CS 402
Operating Systems

Bill Cheng

http://merlot.usc.edu/cs402-s16

Fact Or Myth?

- Is it true that doing well in this class gets you job offers?

Fact Or Myth?

- Is it true that doing well in this class gets you job offers?
  - definitely true!
    - if you *learn the course material* well, and
    - if you implemented all the assignments *yourself* and without looking at other people’s code)
    - if you *participate* in the class Google Group discussions, especially to help out other students
- Whatever you can find on the Internet, everyone else can find it
  - you cannot distinguish yourself by just reading
  - if you want to impress your interviewers, you need to impress them with your *experience (integrated with your knowledge)*
    - the "theory" stuff is just as important as "kernel hacking" and this is *not* a programming / kernel hacking class

My Teaching Style

- I'm a strong believer in: (adapted from Lao Tzu)

  Give a man a fish and you feed him for a day.
  Teach a man to fish and you feed him for a lifetime.

  - except for the warmup programming assignments, I tend not to give a straight answer
    - I want you to find the answers yourself (together with your peers)
    - I will help by pointing you to the right direction
      - so, you should feel free to ask me questions

No "Spoon-feeding"

- The coverage of this class is quite vast
  - too many important concepts to cover in 15 weeks
  - you have to *learn some things on your own*
    - it's part of your education
  - it is not feasible to explain everything till you understand it perfectly
    - *please do not expect "spoon-feeding"*
  - I will explain the concepts, but you have to work hard so you fully understand them
    - come *talk to me* as much as you need!
      - if you are not used to talking to an instructor, you need to get comfortable with it!
      - you are *expected* to come talk to me if you have trouble with course material or assignments

Summary of Important Rules (from PREVIEW)

- DEN Videos
  - DEN lecture videos are accessiable to everyone
- Kernel Teams
  - up to 4 students per team will be permitted but no more
  - all team members must be registered in the same *lecture* section
- Exams
  - you must take all the exams in the section for which you are registered.
- Late Registration
  - you will be expected to turn in all assignments on time, no matter when you get into this class
- Participation (extra credit)
  - only if you attend the lectures and discuss sections for which you are registered

Summary of Important Rules (from PREVIEW)

- Preparation
  - the "prerequisites" are for undergrads, for grad students, they are considered "recommended preparation"
  - you must *learn C* and *Unix on your own*
    - we will *not* teach you C or Unix/Linux
      - you don't need to be an Unix expert, you just need to know the basics such as directory listing, creating directories, changing directories, copy files, delete files, etc.
      - you need to know Unix/Linux to *test your kernel code*
    - best way is to install Ubuntu Linux on your laptop/desktop and work on your warmup assignments on it
      - Linux is not Unix, but very similar
      - warmup assignments must run on `nunki.usc.edu`
      - kernel assignments must run on Ubuntu Linux 12.04

The Importance of "Written Words"

- We communicate (here and in the real world) using "words"
  - you need to learn to take "words" seriously
  - especially when it comes to *rules written in words*
    - things that are not written can get messy
- If it's writtent that X is the grading procedure and that we must follow the grading procedure
  - what would we do if you ask us to grade your submission using a different procedure?
    - we won't, because we take written words very seriously
- When it comes to exams, we can only grade based on what you wrote on the exam paper (and not what's in your mind)
  - you need to learn to choose your words carefully
- Please pay attention to *all* the written words anywhere in the *class web site*, *lecture slides*, posting to class Google Group by *me*
  - setup your e-mail filter to not miss messages from <bill.cheng@usc.edu>

Fairness

- Fairness
  - without fairness, grades have little meaning
  - the instructor *must treat all students equally* and cannot give special treatment to any particular student
  - therefore, please do not ask special favors from the instructor because of your circumstances (except for ones that are explicitly allowed by the university)
  - this may seem unfair to you because you believe that your circumstances are special (understandably, everyone does)
  - bottom line, the rule the instructor must follow is that *whatever he offers you, he must offer to the entire class*

Today's Topics

- Administrative Stuff
  - the instructor *cannot* give D-clearance
    - please use the on-line D-clearance system
  - undergrad students must take CS 350 for Operating Systems
  - our class is *significantly different* from CS 350
    - textbook and programming assignments are all different
- Review Course Organization
  - please read all the administrative lecture slides
- The discussion section this week will start you off working on "warmup" assignment #1
  - please read all the week 1 discussion section slides
  - please *read the spec yourself*
- Chapter 1
  - 1.1 introduction
  - 1.3 a simple OS

Class Structure & Teaching Staff

- Instructor: Bill Cheng <bill.cheng@usc.edu>
  - email: 24 hour turn-around
  - office hours: in SAL 302, M/W 2:00-3:00pm, Tu/Th 12:30-1:30pm, or by appointments
- Lectures
  - *(MW section)* MW 12:00pm - 1:50pm in ZHS 252
  - *(AM section)* TuTh 9:30am - 10:50am in SOS B46
  - *(DEN section)* TuTh 11:00am - 12:20pm in OHE 132
  - The lectures of these sections will be *synchronized*
- You are expected to attend every lecture and discussion section
  - OS is not just about programming, it's about important OS concepts

Class Structure & Teaching Staff

- TA
  - *(MW section)* Muhammad Rizwan Saeed <saeedm@usc.edu>
  - *(AM section)* Yue Shi <yueshi@usc.edu>
  - *(DEN section)* Sung-Han Lin <sunghan@usc.edu>
  - email: 24 hour turn-around
  - the TA's job is to *help* you with the course materials, programming assignments, *grade exams*, and conduct discussion sections
    - TA cannot *do work for you* (such as *find bugs* in your code, *write code* for you to use, etc.)
    - TA *cannot tell you what code to write*
    - TA can sit down and run the debugger *with you*
- You can go to *any* TA for help with course materials and programming assignments

Class Structure & Teaching Staff

- Course producer (CP)
  - Rahulkumar Mishra <rmishra@usc.edu>
  - email: 24 hour turn-around
  - the CP's job is to *help* you with the course materials, programming assignments
    - CP cannot *do work for you* (such as *find bugs* in your code, *write code* for you to use, etc.)
    - CP *cannot tell you what code to write*
    - CP can sit down and run the debugger *with you*
- No matter which section you are in, you can go to the CP for help with course materials and programming assignments
- The TAs and the CP will post their office/helpdesk hours
  - if you are not available during any of their office/helpdesk hours or my office hours, please then make an appointment with me or any one of them

