Welcome to CS4411/5411
Practicum in Operating Systems

"What I cannot create, I do not understand."
— Richard Feynman
Course staff

Instructors

Yunhao Zhang
PhD Candidate

Office hours: Thursday, 6pm-9pm, Gates 437

Lorenzo Alvisi
Tisch University Professor

Teaching Assistants

Justin Lee and Oliver Matte
I wish to bring you the fun of building operating systems.
How to have fun?

• No exam
• No textbook
• 6 coding projects
  • the last one is optional
• 3 running on Linux/Mac
• 3 running on a RISC-V board
• More about these later
Communications

• Website:
  • https://www.cs.cornell.edu/courses/cs4411/2022fa/
• CMSx
  • distribute projects; submit solutions
• Ed Discussion
• For time-sensitive matters: cs4411-staff@cornell.edu
• For sensitive matters: cs4411-prof@cornell.edu
Teamwork

• P0: individually
• P1-P5: teams of 2-3 students
  • real-world softwares are built by teams
  • learn how to collaborate, trust and respect
• P1 will be released on Sep 9
  • two-week time to find teammates
Slip days

• No penalty
• 2 per project, 5 in total
• If you need any accommodation, let us know.
Academic Integrity

• Each team has one submission.
• Do not share code with other teams.
• All submitted code must be your own.
• Put your code in *private* repositories.
• Violations will be prosecuted.
Grading

• No “curve”
• CS4411/5411 is not a competition.
• Final grade is a weighted sum of all the projects.

<table>
<thead>
<tr>
<th>Project</th>
<th>Weight</th>
<th>Project</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>P0</td>
<td>1</td>
<td>P1</td>
<td>2</td>
</tr>
<tr>
<td>P2</td>
<td>2</td>
<td>P3</td>
<td>2</td>
</tr>
<tr>
<td>P4</td>
<td>3</td>
<td>P5</td>
<td>0</td>
</tr>
</tbody>
</table>
Any questions?

Next, demo time.
OS, big and small

• Earth and Grass Operating System (EGOS)
  • written by Robbert van Renesse
  • with rich functionalities: a compiler inside!

• EGOS-2000
  • written by Yunhao Zhang
  • https://github.com/yhzhang0128/egos-2000
  • with very few lines of code: only 2000 in total!
Learning tips

• Earth and Grass Operating System (EGOS)
  • with very rich functionalities
  • Try to explore what functionalities an OS could have
• EGOS-2000
  • with very few lines of code
  • Try to read every line of code
<table>
<thead>
<tr>
<th>What?</th>
<th>Lines of Code (LOC)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SD card driver</td>
<td>222</td>
</tr>
<tr>
<td>Exception Handling</td>
<td>48</td>
</tr>
<tr>
<td>Memory Management</td>
<td>106</td>
</tr>
<tr>
<td>Kernel (scheduler + syscall)</td>
<td>358</td>
</tr>
<tr>
<td>File System</td>
<td>339</td>
</tr>
<tr>
<td>Applications</td>
<td>264</td>
</tr>
<tr>
<td>Library</td>
<td>262</td>
</tr>
<tr>
<td>Makefile</td>
<td>54</td>
</tr>
<tr>
<td>Board-specific tools</td>
<td>172</td>
</tr>
<tr>
<td>Other</td>
<td>175</td>
</tr>
</tbody>
</table>

Read every line of code?
<table>
<thead>
<tr>
<th>What?</th>
<th>Lines of Code (LOC)</th>
<th>Projects</th>
</tr>
</thead>
<tbody>
<tr>
<td>SD card driver</td>
<td>222</td>
<td>P5 (optional)</td>
</tr>
<tr>
<td>Exception Handling</td>
<td>48</td>
<td>P3</td>
</tr>
<tr>
<td>Memory Management</td>
<td>106</td>
<td>P3</td>
</tr>
<tr>
<td>Kernel (scheduler + syscall)</td>
<td>358</td>
<td>P0, P1, P2</td>
</tr>
<tr>
<td>File System</td>
<td>339</td>
<td>P4</td>
</tr>
<tr>
<td>Applications</td>
<td>264</td>
<td>P3</td>
</tr>
<tr>
<td>Library</td>
<td>262</td>
<td>P3</td>
</tr>
<tr>
<td>Makefile</td>
<td>54</td>
<td></td>
</tr>
<tr>
<td>Board-specific tools</td>
<td>172</td>
<td></td>
</tr>
<tr>
<td>Other</td>
<td>175</td>
<td></td>
</tr>
</tbody>
</table>
High-level Keywords

- P0
  - memory and pointer
  - instead of object and reference in Java or Python
- P1
  - context, thread and context-switch
- P2
  - timer interrupt, scheduling and priority
High-level Keywords

- P3
  - privilege level/mode
  - control and status registers (CSR)
  - memory exceptions and system call
- P4
  - layering design: inode layer and directory layer
- P5
  - I/O bus and memory-mapped bus controller
More fun after this semester

- Future work #1
  - A minimal RISC-V processor that can run egos-2000
  - Thanks to Ted Yin and Adrian Sampson

Picture from: https://github.com/Determinant/mriscv
More fun after this semester

- Future work #2
- Graphic User Interface

Homework

• P0 has been released on CMSx
  • Due on Sep. 9
  • Read README, Makefile and queue.h
• Modify and submit queue.c and test_queue.c

• Next lecture
  • Memory and C Programming
Memory and C 101

// standard library for input/output
#include <stdio.h>

int main() {
    printf("Hello World!\n");
    return 0;
}

#include <stdio.h>

int main() {
    printf("Hello World!\n");
    return 0;
}

Compile to machine code
And some read-only data

```c
#include <stdio.h>

int main() {
    printf("Hello World!\n");
    return 0;
}
```
Global variable in the data section

```c
#include <stdio.h>

int global_variable = 0xab;

int main() {
    printf("Hello World!\n");
    return 0;
}
```
Local variable in the stack frame

#include <stdio.h>
int main() {
    int local_variable = 0xcd;
    printf("Hello World!\n");
    return 0;
}
int main() {
    char* str = malloc(14);
    memcpy(str, "Hello World!\n", 14);
    printf("%s", str);
    return 0;
}
Key of C Programming

- Machine code is in the code section
- Variables can be in
  - the read-only data section
  - the data section
  - the stack section
  - the heap section
- The key
  - understand (1) where is the variable and (2) how many bytes