15-410

"...[insert witty quote here]..."

Exam #1
Mar. 14, 2005

Dave Eckhardt

Synchronization

Checkpoint 2 – Friday, in cluster

- Reminder: context switch ≠ interrupt
  - Later other things will invoke it too

Upcoming events

- 15-412 – switched to Fall semester!
- Summer internship with SCS Facilities?

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 implemention

Examination will change to match

- More design questions
- Some things you won't have implemented

Outline

Question 1

Question 2

Question 3

Question 4

Question 5

Q1 – Short Answer

User mode vs. kernel mode

- Some instructions reserved for kernel
- All memory is (in some sense) available to kernel
- All hardware devices available to kernel
- Interrupts, traps enter kernel mode

Less strong

- Kernel stacks are smaller (OS design choice)
- Kernel schedules threads (ususally; if they're k threads...)

Q1 – Definitions

runnable

- "...if it can be run" - not very deep
- "right answer" has two parts
  - No impediment to running / eligible for scheduling
    - » No I/O in progress; not stunned by deschedule()
  - Not currently running on any CPU

Q1 – Definitions

progress

- We were looking for: critical-section progress requirement
- Less strong: “doing useful work”

bus lock

- Almost always refers to *memory* bus, during atomic operation (Test&Set, Exchange, etc.)
- Less strong: some notion of interrupt queueing

Q2 – Deadlock ("ClusterF")

Part A – Prevention

- Recite the "Four Ways to Forgiveness"
- Not right: describing some dynamic scheme

Part B – Analysis of Prevention Approaches

- Should be clear that API *requires* mutex, non-preemption
- Banning hold & wait
  - Changes protocol, not API – fine
  - Risks starvation – better than deadlock
- Banning circular wait
  - "Always go clockwise" doesn't work: consider *two* circuits...
  - Acquiring links in numerical order (vs. *circuit* order) works
    - » requires changing protocol – fine

Q3 – iSpend Interrupt Handling

Key concept

- Trap gates ⇒ *each* interrupt can arrive once
- Stack must hold *sum* of all stack requirements, not max

Q4 – Porting traceback() to P1

What's missing in the P1 environment?

- Convenient pointer-validation approach!
  - No msync(), no /proc, no write()

How would you approximate?

- Basic idea: pointers outside 0...16M are not valid
- Refinements
  - Stack frames probably don't occupy “special” memory
    - » Video RAM, kernel code, ...
  - Stack frames should probably be in stack region(s)
    - » ...various approaches...

Q5 – Critical-section Algorithm

Algorithm – Dekker (see text), broken subtly

Result: doesn't guarantee *bounded waiting*

- The "back off if we're both trying to enter" part is broken
  - One party has an execution pattern which backs off *always*
  - Then other party backs off *never*
- Work through this... see a course staff member if further hints necessary

Popular near-miss

- Showing bounded waiting is broken, calling it progress

Summary

```
90% = 67.5    7 students
80% = 60.0   13 students
70% = 52.5   10 students
60% = 45.0   11 students
<60%         10 students
```

Summary

```
90% = 67.5    7 students
80% = 60.0   13 students
70% = 52.5   10 students
60% = 45.0   11 students
<60%         10 students
```

But...

- **Most popular score was 52.0!!!**

Summary (tweaked)

```
89% = 67.0    9 students
80% = 60.0   11 students
69% = 52.0   16 students
60% = 45.0    5 students
<60%         10 students
```

Implications

Score below 52?

- Figure out what happened
- Probably plan to do better on the final exam

Score below 35?

- Something went *drastically* wrong
- Passing the final exam may be a serious challenge
- To pass the class you must demonstrate some proficiency on exams (project grades alone are not sufficient)