How To *Fail* This Class
(or achieve an alternative)

Dave Eckhardt
de0u@andrew.cmu.edu

Outline

- Students ***fail*** this class
- What to do instead

Outline

- Students ***fail*** this class
  - Real students!
  - Students who have never failed a class before!
- What to do instead

Outline

- Students ***fail*** this class
  - Real students!
  - Students who have never failed a class before!
- What to do instead

Hey, buddy! Why all the gloom and doom? Don't you ***want*** people to take this class?

Motivation

- A question from an F'14 student
  - “Does this class really have 'D' grades?”
  - “Is there some 'extra credit' so I can graduate?”

Core Issue

- There is a pattern of things to avoid
  - Don't attend class
  - Start projects late
  - Code to the test suite
  - Fail exams
  - Fail kernel project
- *Some of these practices have worked for you in other classes*
  - If so, this class is different

This is a *Transformative* Class

- Genuine achievement, available to you
  - What is an OS, ***really?***
  - Concurrency (locks, races, deadlock)
  - What is VM, really?
  - Process model, C run-time model
  - Interrupts
  - Design synthesis, planning
  - Serious competence in debugging!
- If that sounds like a lot, it is!

This Is a *Hard* Class

- Traditional hazards
  - 410 letter grade one lower than typical classes
  - All ***other*** classes this semester: one grade lower
- Aim
  - If you aim for a B you might not get one
  - If you aim for a C you might not get one
  - “I'll drop if/when I can't definitely get an A”
    - (You ***must*** discuss this with your partner ***early***)

Grading philosophy

- C – all parts of problem addressed
- B – solution is ***complete***, ***stable***, ***robust***
- A – excellent
  - Somebody might want to re-use some of your code

Grading philosophy

- C – all parts of problem addressed
- B – solution is ***complete***, ***stable***, ***robust***
- A – excellent
  - Somebody might want to re-use some of your code
- ***SERIOUS IMPORTANT WARNING***
  - “Passes most of the test suite” does NOT imply 'A'!

Grading philosophy

- C – all parts of problem addressed
- B – solution is ***complete***, ***stable***, ***robust***
- A – excellent
  - Somebody might want to re-use some of your code
- ***SERIOUS IMPORTANT WARNING***
  - “Passes most of the test suite” does NOT imply 'A'!
  - “Passes most of the test suite” does NOT imply 'B'!

Grading philosophy

- C – all parts of problem addressed
- B – solution is ***complete***, ***stable***, ***robust***
- A – excellent
  - Somebody might want to re-use some of your code
- ***SERIOUS IMPORTANT WARNING***
  - “Passes most of the test suite” does NOT imply 'A'!
  - “Passes most of the test suite” does NOT imply 'B'!
  - “Passes most of the test suite” does NOT imply 'C'!

Grading philosophy

- C – all parts of problem addressed
- B – solution is ***complete***, ***stable***, ***robust***
- A – excellent
  - Somebody might want to re-use some of your code
- ***<u>SERIOUS IMPORTANT WARNING</u>***
  - “Passes most of the test suite” does NOT imply 'A'!
  - “Passes most of the test suite” does NOT imply 'B'!
  - “Passes most of the test suite” does NOT imply 'C'!
  - “Passes most of the test suite” does NOT imply 'D'!

Grading philosophy

- “Passes most of the test suite” isn't even a 'D'???
  - Really?

