CS 318 Principles of Operating Systems

Fall 2020

Lecture 22: Final Review

Prof. Ryan Huang

JOHNS HOPKINS
WHITING SCHOOL
of ENGINEERING

Course Plug

- **If you enjoy CS 318 topics, consider the advanced OS course ☺**
  - 601.718: Advanced Operating Systems
    - Studying different system structures and different operating systems from design point of view by reading classic and recent papers
    - Focus more on reading, design, and research
    - No labs --- much less load on coding
  - Not offered in Spring 2021 (in lieu of 601.624 Reliable Software Systems)
    - Syllabus of last offering: https://cs.jhu.edu/~huang/cs718/spring20/syllabus.html

Final Mechanics

- **Date & Location: Dec. 16th (Wed), 6-9pm**
  - A request-only alternative slot after 9pm the same day.
- **Open lecture notes**
- **Bulk of the final covers material after midterm**
  - Memory management, file systems, advanced topics
- **Some material on concurrency, synchronization**
  - Synch primitives, basic synch problems
- **Based upon lecture (textbook), homework, and project**
  - Same format as midterm exam
- **Do the homeworks to practice for the exam**

Logistics

- **Exam will be released on Gradescope**
  - Same format as midterm
- **Complete exam in Lockdown browser**
  - Lecture slides will be linked in the exam
    - You can also view lecture slides on another device, but no other resources should be used
  - Other resources besides lecture notes are disallowed
  - Using search engines or discussing with classmates are disallowed
  - Sign honor code
- **A shared Google doc to ask clarification questions**
  - A backup Zoom Q&A session (both links will be provided in the exam)

Overview

- Final mechanics
- **Memory management**
- **Paging**
- **Page replacement**
- **Disk I/O**
- **File systems**
- **The End**

Memory Management

- **Why is memory management useful?**
  - Why do we have virtual memory if it is so complex?
- **What are the mechanisms for implementing MM?**
  - Physical and virtual addressing
  - Partitioning, paging, and segmentation
  - Page tables, TLB
- **What are the policies related to MM?**
  - Page replacement
- **What are the overheads related to providing memory management?**

Virtualizing Memory (1)

- **What are the issues with physical addressing?**
  - Protection, transparency, resource exhaustion
- **What are the goals of virtual memory?**
- **What are the advantages of virtual memory?**
- **What is the difference between a physical and virtual address?**
- **Which component does the translation and management?**

Virtualizing Memory (2)

- **How does load-time linking work?**
  - What are its advantages and disadvantages?
- **How does partitioning work?**
  - Fixed-sized partitioning (base)
  - Variable-sized partitioning (base + bound registers)
  - What are its advantages and disadvantages?
- **What is internal fragmentation?**
- **What is external fragmentation?**

Segmentation

- **What is segmentation?**
- **What is a segment table?**
- **How is virtual address translated with segmentation?**
- **What are its advantages and disadvantages?**
- **How does it compare/contrast with paging?**
- **How can paging and segmentation be combined?**

Paging

- **How is paging different from partitioning?**
- **What are the advantages/disadvantages of paging?**
- **What are page tables?**
- **What are page table entries (PTE)?**
- **Know these terms**
  - Virtual page number (VPN), physical page number (PPN)/page frame number (PFN), offset
- **Know how to break down virtual addresses into page numbers, offset**
- **How have you implemented paging in Pintos?**

Page Table Entries

- **What is a page table entry? (In Pintos?)**
- **What are all of the PTE bits used for?**
  - Modify
  - Reference
  - Valid
  - Protection

Page Tables

- **Page tables introduce overhead**
  - Space for storing them
  - Time to use them for translation
- **What techniques can be used to reduce their overhead?**
- **How do two-level (multi-level) page tables work?**
- **Know how to break down virtual addresses into page directory, page numbers, offset**

TLBs

- **What problem does the TLB solve?**
- **How do TLBs work?**
- **Why are TLBs effective?**
- **How are TLBs managed?**
  - What happens on a TLB miss fault?
- **What is the difference between a hardware and software managed TLB?**

Page Faults

- **What is a page fault?**
- **How is it used to implement demand paged virtual memory?**
- **What is the complete sequence of steps, from a TLB miss to paging in from disk, for translating a virtual address to a physical address?**
  - What is done in hardware, what is done in software?

Advanced Mem Management

- **What is shared memory?**
- **What is copy on write?**
  - Why is CoW useful?
- **When is copy on write used?**
- **What are memory mapped files?**
  - What is the benefit of memory mapped file?
  - What is its drawback?

Page Replacement

- **What is the purpose of the page replacement algorithm?**
- **What application behavior does page replacement try to exploit?**
- **When is the page replacement algorithm used?**
- **Understand**
  - Belady's (optimal), FIFO, LRU, Approximate LRU, LRU Clock, Working Set, Page Fault Frequency
- **What is thrashing?**

Dynamic Memory Allocation

- **What does dynamic memory allocator do and what it cannot do?**
- **What are the decisions to make?**
- **What is the strategy of a best-fit and first-fit allocator, respectively?**
  - What the potential problems for them
- **Why is buddy allocator proposed?**
- **Why is slab allocator proposed?**

Disk

- **Understand the memory hierarchy concept, locality**
- **Physical disk structure**
  - Platters, surfaces, tracks, sectors, cylinders, arms, heads
- **Disk interface**
  - How does the OS make requests to the disk?
- **Disk performance**
  - What steps determine disk request performance?
  - What are seek, rotation, transfer?
- **Disk scheduling: FCFS, SSTF, SCAN, C-SCAN**

File Systems

- **Topics**
  - Files
  - Directories
  - Sharing
  - Protection
  - Layouts
  - Buffer Cache
- **What is a file system?**
- **Why are file systems useful (why do we have them)?**

Files and Directories

- **What is a file?**
  - What operations are supported?
  - What characteristics do they have?
  - What are file access methods?
- **What is a directory?**
  - What are they used for?
  - How are the implemented?
  - What is a directory entry?