Class Structure & Teaching Staff

- Graders:
  - email: 24 hour turn-around
  - the grader will hold regrade sessions after you get grade notification e-mail
  - we have different rules about grader involvement for our class
    - the grader's only job is to ***grade your programming assignments***
    - it is inappropriate to contact the grader about an assignment (especially about "how many points I would get if I do it this way") before the assignment is due
      - you should be able to figure it out from the spec and the "grading guidelines" or you can just ask me
      - the grader will not answer questions before the assignment is due

Class Resources

- Class web page: http://merlot.usc.edu/cs402-s16
  - everything about this class is there
    - anything related to *grading*, you are *required to know*
  - get familiar with it
    - if you are not used to reading a lot of stuff, you should start reading a lot of stuff in this class
      - it's important to learn how to *read* documents and *interpret* them *correctly*

- If you see inconsistencies, especially regarding any type of "rules" between what's on the class web page and what's on a set of lecture slides that has been covered in class
  - usually, the lecture slides are correct
  - but, you should check with the instructor as soon as possible!
    - so that things can be consistent again

Lectures

- The posted lecture slides have a lot of details
  - I will *not* cover every posted lecture slides
    - not even for *this* set of lecture slides!
    - there's not enough time
    - although *you will be responsible for everything posted* in lecture slides (and the corresponding materials in textbook)
      - *except* the ones that are marked with a grey ✖ in the *lower left corner* of the slide
      - feel free to ask me about things on the lecture slides but were not covered during lectures
- *Exam questions* will be primarily based on the lectures and lecture slides
  - it's important that you *understand* the lecture slides well
  - you should use the lecture slides as a study guide

Discussion Sections

- Conducted by the TAs
  - Fri 10:00am - 10:50am in OHE 136 (on DEN - Sung-Han will lead)
  - Fri 11:00am - 11:50am in GFS 116 (Yue will lead)
  - Fri 12:00pm - 1:50pm in MHP 106 (Rizwan will lead)
- The TA will use the discussion sections to:
  - go over background information for programming assignments
  - help with programming assignment specs and requirements
  - answer questions (if you send questions to them the day before)
- *Exam questions* can also be based on discussion section lecture materials (i.e., materials on posted slides)
- Please understand that discussion section material are *NOT* substitute for reading the specs and the grading guidelines
  - you are expect to read the entire *spec*
  - you are expect to read the *requirements* the spec refers to
  - you are expect to read the *grading guidelines*
  - they are your responsibility

Class Discussion (Google) Group

- Google group
  - student-to-student discussions about programming assignments and course materials
  - exchanging *ideas* are allowed
  - posting code is *not* allowed (short code segments to illustrate ideas are allowed; short means < 5 lines)
    - *first offense* (in the entire semester) gets a warning (unless it's a lot more than 5 lines of code)
    - *2nd offense*, you will lose 50% of the corresponding assignment points and lose posting privileges
  - instructor and TAs will also post answers to questions here
    - if appropriate for whole class
  - you can get *extra credit* if you post *timely* and *good/useful* answers to other students' questions for *kernel* assignments

Class Discussion (Google) Group

- You *must be a member* of the *class Google Group*
  - *all* important announcements will be posted to this group
- You are expected to read *every one of my posts* to the class Google Group
  - *my posts* to the class Google Group is considered *course material*
    - I use it to explain lectures (if someone asks)
    - I use it to explain programming assignments (if someone asks)
    - you must not block these messages!
      - if you really don’t want to read them, you can setup a filter to skip your inbox
    - I may ask *exam questions* from them

Class Discussion (Google) Group

- There are two way to use the class Google Group
  1) use it as an e-mail reflector (i.e., don’t need to "login" to the Google Group)
     - get an e-mail copy of everything posted to the group
     - send postings to the group
     - you *don’t need* a Google account
     - this option is *not* available to you if your USC e-mail is accessed through *Google Apps at USC*
     - if you are a conscientious objector to Google, this may be your only option (and you will have to manage ALL messages posted to the class Google Group)
  2) full access (i.e., "login" to the group, search the message archive, post using web form, etc.)
     - you *must have* a Google account
     - this is the *preferred* way

Class Discussion (Google) Group

- If you are on the class roster, I will invite you to join the class Google Group
  - by default, I will use your USC e-mail address
    - therefore, the default mode of using the class Google Group is e-mail reflector (although the other mode is preferred)
    - if you read e-mail using *Google Apps at USC*
      - a) *do not accept* the invitation because it won't work
      - b) you have to use method (2)
  - if you want to use a different e-mail address (such as gmail) to access Google Group, please do the following
    - a) *do not accept* the invitation
    - b) login to Google with your other e-mail address
      - click on our class Google Group link and apply for membership
      - make sure you provide your *USC e-mail* address (not USC ID) in the *"additional information"* field (so I can verify that you are really in my class)

Grading

Roll call	4%	starting with week 4 (extra credit)
Projects	40%	2 warm-ups (individual), 3 group projects
Midterm	25%	in class, Wed/Thu, 3/23,24/2016 *(firm)*
Final	35%	*(MW section)* 11am-1pm, Fri, 5/6/2016 *(firm)*
		(AM section) 8am-10am, Tue, 5/10/2016 *(firm)*
		(DEN section) 11am-1pm, Tue, 5/10/2016 *(firm)*

- Additional extra credit
  1) turn in *assignments* more than 48 hours before deadline
  2) giving good/useful answers in the class Google Group for *kernel assignments*

- No *individual* extra credit
  - try your best from the beginning!

