15-410

"My other car is a cdr" -- Unknown

Exam #1
Oct. 19, 2011

Dave Eckhardt

Synchronization

Checkpoint 2 - alerts

- 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

Asking for trouble

- If your code isn't in your 410 AFS space every day, you are asking for trouble
- 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

Upcoming Events

Google "Summer of Code"

- http://code.google.com/soc/
- Hack on an open-source project
  - And get paid (possibly get recruited, probably not a lot)
- Projects with CMU connections: Plan 9, OpenAFS (see me)

CMU SCS "Coding in the Summer"?

15-412 (Fall)

- If you want more time in the kernel after 410...
- If you want to see what other kernels are like, from the inside

Synchronization

Crash box

- How many people have had to wait in line to run code on the crash box?
  - How long?

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

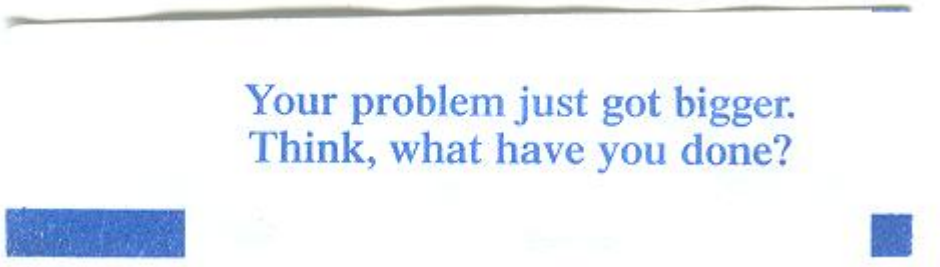

Image credit: Kartik Subramanian

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 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 – “rodata”

Main idea

- “read-only data” - part of an ELF file with semantics
  - Read-only – if program writes, program is broken
  - Data – as opposed to instructions (“text”)
- Deployed as read-only memory (if feasible)

What you should tell us

- Read-only, data...
- “String constants go there”

What you should *not* tell us

- “Code goes there”

Q1b – "internal fragmentation"

Key ideas

- Fragmentation (you should definitely know this)
- Internal vs. external (minor ding if you got them swapped)

Common troublesome answer

- "internal fragmentation is that mysterious thing which causes each region of the ELF file to consume an extra ½ page in memory"
  - Examples are good when they help you remember a principle
  - Make sure you *are* remembering the principle, not just the example
  - This example was mentioned in class as being of trivial importance
    - » The *other* example (also 213 material) has a much larger effect

Q2 – Game-console deadlock

Good news

- Many people did "pretty well"

The key insight

- One part has "hold AND wait", the other has "hold XOR wait"

Hazardous misconception

```
while (!got_x || !got_y) {
  got_x = acquire_x_if_available();
  got_y = acquire_y_if_available();
  if (!got_x || !got_y) block();
}
// This code does NOT "lock in order"!
```

Q3 – Race conditions

3(a) - warmup

- The "cannot fail" code just failed... why?
  - Most people got this
- How to fix it?
  - Beware "layer violation"
    - » If code outside a module that calls interface functions has a problem, fixing the problem should involve using the interface functions differently
    - » Avoid trying to call the internals

Q3 – Race conditions

3(b) – Find race conditions on your own

- Again, most people did well here
- Some people identified "interesting event sequences" that are not race conditions
  - "Sometimes A happens, but sometimes B happens" is not a race bug...
  - ...unless either A or B is *wrong* (an invalid computation)

3(c) – Propose fixes

- "Just broaden the scope of some lock"?
  - "Single global mutex": the "global variable" of concurrent programming. Of course concurrency is easy when there isn't any! Be careful.
  - Increase the scope of a lock to include *long-running* code: only as a last-ditch emergency. Others will pile up on such a lock.

Q4 – “big reader locks”

Question goals

- “Write a synchronization object” - typical exam question
- “Avoid cache-coherence traffic” - an unusual design constraint, but valid given limitations

Basic idea

1. By definition (“mission statement”): Each reader must be able to acquire/release a read lock by operating on state no other reader uses...

Q4 – “big reader locks”

Question goals

- “Write a synchronization object” - typical exam question
- “Avoid cache-coherence traffic” - an unusual design constraint, but valid given limitations

Basic idea

1. By definition (“mission statement”): Each reader must be able to acquire/release a read lock by operating on state no other reader uses...
2. Ok... looks like we will need one _____ per reader, all of them stored in _____

Q4 – “big reader locks”

Question goals

- “Write a synchronization object” - typical exam question
- “Avoid cache-coherence traffic” - an unusual design constraint, but valid given limitations

Basic idea

1. By definition (“mission statement”): Each reader must be able to acquire/release a read lock by operating on state no other reader uses...
2. Ok... looks like we will need one _____ per reader, all of them stored in _____
3. Light bulb should illuminate around now...

Q4 – “big reader locks”

Specific issues

- Long-term blocking using mutexes?
  - Please review “atomic instruction sequence” assumptions
  - Mutexes are for something, please use them for that thing
- Reimplemented some thread synchronization object from scratch?
  - That's kind of non-modular...
  - ...and sometimes it was done wrong
  - Advice: consider *multiple* standard synchronization objects before picking one.
- “Be careful out there”:
  - “Paradise Lost”
  - Progress problems
  - Some random races

Q5 – interrupt/exception/trap frame

Q5(a): In P1 (no privilege change ⇒ no stack switch), what does the CPU save when there is a "surprise"?

- In general, three pieces
  - %EIP – most people got this
  - %EFLAGS – most people remembered this
  - %CS – the x86-specific weirdness (it's part of %EIP)
- Some people mentioned: error code
  - True for some surprises
- Anti-answers
  - %CR2 – a rational design arguably would, but this one doesn't
  - %EAX, "callee-save registers", "all registers"

Q5 – interrupt/exception/trap frame

Q5(b): Explain rationale for each save

- Most-commonly-missed fact: %EFLAGS contains the "condition codes" (set by ALU ops, tested by branches)
- Frequent claim: we need to save and restore %ESP
  - In this case (no stack switch), we "restore" %ESP by popping stuff

Breakdown

```
90% = 67.5    8 students (66 and up)
80% = 60.0   15 students (59 and up)
70% = 52.5    9 students (51 and up)
60% = 45.0    5 students
50% = 37.5    3 students
<50%          3 students
```

Comparison/calibration

- People took longer than usual on the exam
- Grades aren't unusually low

Implications

Score under 51?

- 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?
- Probably plan to do better on the final exam

Score at/below 35?

- Something went *dangerously* wrong
  - It's important to figure out what!
- Passing the final exam may be a *serious* challenge
- To pass the class you must demonstrate proficiency on exams (not just project grades)

Implications

"Special anti-course-passing syndrome":

- You got only the "mercy points" on several questions
- Extreme case: *no* question was convincingly answered
  - It is very important that you don't have *two* exams without evidence that *some* topics have been mastered!
    - » So if this exam looks that way, you should probably see course staff to reduce the likelihood that both do!

Reminder...

- Final exam will focus more on "design"
  - On this exam, design was best represented by brlocks ...