too late
  - Good scenario explanation required
- [Other, more-rare suggestions]
  - Good scenario explanation required

Q5 – Interrupt-Handler Wrapper

Conceptual warnings

- %EFLAGS *is* important to preserve
  - So important that IRET does it
- PUSHA/POPA handle general-purpose registers (only)
- Words don't have types: INCL doesn't know whether it's running on a signed or unsigned value

Breakdown

```
90% = 63.0
80% = 56.0
70% = 49.0
60% = 42.0
50% = 35.0
<50%
```

Breakdown

```
90% = 63.0    1 student
80% = 56.0    9 students
70% = 49.0   10 students
60% = 42.0   11 students
50% = 35.0   15 students
40% = 28.0    8 students
<40%          5 students
```

Comparison/calibration

- These scores are low – plausibly 10% too low
- Some adjustment is likely after detailed analysis

Implications

Score below 42?

- 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: draft plan, see instructor

Implications

Score below 28?

- Something went *dangerously* wrong
  - It's important to figure out what!
- Beware of "triple whammy"
  - Low score on deadlock *and* fifo_cond *and* mutex
    - » Those questions are the "core material"
    - » Strong scores on Q1+Q5 don't make up for serious trouble with core material
    - » This was a comparatively hard mutex question, the other two core questions were comparatively easy
- 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)
- 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!**