Grading

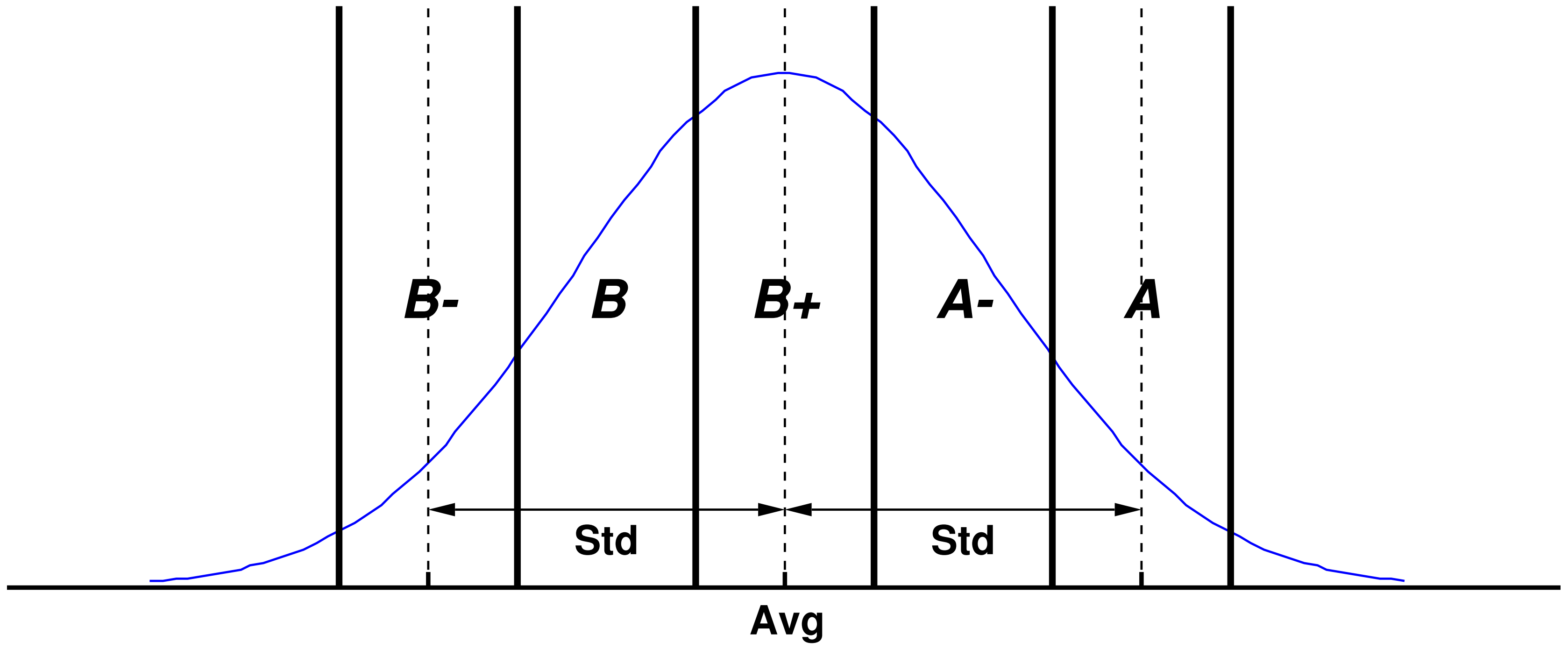
- Projects graded by the graders
  Exams graded by the TAs

- Final letter grade assigned by the instructor
  - a grade of *incomplete* is only possible for *documented* illness or *documented* family emergency (according to USC policy)

- Two methods
  1) on a curve
  2) fixed scale
  - your class letter grade will be the *higher* of the two
  - C's may be given, F's if necessary

Grading

- **Curve**
  - one curve for each section (since exams are different)
  - loose guideline depicted below:



Grading

- **Fixed scale**

Percentage	Letter Grade
91% or higher	A
82-91%	A-
73-82%	B+
64-73%	B
55-64%	B-
46-55%	C+
37-46%	C
28-37%	C-
below 28%	F

Academic Integrity Policy

- The USC Student Conduct Code *prohibits plagiarism*
  - for warmup projects, all submitted work must be *your own work*
    - you must not *look at* code from previous semesters
    - you must not *copy* a single line of code from other sources
  - for group projects, all submitted work must be work done by members of your group
    - any member of your team must not *look at* code from previous semesters
    - any member your team must not *copy* a single line of code from other sources

- Two exceptions for copying code
  - code fragments done by yourself for other classes or given to you as class resource (in a class you have taken and completed) must be cited explicitly
    - must cite the code *inline* (or points will be deducted)
  - you can use any code *given to you* as part of *this class*
    - no need to cite such code

Academic Integrity Policy (Cont...)

- You are encouraged to work with other students for individual assignments or with other groups for group projects
  - "work with" does not mean "copy each other's work"
  - "work with" means discussing and solving problems together
    - this should happen at a *high level*
  - but be very careful when it's time to write code
    - must write code completely on your own
    - *do not write code together*
    - "sharing" even a *single* line of code is considered *cheating*
- If you cannot work together at a high level
  - you are advised not work together with other students for individual assignments or with other groups for group projects
- For more details, please see the *Academic Integrity Policy* section on the *Course Description* web page

Academic Integrity Policy (Cont...)

- For kernel assignments, if you know your partner is cheating, you must tell him/her to stop
  - there is only one submission from your team
  - you won't get graded separately from your teammates
    - if the university determined that the submission by your team was the result of plagiarism, everyone in the team will receive a grade of F for the class
  - even if you don't know that your partner is cheating, the same policy applies
    - choose your partners carefully
    - talk to your partners to make sure that no one is cheating
      - *throw away code written by other people*
      - if you know that a partner of yours has code from previous semester, you should ask him/her to throw away the code
    - talk to your partner if he/she cannot explain the code he/she claimed to have written

Academic Integrity Policy (Cont...)