Grading philosophy

- “Passes most of the test suite” isn't even a 'D'???
  - Most often it is
  - But “passes most of the test suite” can still mean ***failing the kernel project***

Grading philosophy

- Key issue
  - Robust code is ***structurally different*** than fragile code

Grading philosophy

- Key issue
  - Robust code is ***structurally different*** than fragile code
- Surprising problems
  - You can't ship “hacked on madly until it sort of passes the test suite” to customers
  - Can't write bad code, ***then*** make it robust, ***then*** design it to be modular
  - Can't write race-infested code and then add locking
  - Can't write non-preemptible code and then make it preemptible

Grading philosophy

- Key issue
  - Robust code is ***structurally different*** than fragile code
- Surprising situation
  - *A substantial part of each project grade will be an assessment of whether your code is robust*

Grading philosophy

- Key issue
  - Robust code is ***structurally different*** than fragile code
- Surprising situation
  - *A substantial part of each project grade will be an assessment of whether your code is robust*
  - Code that passes the test suite but which is not robust will not get a good grade
    - There will be lectures on robust design and robust code
    - They will seem weird
    - To do well, you will need to apply those weird ideas
    - *Exam questions* may be based on those weird ideas

Grading philosophy

- C – all parts of problem addressed
- B – solution is ***complete***, ***stable***, ***robust***
- A – excellent
  - Somebody might want to re-use some of your code
- Numbers?
  - A = 90-100%, B = 80-90%, ... (roughly)
- "Curving"? Maybe, not necessarily
  - Lots of A's would be ***fine with us***
  - ***But this requires clean, communicative code!***

Grading philosophy

- C – all parts of problem addressed
- B – solution is ***complete***, ***stable***, ***robust***
- A – excellent
  - Somebody might want to re-use some of your code
- By the way, there are grades above A!

Grading philosophy

- C – all parts of problem addressed
- B – solution is ***complete***, ***stable***, ***robust***
- A – excellent
  - Somebody might want to re-use some of your code
- By the way, there are grades above A!
  - "Recommended for 15-712"

Grading philosophy

- C – all parts of problem addressed
- B – solution is ***complete***, ***stable***, ***robust***
- A – excellent
  - Somebody might want to re-use some of your code
- By the way, there are grades above A!
  - "Recommended for 15-712"
  - "TA candidate"

Grading philosophy

- C – all parts of problem addressed
- B – solution is ***complete***, ***stable***, ***robust***
- A – excellent
  - Somebody might want to re-use some of your code
- By the way, there are grades above A!
  - "Recommended for 15-712"
  - "TA candidate"
  - "Recommended for research / Ph.D. programs"

Grading philosophy

- C – all parts of problem addressed
- B – solution is ***complete***, ***stable***, ***robust***
- A – excellent
  - Somebody might want to re-use some of your code
- By the way, there are grades above A!
  - "Recommended for 15-712"
  - "TA candidate"
  - "Recommended for research / Ph.D. programs"
  - These pretty much ***require*** starting early + excellence

