CS161 Introductory Lecture: How to Crush CS161

Geoffrey Werner-Allen
07 Feb 2007
Course Objectives

Really fall into two categories:

Learn about Operating Systems:
- After all, this is the topic of the course...
- History
- Theory
- Concepts
Course Objectives

Really fall into two categories:

Learn about Operating Systems:
• After all, this is the topic of the course...
• History
• Theory
• Concepts

Become a Systems Programmer:
• Harvard has no general systems programming class...
• Learn C backwards and forwards
• Solve complex problems
• Write well-documented code
• Evaluate the efficiency and performance of your system
Course Objectives

Really fall into two categories:

**Learn about Operating Systems:**
- After all, this is the topic of the course...
- History
- Theory
- Concepts

**Become a Systems Programmer:**
- Harvard has no general systems programming class...
- Learn C backwards and forwards
- Solve complex problems
- Write well-documented code
- Evaluate the efficiency and performance of your system

...double your pleasure...
(as it were)
Concept Mappings

Class
  Synchronization
  Threads/ Processes
  Virtual Memory
  Filesystems
  Special Topics

Lecture
Concept Mappings

Class
- Synchronization
- Threads/ Processes
- Virtual Memory
- Filesystems
- Special Topics

Assignments
- Locks/CVs
- Hash tables
- Allocators
- Buffer management
- Thread-safe code

Lecture

Office Hours

© Geoffrey Werner-Allen – Harvard University
Assignment Overview

Four distinct programming challenges:

- **ASST1**: Big Idea
- **ASST2**: Well-defined interface
- **ASST3**: Do-it-yourself
- **ASST4**: Itty-bitty changes
ASST1 : Big Idea

- Synchronization!
- Remember recursion? Difficult to get your head around.
- Get it now. You’ll have to use it later.
ASST2: Well-defined interface

- Series of well-defined tasks, very self-similar
- Breaks down easily into groups
- Process support gives you a taste of ASST3
ASST3 : Do-it-yourself

- ASST3 is a big piece of code
- Most people ‘hit the wall’ during this assignment
- Tremendous feeling of accomplishment when you are done
ASST4 : Itty-bitty changes

- Much smaller code volume than ASST3
- But where to put it?
- Practice modifying existing code base
Finally, a course where design is crucial. The more detailed the design, the more cogent the feedback. Should reach from a high-level plan down into nitty-gritty low-level details. High correlation between good designs and good submissions.
Testing

- We provide you with tests. Use them!
- The tests that we give you are a significant portion of our assessment.
- Do not even try to fiddle with your code in order to pass our tests. It doesn’t work, and we have plenty of other tests to run.
Collaboration

- Nothing that you aren’t use to.
- English is OK.
- Pseudo-code is usually OK.
- C code is not OK.
Getting Help

- You are each other’s best resource!
- Use the bulletin board.
- Work together. Long nights in the Science Center terminal room are a traditional part of this course.
- Concepts in lecture, connections in section, code help in OH’s.
Survival : Balance

- The workload for CS161 varies, but is always quite a bit. Be ready!
- Balance is achieved primarily when you **start early**.
- TF’s will provide prompt feedback on design docs so that you can **start early**!
- **START EARLY!**
Survival: Partnering

- Your partner is your best friend.
- You may hate your partner on occasion.
- Communication is essential.
- Come to the teaching staff early if you think you have a serious situation.
- 99% of people who didn’t like this class didn’t have a good partnership.
“(Assignments) could have been significantly improved if the assignment description included a more detailed outline of the design of the project and/or a very complete overview of the design of the existing portions of the operating system that we have to work from.” (CUE, Spring 2006)
Criticism: Responses

“(Assignments) could have been significantly improved if the assignment description included a more detailed outline of the design of the project and/or a very complete overview of the the design of the existing portions of the operating system that we have to work from.” (CUE, Spring 2006)

(This is a common complaint!)