- You must *not* post your code to a *public* github (or anything similar)
  - if you post it to a *private* github, you need to *verify* that it's truly private
    - ask your friend or kernel partners to verify
- Since our projects are on-going projects, you must not knowingly make your code public (not even pseudo-code)
  - if I find your code posted in a public place, you (and your team) will get a zero for your assignment
    - if you post your code after the semester is over, I will ask the university to change your grade

Displaying Your Code in Public Repositories

- As a general rule, you should only post code to the public if the spec is public and the code you are depending on is also public
  - all our assignment specs are private
  - the code given to you in our assignment are private
  - *you must not post any of your CS 402 code to a public repository*
- You do not have the right to post it as part of your online resume
  - because your code depends on the rest of the assignment which you do *not* have the rights to *display* or *distribute*
  - this is serious business!
    - your future boss would/should appreciate that you understand about software copyrights

Program Checker

- Do not submit someone else's code
- How do we catch cheaters?
  - we use MOSS to analyze your submissions
    - http://theory.stanford.edu/~aiken/moss/
  - analyzes code structure intelligently
  - we have *all* projects from previous semesters

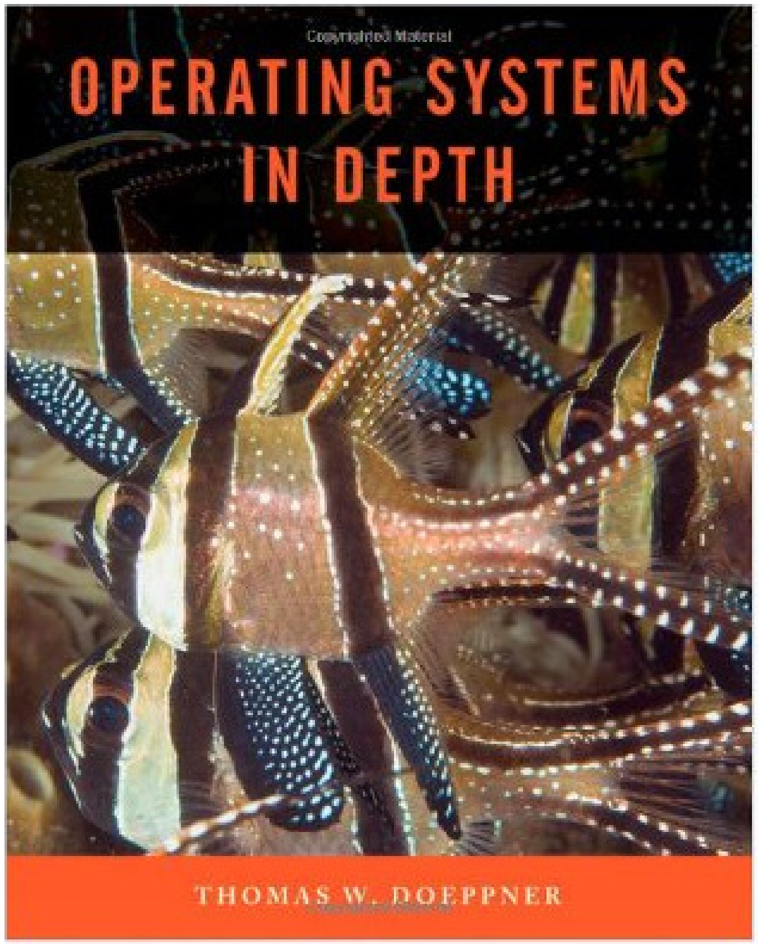
E-mail Questions

- One type of question I often get over e-mail or see in class Google Group:
  - here is my understanding of X. "Am I right?" "Is this correct?" "Correct me if I’m wrong..." "Please confirm that I’m right..."
    - this is really not a good way to ask something
      - if no one corrects you, you must *not* conclude that you were correct
    - stick to the definition of X in lecture slides or in the textbook and try to understand why it was stated that way
    - perfectly reasonable to discuss this during office hours
    - find a different way to ask *over e-mail* to be more productive

- Another type of question I often get about assignments:
  - I am thinking about not following the spec and do this instead. Is it acceptable (or is this okay)?
    - stick to the written words (spec and grading guidelines)!

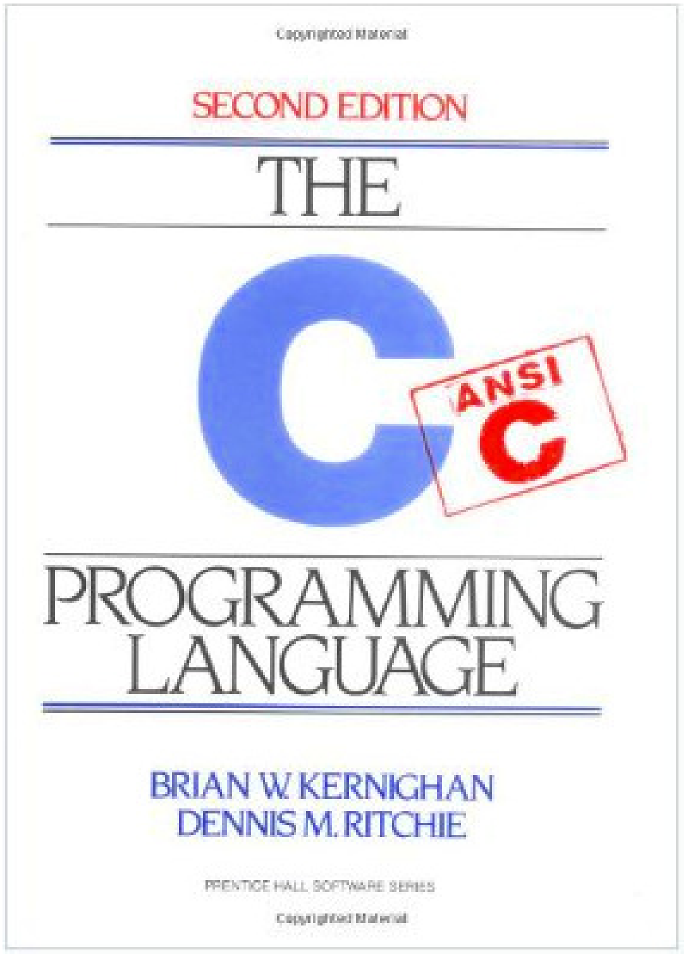
Course Readings

- ***Required*** **textbook**
  - ***"Operating Systems In Depth: Design and Programming"*** **by T. W. Doeppner**
  - **we will follow this book closely**



Course Readings

- *Optional* textbook
  - *"C Programming Language"* by B. Kernighan and D. Ritchie
  - all programming assignments must be done in C



Class Structure

- Lectures mostly based on Doeppner
  - slides
  - lectures should be interactive
  - lecture slides posted on class web site soon after class
- You must keep up with the assigned readings
  - you are expected to read the relevant textbook chapter thoroughly
  - not all details will be covered in class
  - exam may test material covered in the textbook but not discussed in class
- I expect you to attend every class meeting
  - if you do happen to miss a class, you are responsible for finding out what material was covered and what administrative announcements were made
  - you will be expected to attend in-class exams

Lecture Slides

- Lecture slides came with the book
  - the purpose of lectures is to explain what’s in the textbook
    - the textbook is not all that easy to understand
    - lecture slides are not meant as "extra material"
  - lecture slides have a lot of details
    - too much details?!
      - some are verbatim from the textbook
    - too much details can be a good thing
      - it tells you what’s important to study
      - you don’t need a study guide for exams!
- You need to understand every slide
  - do you need to read the textbook?
    - no if you just want to do well in the exams and you understand everything covered in lectures
    - although highly recommended
      - who can remember everything from lectures?