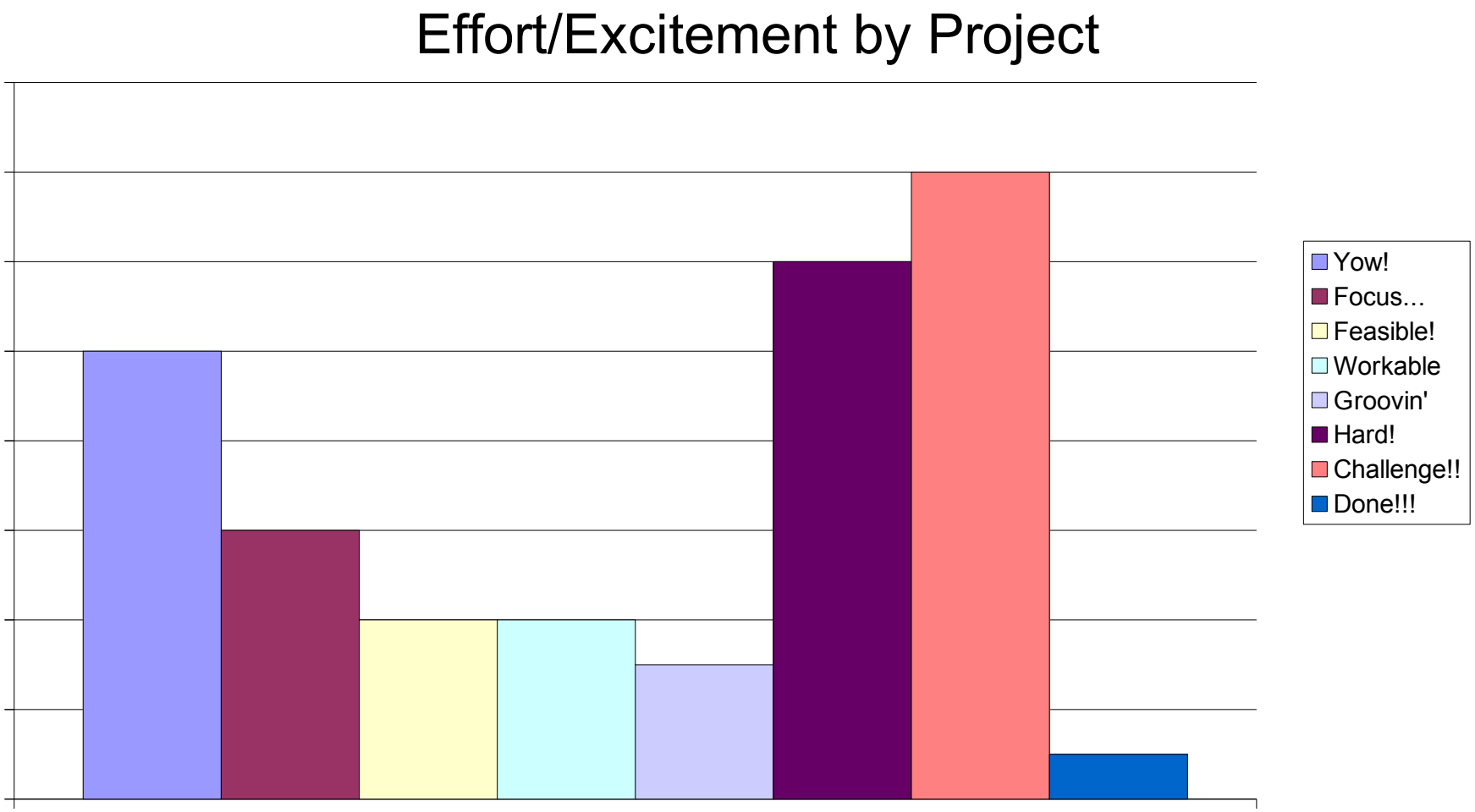
“TA Candidate”

- Two ***mutually exclusive*** options...
  - Deploy your high-powered “throwing stuff together at the last minute” skills for this class

