15-410

"My other car is a cdr" -- Unknown

Exam #1
Mar. 2, 2015

Dave Eckhardt

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 10:42:30
    - » 6-9 arrive at 10:59:27
- 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

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, Spring 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

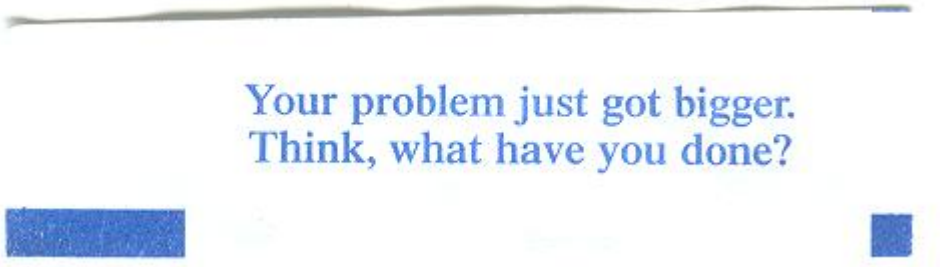

Image credit: Kartik Subramanian

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 – “BSS”

For full credit

- Segment/region
- Holds program's global/static variables that start at 0
- Saves space in the executable file
- Expanded by loader

Q1a – “BSS”

For full credit

- Segment/region
- Holds program's global/static variables that start at 0
- Saves space in the executable file
- Expanded by loader

Typical issues

- Confusion with ZFOD
  - BSS and ZFOD both have something to do with 0
  - BSS saves disk space (can save time too)
  - ZFOD is a time hack (can save RAM too)

Q1a – "BSS"

For full credit

- Segment/region
- Holds program's global/static variables that start at 0
- Saves space in the executable file
- Expanded by loader

Typical issues

- Confusion with ZFOD
  - BSS and ZFOD both have something to do with 0
  - BSS saves disk space (can save time too)
  - ZFOD is a time hack (can save RAM too)
- Conceptual
  - Some answers indicated *dangerous* confusion
    - » BSS is in the stack with the shared libraries
    - » When a zero variable gets a new value, it moves

Q1b – "Three Kinds of Error"

Note condition in text at top of page!

- "As it applies to this course"

For full credit

- "Three kinds of error" from the "Errors" lecture
  - Name
  - What to do when it happens
  - Ideally an example
- This is *genuinely important* for you to know
  - Robustness is a substantial part of passing P3

Most-common notable issues

- Remembering only the names (won't help your code)
- Various error taxonomies unrelated to the class
- "3 kinds of surprise" (trap/fault/interrupt)
  - Good to know, but 2.5/3 are not errors

Q2 – Faulty Mutex

Problem

- Find the problem in the mutex code

Q2 – Faulty Mutex

Problem

- Find the problem in the mutex code

Solution

- Well, there were several

Q2 – Faulty Mutex

Problem

- Find the problem in the mutex code

Solution

- Well, there were several
  - Progress, in a lock/unlock race
  - Mutual exclusion, due to ambiguity of "turn"
    - » turn should authorize *one* entry, but...
  - Bounded waiting (due to a dependency)

Q2 – Faulty Mutex

Problem

- Find the problem in the mutex code

Solution

- Well, there were several
  - Progress, in a lock/unlock race
  - Mutual exclusion, due to ambiguity of "turn"
    - » turn should authorize *one* entry, but...
  - Bounded waiting (due to a dependency)

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*

Q3 – Deadlock

Parts of the problem

- Basic deadlock explanation
- Deadlock prevention?
- Deadlock avoidance?

Q3 – Deadlock

Parts of the problem

- Basic deadlock explanation
  - Most people did well here
- Deadlock prevention?
  - Easy (*clear* explanation required)
- Deadlock avoidance?
  - Dubious (a few people gave an ok argument)

Q3 – Deadlock

Concerns

- This was a very straightfoward question
  - If you didn't get 8/10, you should aim to find and fix the issue
- Invalid process-resource graph, invalid trace
  - Tools are important for analyzing problems
- T/F: Avoidance requires multi-instance resources?

Q3 – Deadlock

Concerns

- This was a very straightfoward question
  - If you didn't get 8/10, you should aim to find and fix the issue
- Invalid process-resource graph, invalid trace
  - Tools are important for analyzing problems
- T/F: Avoidance requires multi-instance resources?

Warnings

- "Global mutex" is *a* solution
  - *Every* concurrency problem can be solved by a global mutex
  - It is never a *high-quality* solution
- "Invalid usage results in deadlock" isn't a valid argument
  - Invalid usage of *anything* results in bad outcomes!

Q4 – "Master-slave message passing"

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

Q4 – “Master-slave message passing”

“Be careful out there”

- Busy-wait/spin-loop – use an accepted synch object!
- Waking up threads when it really doesn't make sense
- Deadlock scenarios
- Memory leaks

Question-specific conceptual problems

- recv() isn't a barrier with respect to recv()
- send() isn't a barrier with respect to recv()
- send() isn't a barrier with respect to send()
- Object works only once (state not reset when people finish)

Q5 – Segmented Stacks

Question goals

- Test understanding of stack discipline
- Test design (What do I need to accomplish?)
- Test assembly-language ability (to some extent)

Q5 – Segmented Stacks

Question goals

- Test understanding of stack discipline
  - Which parts need to be updated/preserved?
- Test design (What do I need to accomplish?)
  - Calling the function *and* getting back when it returns
- Test assembly-language ability (to some extent)
  - It is possible to use a C helper function, but also a little dangerous

Q5 – Segmented Stacks

Question goals

- Test understanding of stack discipline
  - Which parts need to be updated/preserved?
- Test design (What do I need to accomplish?)
  - Calling the function *and* getting back when it returns
- Test assembly-language ability (to some extent)
  - It is possible to use a C helper function, but also a little dangerous

Q5 – Segmented Stacks

Conceptual problems

- "`set_esp(val)`"
  - If a function is called which internally sets %esp to an indicated value, what happens when that function returns?
- `RET` while %ebp is at the top of stack
- `RET` into places where there isn't code

Common implementation problems

- Calling into C code and expecting %ebp to remain where it was
- Not preserving/restoring %ebp
- Copying parameters (some fixed number of them, or some giant region containing parameters)
- ...others...

Breakdown

```
90% = 67.5
80% = 60.0
70% = 52.5
60% = 45.0
50% = 37.5
40% = 30.0
```

Breakdown

```
90% = 67.5     7 students (top: 74/75)
80% = 60.0    18 students
70% = 52.5    13 students (52 and up)
60% = 45.0     7 students (44 and up)
50% = 37.5     6 students (37 and up)
40% = 30.0     1 student
<40%           1 student
```

Breakdown

```
90% = 67.5     7 students (top: 74/75)
80% = 60.0    18 students
70% = 52.5    13 students (52 and up)
60% = 45.0     7 students (44 and up)
50% = 37.5     6 students (37 and up)
40% = 30.0     1 student
<40%           1 student
```

Comparison/calibration

- Scores “look ok”

Implications

Score “sub-C” (37..47?)?

- 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 37?

- Something went *dangerously* wrong
  - It's important to figure out what!
- Beware of "triple whammy"
  - Low score on mutx *and* deadlock *and* ms-broadcast
    - » Those questions are the "core material"
    - » Strong scores on Q1+Q5 don't make up for serious trouble with core material
- 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!