Sign Roll Sheet

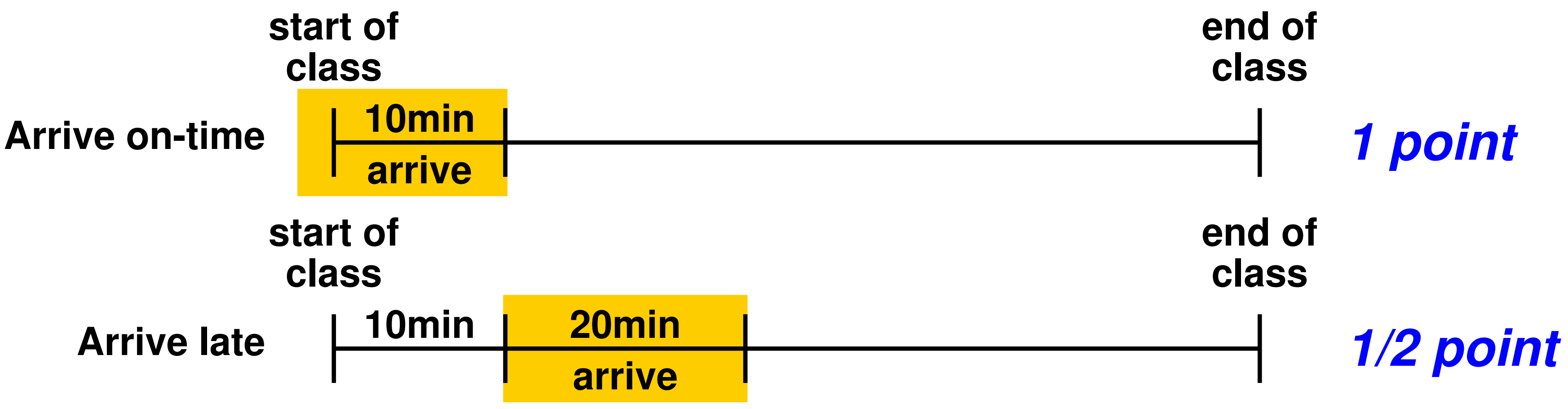
- For *on-campus* students only
  - if you are registered in the 29946D section, you are considered a "DEN remote" student, and you are not required to sign roll sheets
    - you will get the 4% extra credit automatically
  - DEN remote students normally have a slight disadvantage that they can miss stuffs happened in class
    - this is served as an *"equalizer"*

Sign Roll Sheet

- Starting with *week 4*, each lecture is 1 point and each discussion section is 1 point
  - you should sign the roll sheet as soon as you get into class
  - if your signature is on the roll sheet, you get the 1 point
    - *sign roll sheet only if you plan to stay till the end of lecture*
  - if you come in between 11 to 30 minutes into class, check *late* box
    - you will receive 1/2 a point credit instead
    - if you have to leave during class and cannot remove your signature, please e-mail me to have your signature removed
      - failure to do so would be considered cheating and will lose *all* class participation credits
  - if you are sick or have a family emergency, please bring a note from a doctor to receive credit
  - if you ask another student to sign you in or out, both students are considered cheating

Sign Roll Sheet

- Sign Roll Sheet Summary



  - ***0 point*** **if** ***arrive way late***

- **Personally, I really don't understand why anyone wants to come to class late on purpose!**
  - **if you miss the first 10 minutes of a lecture, you may be confused for most of the lecture**

- **You only get extra credit if you** ***attend lectures for the section you are registered***

Sign Roll Sheet

- If you plan to cheat (i.e., sign roll sheet and sneak out)
  - this 4% extra credit is really not for you
  - by signing the roll sheet, you are promising that you will stay for the entire lecture or discussion section
    - please don’t sign the roll sheet unless you are planning to stay to the end
    - occasionally, I will ask everyone to "sign out" right before class ends
      - if you signed in but did not sign out, it’s considered cheating and I will report the incident to USC Student Judicial Affairs

- Please understand that *cheating on roll sheeting signing* is a *very serious offense*
  - I can understand why people cheat on programming assignments when people get desperate
  - cheating on roll sheeting signing is the same as *stealing*
    - there is *no justification* for stealing

Projects / Programming Assignments

- Programming assignments
  - (8%, 17%) 2 warm-up projects *(to be done individually)*
    - linked list in C, pthreads (no kernel code)
  - (25%, 25%, 25%) 3 kernel projects *(to be done individually or in a group of 2-4 students)*

    1) kernel threads & processes 2) virtual file system layer

    3) virtual memory (extremely difficult)
  - no solutions will be given
  - program in C only (and you must learn C on your own)
    - C is a proper subset of C++
      - although stream I/O and strings are not available in C
      - you must learn how to do I/O the right way
      - you must learn how to deal with C-strings (i.e., null-terminated array of chars)
  - the kernel assignments are *extremely difficult*
    - should get started as soon as possible

Projects / Programming Assignments

- Kernel assignments must be done on Ubuntu 12.04 (with QEMU 1.0)
  - install the *latest* Ubuntu 12.04 LTS (currently at 12.04.5)