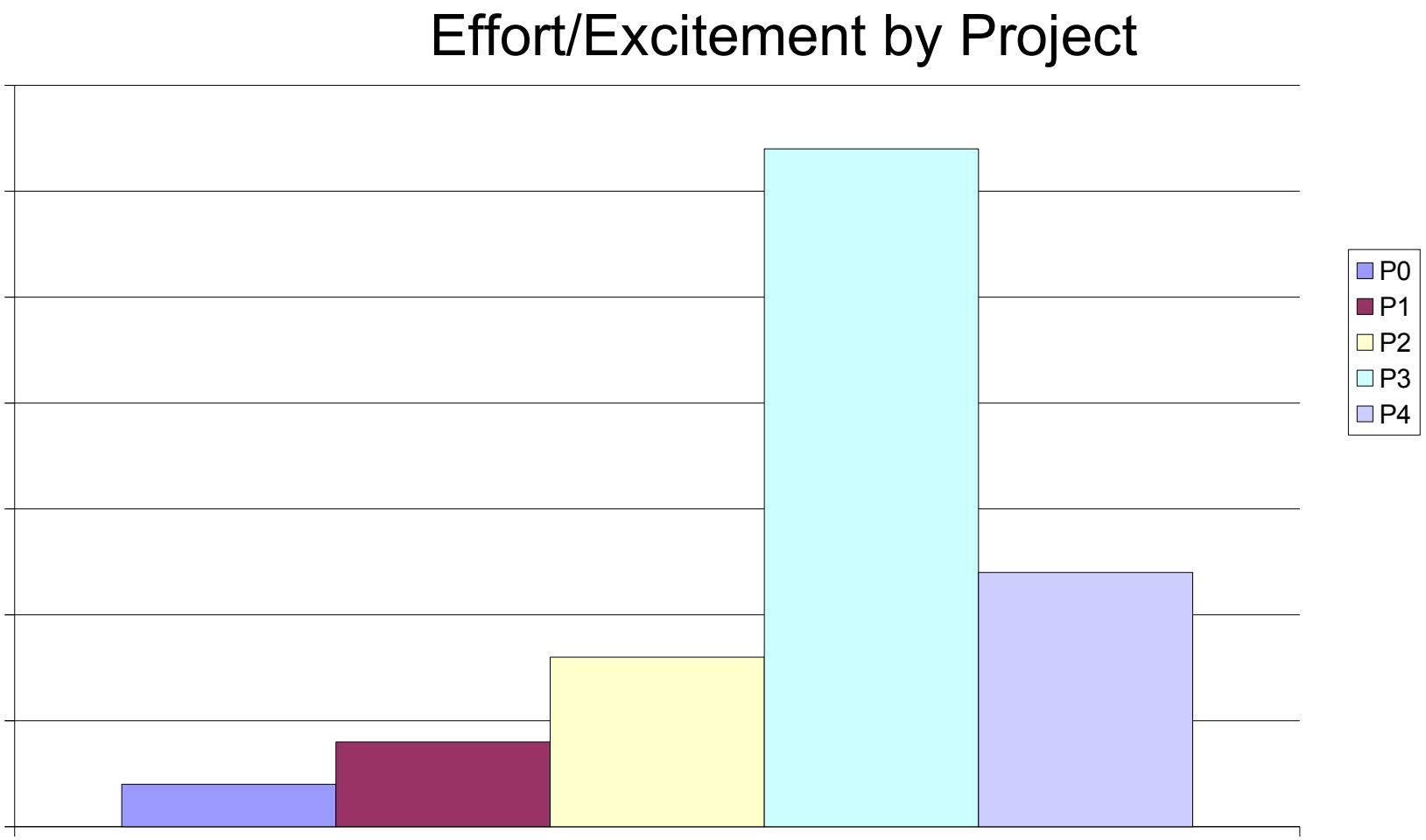
    XOR
  - Start on time, engage new design tools/practices, write clear & excellent code, work well with your partner, hit milestones early, do optional parts

