CS 253: Intro to Systems Programming

Department of Computer Science
College of Engineering
Boise State University
Topics

- Intro to Team-Based Learning (TBL)
- Syllabus and class logistics
- What is Systems Programming?
Team-Based Learning

- Evidence-based instructional practice proven to increase student motivation and comprehension.
- You are responsible for reviewing material *before* class.
- We make sure you have a basic understanding before we jump into more complicated material.
- You get instant feedback.
- Our class discussion is based on what you need.
- And it's just more fun!
“Random” teams of 3-4 students.

Team work is only in-class (quizzes, exercises).

TBL quizzes

- Quiz over reading assignments and pre-quiz exercises.
- Every week unless specified otherwise.
- Each quiz will be taken first as an individual.
- Same quiz will be taken as a team.
  - Instant feedback, partial credit for the team.
- Keep track of individual vs team performance.

In-class exercises

- Team exercises to make sure you are absorbing material.
- Give you a break from listening to me talk.

Let’s make some teams...
Build Teams

Prioritized Sorting Criteria:

- Do you have any experience programming in C? Number of years?
- Do you have any real-world software development experience (internship or career)?
- Prior experience using Linux/Unix command line and system utilities/scripting?
- Is computer science your first major?
- Have you ever lived outside of Idaho?
- Are you excited to take this class? :-(
Meet team members and introduce yourself!

- Syllabus and practice quiz.
Collaborative Learning using Piazza

“Piazza is a free online gathering place where students can ask, answer, and explore 24/7, under the guidance of their instructors.”

- Piazza invite sent out
- Use Piazza to help each other
- Ask questions anonymously
- Answer questions and doubts that everyone seems to be having
Goals

By taking this course the student will be able to:

- design and develop programs of moderate complexity in C,
- translate their knowledge of object-oriented programming in Java to C,
- use various tools like IDEs, build tools, debuggers, version control and memory checkers to improve their productivity,
- use shell commands and system utilities, and
- use basic system calls related to files, processes and threads.
Where does 253 fit?

- Prerequisite for required class: 453 (Operating Systems)
- Prerequisite for various electives:
  - 425 (Introduction to Computer Networks)
  - 430 (Parallel Computing)
  - 450 (Programming Language Translation)
  - 455 (Distributed Systems),
  - 457 (Artificial Intelligence)
Major topics

- Linux (and Microsoft Windows) programming environments (1 week)
- C programming (7 weeks)
- Programming tools (2 weeks)
- Shell commands and scripts (1 week)
- Basic systems programming in Linux (and Microsoft Windows) (4 weeks)
So, what is systems programming?

<table>
<thead>
<tr>
<th>Layer</th>
<th>People Involved</th>
<th>Classification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application Programs</td>
<td>Application Programmers</td>
<td></td>
</tr>
<tr>
<td>System Utility Programs</td>
<td></td>
<td>Software</td>
</tr>
<tr>
<td>Operating Systems</td>
<td>Systems Programmers</td>
<td></td>
</tr>
<tr>
<td>I/O System (BIOS)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Computer System</td>
<td>Computer Engineers</td>
<td></td>
</tr>
<tr>
<td>CPU</td>
<td>Computer Architects</td>
<td>Hardware</td>
</tr>
<tr>
<td>Memory, logic circuits, etc.</td>
<td>Logic Designers</td>
<td></td>
</tr>
<tr>
<td>Transistors, Diodes, Resistors, etc.</td>
<td>Materials Scientists</td>
<td></td>
</tr>
</tbody>
</table>

Borrowed from: http://cs.lmu.edu/~ray/notes/sysorg/
# Application vs. Systems Programming (1)