- You should install Ubuntu 12.04 on your laptop/desktop as early as possible and let me know if there are problems
  - follow the instruction at the bottom of the class web page
    - if you have a Windows machine with 4GB of memory and Intel Core i3 or faster processor, download *VMware Player 7* and install Ubuntu 12.04 into it
      - on Mac OS X, use Oracle's VirtualBox (free) or "parallels"
    - otherwise
      - if you have a older Windows 7/8 laptop/desktop, use *WUBI* to install Ubuntu 12.04
      - if you have a slow Mac, you may have to create a separate disk partition and install Ubuntu 12.04 into it

this is the *preferred* method

Projects / Programming Assignments

- For kernel (group) projects, half the grade is "team grade" and half is "individual grade"
  - in the README file your team will submit, you need to:
    - specify how to split the points (in terms of percentages and must sum to 100%)
    - give a brief justification about the split
  - for exmaple, the grader gives you 80 points and everyone contributed equally, everyone gets 80 points
  - what if it's not equal contribution?
    - what if the split is 0/50/50? what should be your scores?
    - what if the split is 20/30/50? what should be your scores?
- We would like to *encourage* that everyone contribute to the team *equally* (or at least for you to declare as so in your README file)
  - your score is maximized when you have equal contribution
  - there would be "penalty" for not dividing the scores equally
    - this can lead to unfairness, but that's the nature of group projects

Projects / Programming Assignments

- How do we figure out the scores when there is an uneven breakdown?
  - it's kind of complicated
    - the basic idea is to calculate the standard deviation among the contribution percentages
    - if there is an even breakdown, the standard deviation is zero
      - this gives an upper bound and no penalty
    - if the breakdown is 0/0/100 (highest possible standard deviation), the would be the worst case, i.e., most penalty
      - but in this case, the person with 100% will get all the points assigned by the grader and everyone else will get 50% of the points
  - run a program nunki/aludra:
    - general syntax

```
        ~csci551b/bin/cs402calc grade p1 p2 ...
```

      - where `grade` is the grade the grader gave, and `p1, p2, ...` are percentages

Projects / Programming Assignments

- How do we figure out the scores when there is an uneven breakdown?
  - run a program nunki/aludra:
    - for example, if the grader gave 80 points and the breakdown is 20/30/50, run:

      ```
      ~csci551b/bin/cs402calc 80 20 30 50
      ```

      - the scores will be 55.695, 63.5425, and 79.2375
      - so, the person who contribute the most did not get 80
    - if the grader gave 80 points and the breakdown is 0/50/50:

      ```
      ~csci551b/bin/cs402calc 80 0 50 50
      ```

      - the scores will be 40, 70, and 70
      - more penalty for not being able to figure out how to share responsibilities among 3 students!
      - may be it's not worthwhile to assign such percentages!
  - I'll leave it to you to decide!
- If you don't specify, we will assume it's an equal-share split

Projects / Programming Assignments (Cont...)

- Forming groups
  - you can only form a group with *students from your own section*
  - It's probably a good idea to work with other students during the warmup projects
    - although you must *write code independently* for the warmup projects
    - "work with" means working at a high level (*not* code level)
  - all students not belonging to a group by the group forming deadline will be assigned a group
    - algorithm:
      - form a random list from these students (random drawing)
      - assign 3 to a group starting from the beginning of the list
      - remaining students join these randomly formed groups
      - this is the only way to have 4-student groups
      - boundary conditions? hope we don't get there
  - after groups are formed, only mutually agreed swaps are allowed (must let us know)

Projects / Programming Assignments (Cont...)

- We know that our algorithm is far from perfect
  - that's why it's specified way in advance
  - this way, you can plan how to form teams given that you know our algorithm for assigning teams
- If you cannot form a team of your own and end up in a team that cannot get the kernel assignments to work?
  - probably because there are problem members
  - is this unfair to you?
    - No! because you did have options
  - given that you know that this is the rule, what should you do?
    - form your own team, or
    - do the kernel projects by yourself!
    - if you *choose* neither, we cannot grade you differently *given our fairness policy*
  - during the first 6 weeks, work with other students so you know whom you want to be partners

Electronic Submissions

- Use *bsubmit* and the *Bistro* system (see web page for details)
  - you can make multiple submissions
    - will grade last submission by default
  - Bistro system gives tickets and receipts
    - these are *proofs* that the *server* got your submission
    - *we cannot trust any file timestamp*
    - we can only trust things that have made it to our server
  - very important: *read output of bsubmit*
  - very important: *verify your submissions*
    - see the bottom of the Electronic Submission Guidelines web page for details
    - if you forget a file in your submission, you are not allowed to resubmit it after the deadline
  - submit source code only (or 2 points will be deducted)
  - for team projects, only one member needs to submit
  - it is *your responsibility* to make sure that your submission is valid - *Be paranoid!*

Electronic Submissions (Cont...)

- Use *bsubmit* and the *Bistro* system (see web page for details)
  - no other form of submission will be accepted
    - use your judgement under special circumstances

- Account getting full
  - please do not delete or alter anything in your `~/.bistro` directory
  - If you must delete your `~/.bistro` directory, you should tar then gzip the content of your `~/.bistro` directory and e-mail the resulting tgz file to the instructor:

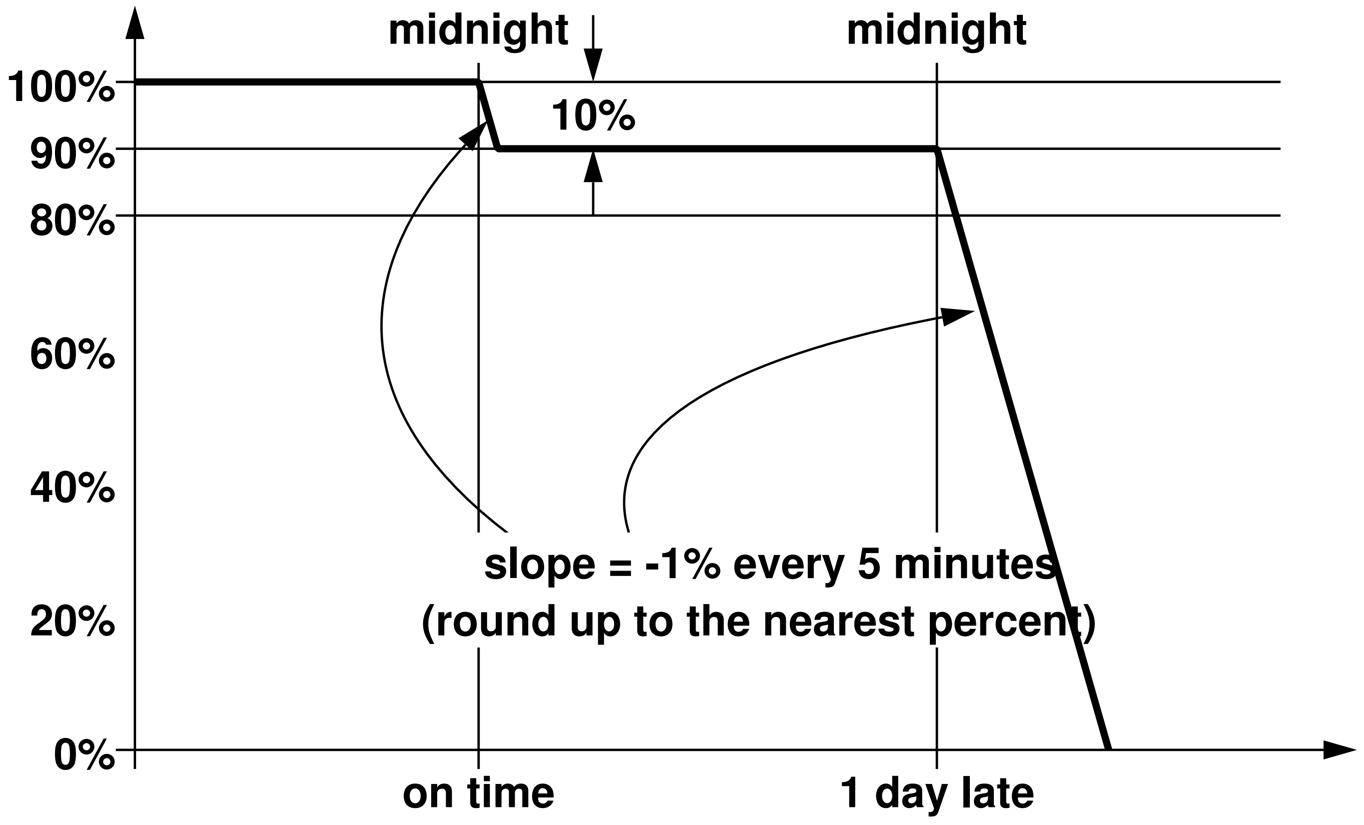
```
cd; gtar cvzf $(USER)-bistro.tgz .bistro
```

  - delete the directory only after you get an confirmation e-mail from the instructor (or at your own risk)

Late Policy

- Electronically Submitted Assignments
  - you can submit multiple times, only the last submission will be graded (unless you send us an e-mail)
  - 15 minutes grace period
  - 90% of your score if within one day late beyond grace period
  - although in the first 50 minutes of this period, you will only lose 1% of your grade every 5 minutes
  - see next page for details
  - time is based on *Bistro* *server timestamp*
- Extension *only possible* if you have a *note from a doctor* or other form of official proof of family emergencies
  - e.g., scheduling conflict with work or other classes cannot be excused

Late Policy


midnight
midnight
100%
90%
80%
60%
40%
20%
0%
10%
slope = -1% every 5 minutes
(round up to the nearest percent)
on time
1 day late

Late Policy

- I must stick to my policies
  1) please do not ask for individual extension unless you have a *documented* proof of *illness* or a *documented* proof of *family (not personal) emergency*
  2) my "fairness" policy is: *"Whatever I offer you, I must offer it to the whole class"*
     - this is why I cannot give individual extensions
  - what if your laptop died? home Internet disrupted? car broken? cousin got stuck at the airport? my house was on fire? need to go to court? need to go to Miami? and so on ...
    - these are personal emergencies
    - please see (1) and (2) above

Modifications After Deadline

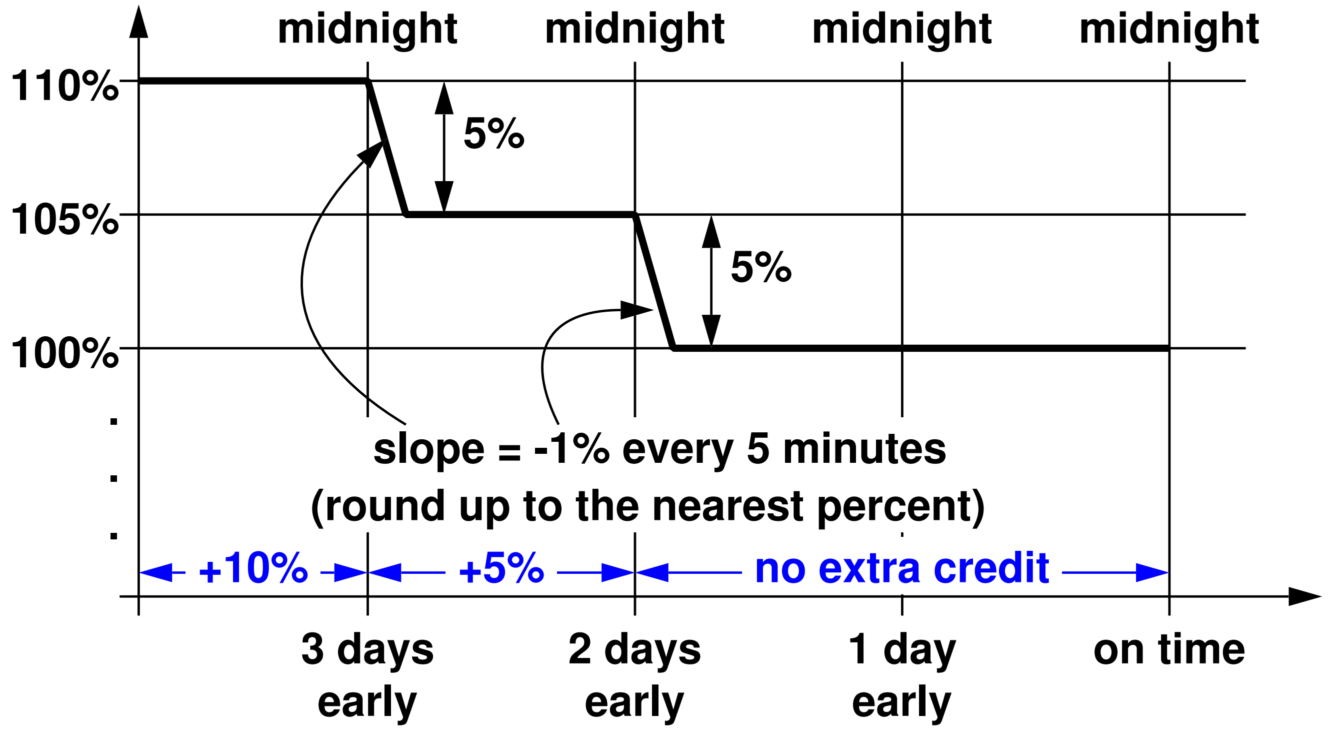
- After the submission deadline has past
  - you are allowed up to *3 lines of free changes*, submitted via e-mail to the instructor, *up to 24 hour after* the project submission deadline
    - clearly, this is not meant for major changes
    - you may want to anticipate that your submission may not be exactly what you thought you had submitted
  - one line (128 characters max) of change is defined as one of the following:
    - *add* 1 line before (or after) line x
    - *delete* line x
    - *replace* line x by 1 line
    - *move* line x before (or after) line y
  - additional modifications at 3 points per line (same deadline)
  - (cont...)