- If you are interested in being a TA next semester, arguably let us know early

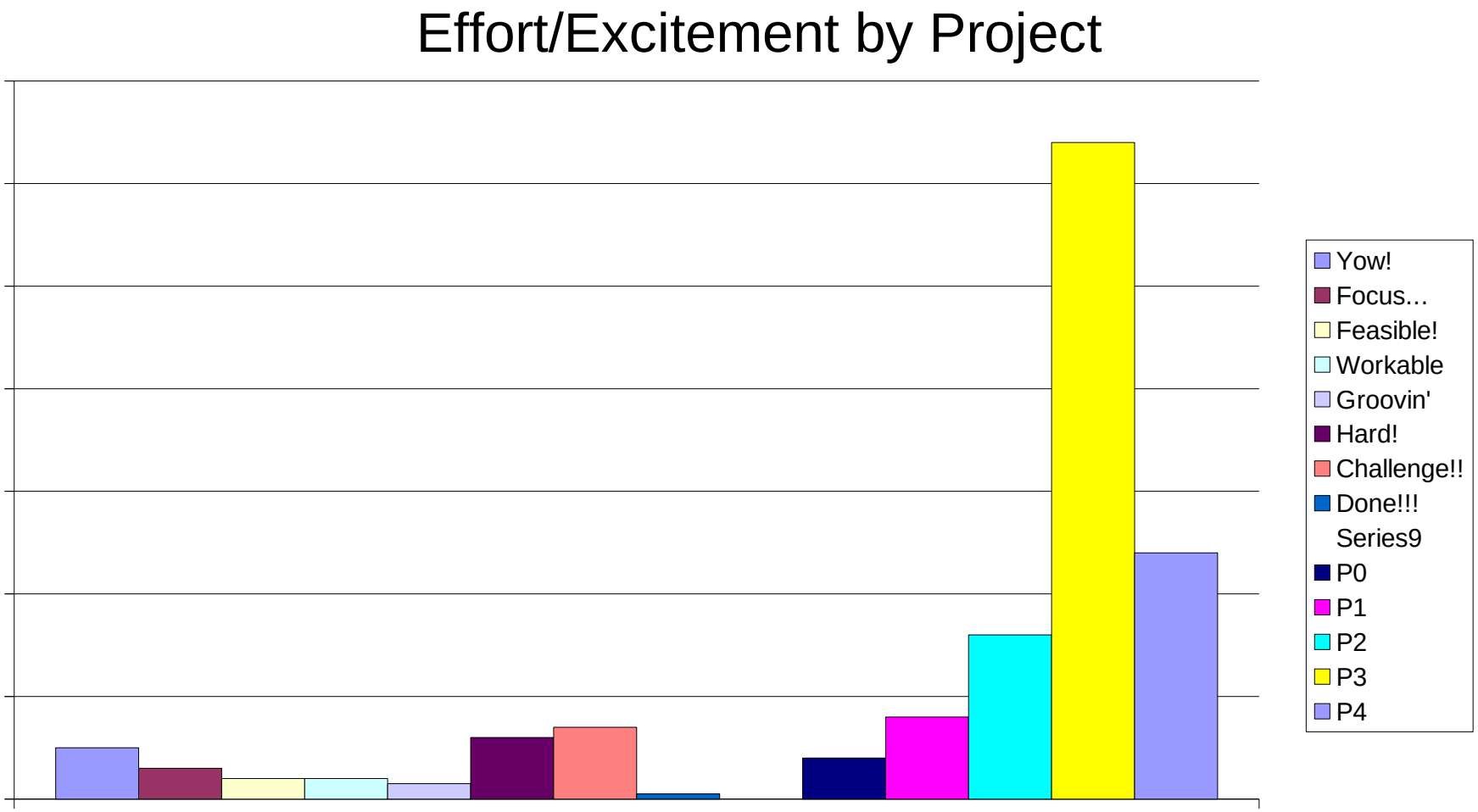
The Shape of Some Class



15-410



Scale Matters!



Implications

- "Trouble with one assignment" is *real* trouble
  - You can't just "swing at the next ball"
    - The next ball is two to four times faster!
  - Each project is training for the next (like Math)
    - If you skip part of one project, the next one might be *unachievable*.
- So...
  - Aim to do *really well* on P0
    - Start *the first day* (*for sure* by the second)
  - Then recover, aim to do *even better* on P1

Academic Integrity

- This is a design class
  - Not a “cut&paste class” or a “looking things up” class
  - We expect you to practice *solving design problems*
- Model: our spec ⇒ your ideas ⇒ your code
  - Not: copy code from Linux kernel
  - Not: port code from some other OS class / web site
  - *Completely* not: use some other student's code
- There are exceptions
  - Some uses of some outside code are ok: *see syllabus!*

Academic Integrity

- Model: our spec ⇒ your ideas ⇒ your code

Academic Integrity

- "We expect you to fail"
  - It is possible to fail an assignment and pass the class
    - (If you come from another university this may be new)
  - It is *not* possible to copy or cheat on an assignment and pass the class
    - Beyond failing, other dreadful things happen too
      - Dean of Student Affairs
      - Scholarship problems
      - Graduation delays
    - Please don't turn a simple failure into a giant catastrophe

Stack Overflow

- It's great!