<table>
<thead>
<tr>
<th>Applications Software</th>
<th>Systems Software</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deals with abstractions like customers, products, orders, balance sheets, employees, and players in a game.</td>
<td>Deals with concepts extremely close to the hardware level, like registers and memory locations.</td>
</tr>
<tr>
<td>Solves problems of interest to humans, usually in application areas like health care, game playing, finance...</td>
<td>Controls and manages computer systems</td>
</tr>
<tr>
<td>Concerned with anything high-level</td>
<td>Concerned with data transfer, reading from and writing to files, compiling, linking, loading, starting and stopping programs, and even fiddling with the individual bits of a small word of memory</td>
</tr>
<tr>
<td>Is almost always device or platform independent; programs concentrate on general-purpose algorithms</td>
<td>Deals with writing device drivers and operating systems, or at least directly using them; programmers exploit this low-level knowledge</td>
</tr>
<tr>
<td>Applications Software</td>
<td>Systems Software</td>
</tr>
<tr>
<td>--------------------------------------------------------------------------------------</td>
<td>-----------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Is often done in languages like Java, C#, Perl, Python, Ruby, Lisp, JavaScript, or ML, that feature automatic garbage collection and free the programmer from low-level worries</td>
<td>Is often done in assembly language and C, where programmers have to <strong>manage memory themselves</strong>.</td>
</tr>
<tr>
<td>Is done in languages that generally have big fat runtime systems</td>
<td>Generally feature extremely small run-time images, because they often have to run in resource constrained environments</td>
</tr>
<tr>
<td>If done properly, can be very efficient — good garbage collection schemes allow much more efficient memory utilization than the usual memory micro-management common in C programs</td>
<td>If done properly, can be very efficient — you can take advantage of the hardware</td>
</tr>
</tbody>
</table>

Borrowed from: [http://cs.lmu.edu/~ray/notes/sysprog/](http://cs.lmu.edu/~ray/notes/sysprog/)
Application vs. Systems Programming (3)?

Classify the following:

- Word Processor
- Google Chrome
- Linux Operating System
- Device Drivers
- System Utilities
- Database Software
Why C?

- C is the most widely used systems programming language (followed by Java and C++)
- C is low-level and procedural while Java is high-level and object-oriented. Knowing these two languages gives you a strong basis for learning other languages down the road
- Overall, Java and C are the two most commonly used languages in the industry. [http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html](http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html)
- Internship and job interview questions are mostly based on CS 121, 221, 253 and 321.
Working on Programming Projects

- Similar to working on projects in CS 121 and CS 221.
- The GCC C compiler is available on the Linux machines in the labs that can be used directly from the command line.
- We recommend using VIM text editor for C development (see notes on website).
- You may, however, use CLion if you want a full featured IDE.
  - Download from [www.jetbrains.com/clion](http://www.jetbrains.com/clion) you will need to get a student account at [www.jetbrains.com/student](http://www.jetbrains.com/student)
- Try downloading and configuring VIM and/or Clion before next class - Post questions on Piazza, drop in the tutoring center, come to office hours for help.
In-Class Exercise: Hello World

- In your teams, write a valid HelloWorld program in Java.
- How would you compile and run your program from the command-line?
Now, compare and contrast it to this Hello World program written in C.

```c
// This code is in hello.c
#include <stdio.h>

int main(int argc, char *argv[])
{
    printf("Hello world!\n");
    return 0;
}
```
First C Program

Note: main prototype is more relaxed than Java. All of the following are valid.

```c
// produces warning, but compiles (don't use).
main() { }
```

```c
// compiles, but unspecified arguments (don't use).
int main() { }
```

```c
// compiles, specifies no arguments (use this).
int main(void) { }
```
Compiling and Running

- `gcc -Wall helloworld.c`
  - The compiler is called `gcc`, which stands for the GNU C Compiler. It is a free, open source compiler that is widely used.
  - Creates an executable named `a.out`
  - Type `./a.out` to run the program
  - The option `-Wall` asks the compiler to provide all warnings about the code, which can save us a lot of effort later!

- `gcc -Wall helloworld.c -o helloworld`
  - To create an executable called `helloworld`

- Now create a C project in Clion and compile and run the hello world program from Clion
Exercises

- Write, compile and run Hello World in C
- Activate your Piazza account
- Read pages 5-21 of the K&R C book
- Configure VIM and/or Clion on your computer