Modifications After Deadline (cont...)

- After the submission deadline has past (cont...)
  - 24 hours after the submission deadline, additional modifications cost *12 points per line* for the next 6 days
  - afterwards, it costs *30 points per line*
  - applies to source code and README files
    - do not forget to submit files, *verify* your submission
      - this is *your responsibility*
    - we cannot accept missing files after deadline
    - a *filesystem timestamp* can be easily *forged*, so they cannot beused as *proof* that you have not modified the file after deadline
  - try things out before your first submission deadline to get familiar with the *Bistro* system
  - *re-test* your code after you have submitted to be sure

Extra Credit 1

- To encourage you to do your projects early, you will get extra credit if you turn in programming assignment 2 or 3 days early
  - if your submission is more than 72 hours before the posted deadline, you get an ***extra 10%***
  - if your submission is between 48 and 72 hours before the posted deadline, you get an ***extra 5%***



Extra Credit 2

- You can get extra credit for posting *timely*, *useful*, and *insightful* answers to the class Google Group in response to questions posted by other students regarding *kernel* programming assignments
  - you probably won't get extra credit if you repeat exactly what another student has already posted
    - you can still post to support another student's position
  - you probably won't get extra credit if you respond more than 48 hours after the time of the original post
  - the maximum number of extra credit points you can get is *10* points for each of the kernel assigments (on a 100-point scale)
  - if you post something good, it's your responsibility to verify that it has been posted to the Google Group
  - you can *lose* extra points you have earned if you post something unprofessional to the class Google Group or exhibit poor netiquette

Regrade Policy

- Grades will be sent to individuals via e-mail
  - you must register for the class mailing list
- Regrade requests in writing
  - submit within 1 week of initial grade notification
    - must follow instruction in grade notification e-mail
  - regrade can be done after the 1-week deadline, but you must *initiate* a regrade request within 1 week
  - we reserve the right to regrade the whole thing

Student Commitments

- Keep up with your reading
  - complete relevant reading before class
  - browse lecture slides before class
    - lecture slides will be available on-line before class
- Do your own work
- Turn in assignments on time
- Ensure gradable assignments
  - read output of bsubmit
  - *verify* your submissions
- You are encouraged to study with other students and *discuss* (no sharing) programming assignments and HWs
- You are encouraged to ask questions, pretty much about anything related to programming
  - when you get stuck with programming, ask the TA or the instructor for help, don't wait too long

Student Commitments (Cont...)

- If you feel that you are falling behind
  - talk to the instructor as soon as possible
- How do you know you are in trouble?
  - assuming that you want a decent grade
  - you look at a set of slides that has been covered in class and have no idea what it means
    - then you read the textbook and understands it perfectly
      - you are probably not in trouble
    - if you read the textbook and are still confused
      - you are somewhat in trouble
    - if you don't read the textbook
      - you are probably in ***big trouble***
  - don't do this right before exams!
  - you should ask youself, "Am I in trouble?" like at least every week if you don't plan to come to lectures

Study for Exams

- Exams are worth a lot! This class is not just about programming!
  - how many kernel programmers do we need in this world?
  - but everyone in CS must understand *OS concepts* well
- Exams mostly are based on lectures
  - I reserve the right to ask anything from required materials
- Exam questions often asks for the *best* answer
  - you get credit for including the "best answer"
  - you may get deductions for including "bad answers"
  - generic answer usually gets you very little partial credit
  - partial credit: better answer may get you more points
- Do not review all lectures only right before the exams
  - otherwise, you may only be able to give generic answers because everything is a blur
  - you need to show that you know the difference between answers of different quality

Things to Do *Today*

- Read entire class web page: http://merlot.usc.edu/cs402-s16
  - check course description and reading list
  - get access to user ID and password
    - should do this even if you are not registered for the class but plan to take this class
  - check warmup project spec and *start coding*
- Additional things to learn/do quickly
  - learn C if you don't know it already
  - learn "git" (see online book)
    - you will need it for group projects
    - should start using it for the warmup projects
  - learn a commandline editor: vim/pico/emacs
    - it's not that hard
  - install *Ubuntu 12.04* on your laptop/desktop and let me know if there are problems
    - VirtualBox preferred if you have Intel Core i3 or faster

Course Content Credit

- Slides and course content primarily came with the textbook and originally written by:
  - Tomas W. Doeppner
- Additional slides and course content from:
  - Ramesh Govindan
- Some (test) code for the kernel assignments from:
  - Ted Faber

ITS Solaris Machine Access

- You need to log into aludra/nunki.usc.edu
  - if your USC e-mail address is YOURLOGIN@usc.edu
    - then your login name is YOURLOGIN (same password)
  - for warmup assignments and to run `bsubmit`
  - SSH from a console (make sure to use "`ssh -X -Y ...`")
  - On Windows, use VirtualBox, Xwin, Cygwin or PuTTY
    - Ubuntu (Linux)
- Transferring Files
  - "`scp`" from a console
  - SFTP/SCP programs
    - Cyberduck, Fugu, etc. (Mac)
    - FileZilla, WinSCP, etc. (Windows)
- Text Editors
  - `emacs`, `pico`, `vi`
- Compiler
  - "`gcc --version`" should say it's version 4.something