Stack Overflow

- It's great!
  - Answers to lots of questions!

Stack Overflow

- It's great!
  - Answers to lots of questions!
  - Arguably *two* kinds of questions...

Stack Overflow

- It's great!
  - Answers to lots of questions!
  - Arguably *two* kinds of questions...
    - WAIT! LET'S DO UNSUPERVISED ONLINE LEARNING!!!

Stack Overflow (2018-08-22)

0 postfix - status=bounced (unknown user)

Stack Overflow (2018-08-22)

0 postfix - status=bounced (unknown user)

Is there a reason to use an SSL certificate other than Let's Encrypt's free SSL?

Stack Overflow (2018-08-22)

0 postfix - status=bounced (unknown user)

Is there a reason to use an SSL certificate other than Let's Encrypt's free SSL?

Chemical equation that can be balanced in infinite number of ways?

Can you sneak into Premium Economy Class?

Three coins are tossed. If one of them shows a tails, what is the probability that all three coins show tails?

Stack Overflow (2018-08-22)

0 postfix - status=bounced (unknown user)

Is there a reason to use an SSL certificate other than Let's Encrypt's free SSL?

Chemical equation that can be balanced in infinite number of ways?

Can you sneak into Premium Economy Class?

Three coins are tossed. If one of them shows a tails, what is the probability that all three coins show tails?

A person who has a hard time throwing away unnecessary things

Stack Overflow (2018-08-22)

0 postfix - status=bounced (unknown user)

Is there a reason to use an SSL certificate other than Let's Encrypt's free SSL?

Chemical equation that can be balanced in infinite number of ways?

Can you sneak into Premium Economy Class?

Three coins are tossed. If one of them shows a tails, what is the probability that all three coins show tails?

A person who has a hard time throwing away unnecessary things

- Who sees the two clusters?

Stack Overflow (2018-08-22)

0 postfix - status=bounced (unknown user)

Is there a reason to use an SSL certificate other than Let's Encrypt's free SSL?

Chemical equation that can be balanced in infinite number of ways?

Can you sneak into Premium Economy Class?

Three coins are tossed. If one of them shows a tails, what is the probability that all three coins show tails?

A person who has a hard time throwing away unnecessary things

- Now who sees the two clusters?

Stack Overflow

- It's great!
  - ...for decrypting an obscure Docker error message
- It's *very not ok*...
  - ...for figuring out how to write 15-513 shellcode
- When you leave here...
  - You may be expected to solve silly glitches answered on Stack Overflow by looking on Stack Overflow
  - You will almost certainly be hired because you can solve hard problems *without* Stack Overflow
  - Please don't confuse these while you're at CMU

Work Flow – You may be used to...

- Assignment handout ⇒ code outline
- Compilation implies correctness
- Graded by a script
- All done!
  - Never use it again
  - Delete it at end of semester
- ***Total opposite of real life***

Work Flow – 410 Additions

- Design
- Divide into parts
- Manage your partner
- Merge
- Debug ***hard*** problems

Surprises

- "Code complete" means ***"I am far behind"***
  - Merge can take ***three days***
  - Then you ***start*** to find bugs (1-2 weeks)
- Code with "the right idea" will ***immediately*** crash
  - If you're lucky!
- This is not a "basic idea is right" class
  - You can't ship "basic ideas" to customers
  - Understand all details–***then*** you have the basic idea

On Debugging

As soon as we started programming, we found to our surprise that it wasn't as easy to get programs right as we had thought. Debugging had to be discovered. I can remember the exact instant when I realized that a large part of my life from then on was going to be spent in finding mistakes in my own programs.

– Maurice Wilkes (1949)

Debugging

- Bugs aren't just last-minute glitches
- They are crucial learning experiences
  - Learning a lot can take a lot of time

What Does A Bug Mean?