Response:
We consider the imperfect specifications of the assignments to be a feature, not a bug. You're never bound to find these sorts of designs in real life, for almost any real system. And OS/161 is actually commented extremely well compared with other systems.
“The assignment description should include a much more detailed outline of what we have to do to complete the assignment, and even include some design specifications for the project--this would greatly decrease the time spent wondering what we are supposed to be doing and how long it will take for a tf to answer our questions on the bulletin board.” (CUE, Spring 2006)
Criticism: Responses

“The assignment description should include a much more detailed outline of what we have to do to complete the assignment, and even include some design specifications for the project--this would greatly decrease the time spent wondering what we are supposed to be doing and how long it will take for a tf to answer our questions on the bulletin board.” (CUE, Spring 2006)

(Another common complaint!)

Response:
This class is not CS50/51. We are frequently going to give you a big problem and ask you to solve it. Suggestions that we may make about how to complete the assignment are just that: suggestions. However, there will be a portion of your time spent figuring out what to do for each assignment. Again, feature, not a bug.

However, we will try and be better about answering the bulletin board this year.
“The problem about a course like this is that since the assignments are compounded, bugs may resurface from previous assignments. For example, when starting asst4, we couldn't even do much of anything because we apparently had messed up something from assignment 3. We had to spend the first two hours that were meant to be on asst4 fixing bugs from asst3. This usually shouldn't happen I guess, but it did with us.” (CUE, Spring 2006)
"The problem about a course like this is that since the assignments are compounded, bugs may resurface from previous assignments. For example, when starting asst4, we couldn't even do much of anything because we apparently had messed up something from assignment 3. We had to spend the first two hours that were meant to be on asst4 fixing bugs from asst3. This usually shouldn't happen I guess, but it did with us." (CUE, Spring 2006)

Response:
Consider yourself warned!

However, also feel free to approach the TF's either to obtain the solution sets (if your work is really, really unsalvageable) or ask for help debugging problems from prior assignments.
“There is unfortunately a huge disjunction between the lectures and the problem sets. This is true both temporally, but in terms of sheer content at any time as well. We may learn about a solution to a problem in lecture, and then have to implement a completely different solution in the problem sets; these are the types of things that are best covered cursorially in lecture and then hammered home in a programming assignment. I would hope that the lectures and assignments line up a bit better in the future.”

(CUE, Spring 2006)
Criticism: Responses

“There is unfortunately a huge disjunction between the lectures and the problem sets. This is true both temporally, but in terms of sheer content at any time as well. We may learn about a solution to a problem in lecture, and then have to implement a completely different solution in the problem sets; these are the types of things that are best covered cursorially in lecture and then hammered home in a programming assignment. I would hope that the lectures and assignments line up a bit better in the future.”
(CUE, Spring 2006)

Response:
This is a bug.

We're going to work on sections this year trying to unite the material in lecture and the assignments more directly. However, Matt isn't going to start lecture on OS/161 (that would be hella boring).
“Maybe 10-20% of the effort on any given assignment involved understanding and implementing the essential (theoretical) functionality, and the remaining 80-90% was tracking down errors. This is not a good development cycle”
(CUE, Spring 2006)
“Maybe 10-20% of the effort on any given assignment involved understanding and implementing the essential (theoretical) functionality, and the remaining 80-90% was tracking down errors. This is *not* a good development cycle”  
(CUE, Spring 2006)

Response:  
Ummm... yes, we agree!

However, you should expect to spend a lot of time debugging in this class. Less to the degree that you learn how to do it efficiently (i.e., intelligent use of GDB). We will help you with this.
(People Also Like This Class)

• “If you're in CS and want to graduate with a sense of decency, you must take this course.”

• “Its strength is in the fact that it addresses really relevant topics. The lectures discuss concepts that are fundamental to computing and show up all over the place, and the assignments teach tricky programming skills that are really important for any type of programming.”

• “Anyone who wants to seriously learn how to program on a large scale or is interested in systems needs to take this class.”

• “I would whole heartedly recommend this course. It makes all who take it much better programmers. And as a bonus, the material is ridiculously interesing.”

• “I would absolutely recommend it. I found the material very interesting and engaging, and I think it's very valuable knowledge to have for any CS concentrator.”

• ...etc...
Questions?