- “It tells me 'triple fault' – why??”
  - Research: 20 minutes
  - Think: 20 minutes
  - Debug: 2 hours.
  - ...three times.
- May need to ***write code*** to trap a bad bug
  - Asserts or more-targeted debug module
- Then you will find your design was wrong!
  - Don't be shocked – this is part of 410 / life

“All Done”?

- Finally, when you're done...
  - You will use your code for the next assignment!
  - We will read it (goal: every line)

The deadline disaster

- "If you wait until the last minute, it takes only a minute!" -- Vince Cate
- Small problem
  - Your grade will probably suffer
- Big problem
  - ***Learning*** and ***retention*** require sleep
  - Why work super-hard only to forget?

How to Have Trouble

- How to get an R
  - Arrive unprepared (e.g., barely escape 213)
  - Do everything at the last minute
  - Don't read the book or attend class
  - Hide from course staff no matter what
- How to get a D
  - Don't get the kernel project genuinely working
    - (There are other ways, but this one is popular)

Interview Advisory

- Taking 7 days off during P2 for interviews won't be recoverable
  - Empirically demonstrated!
- Taking 15 days off during P3 for interviews won't be recoverable
  - Also empirically demonstrated!

- If you find yourself drifting toward either of those, talk to us / your advisor (sooner is better)

How to do well!

- Confront the material
  - Attend class!
- Confront debugging
- Embrace the experience
  - Unix, Simics, revision control
- Invest in *good* code
- Start unbelievably early
- Read your partner's code
- Leave time for design

Confront the Material

- We are doing printf() ***all the way down***
  - Subroutine linkage, how & why
  - Stub routine, IDT entry, trap handler wrapper
  - Output/input-echo interlock
  - Logical cursor vs. physical cursor
  - Video memory (what does scrolling mean?)
- Can't really gloss over ***anything***

Confront Debugging

- Real life: you will debug other people's code
  - Any bug could be yours, partner's, ours, or Simics; you need to ***find*** it.
- ***Can't*** debug using only printf()
  - printf() ***changes your code***
  - printf() may be broken by whatever breaks your code
  - Learn the Simics debugger
  - Assertions, consistency checks
  - Debugging code

On Investing

- A week of coding can sometimes save an hour of thought.
  - Josh Bloch

Confront Debugging

- ½ hour of studying the debugger
  - vs. 2 days of thrashing
- Papering over a problem
  - Re-ordering object files to avoid crash

Doing Well – Embrace the Experience

- Embrace the Unix development experience
  - If you try to keep it at arm's length it will slow you down
- Embrace the Simics debugger
  - If you try to keep it at arm's length it will slow you down
- Embrace source control
  - If you keep it at arm's length ...

Doing Well – Invest in Good Code

- Mentally commit to writing ***good*** code
  - Not just something kinda-ok
  - You will ***depend*** on your code
- Anand Thakker (Fall 2003)
  - Remind yourself that you love yourself...
  - ...so you should write good code for yourself

Doing Well – Start Early

- Starting a week late on a 2-week project will be bad
- Not making “just one” checkpoint can be bad
  - Missing two kernel-project checkpoints...
    - ...may make passing impossible.

Doing Well – Read Partner's Code

- You will ***need*** to read everything your partner wrote
  - (and answer test questions about it)
- Set up a mechanism
  - Daily meeting? Careful reading of merge logs?
- Do “one of each”
  - Partner does N-1 stub routines, you should do the hardest

Doing Well – Time for Design

- “Design” means you may need to think overnight

How to get an A

- Understand ***everything***
  - (consider 2-3 ways to do each thing, pick the best)
- Write ***genuinely excellent code***
  - asserts, good variable names, source control
- Document ***before*** coding
  - Actual 15-410 students do this!
- Read ***all of*** your partner's code
- Work ***with*** your partner (merge *continuously*)
- Be “done” *early*, “just in case”

First Item of Work

- Read the syllabus as soon as it's released
  - It contains things you need to know
    - Things which will be painful surprises if you don't know them
- Thanks!