File System Layouts

- **What are file system layouts used for?**
- **What are the general strategies?**
  - Contiguous, linked, indexed?
- **What are the tradeoffs for those strategies?**
- **How do those strategies reflect file access methods?**

Unix inodes

- **What is an inode?**
  - How are inodes different from directories?
  - How to use inodes and directories used to do path resolution and find files?
    - Like in homework 5 exercise
- **How Unix inodes enable both efficient access of small files and growth of large files**
  - Direct blocks, in-direct blocks
  - How to calculate file and disk access info given some inode info?
    - Like in homework 5 exercise
- **Where are inodes stored?**
  - What about data blocks?

File Buffer Cache

- **What is the file buffer cache, and why do operating systems use one?**
- **What is the difference between caching reads and caching writes?**
- **What are the tradeoffs of using memory for a file buffer cache vs. VM?**

Protection

- **What is file protection used for?**
- **How is it implemented?**
- **What are access control lists (ACLs)?**
- **What are capabilities?**
- **What are the advantages/disadvantages of each?**

Advanced File Systems

- **What is FFS, and how is it an improvement over the original Unix file system?**
- **What is LFS, and how is it an improvement over FFS?**
- **What is the file system consistent update problem?**
  - What are the possible crash scenarios and consequences?
  - What are the different strategies to do the updates?
  - What problems can the file system checkers (FSCK) fix?
  - What is journaling and its steps?

Advanced Topics

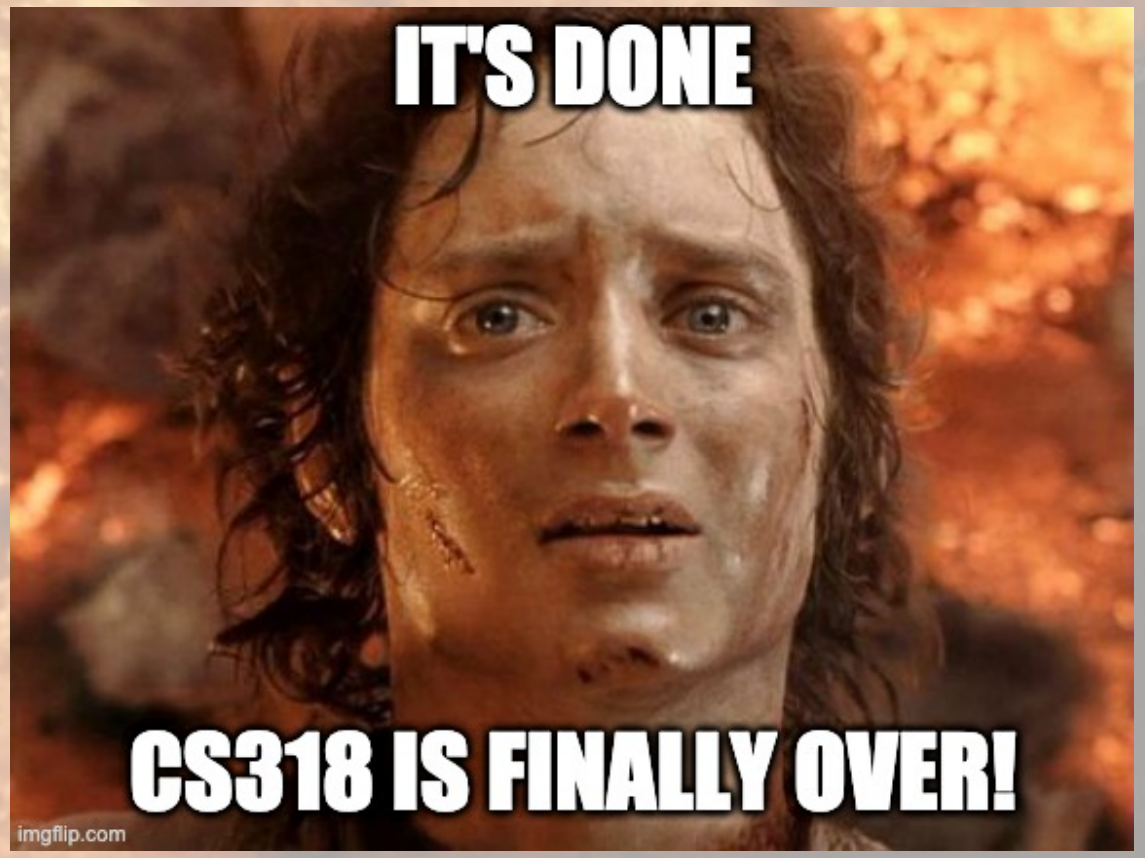
- **What is RPC, how is it implemented and what are the complications to make it work in reality?**
- **What are the design considerations for mobile OS, and how does Android manage apps in terms of processes?**
- **What is the measure for reliability, and how to systematically find bugs in system software?**

Summary

- **Any remaining questions?**

Concluding Remarks

- **Congratulations on surviving CS 318! (+ in the difficult year 2020)**



Concluding Remarks

- **It's a challenging course**
  - It takes courage and hard core to carry through
- **But I hope you found it worthwhile**
  - that it helps improve your system programming skills into the next level

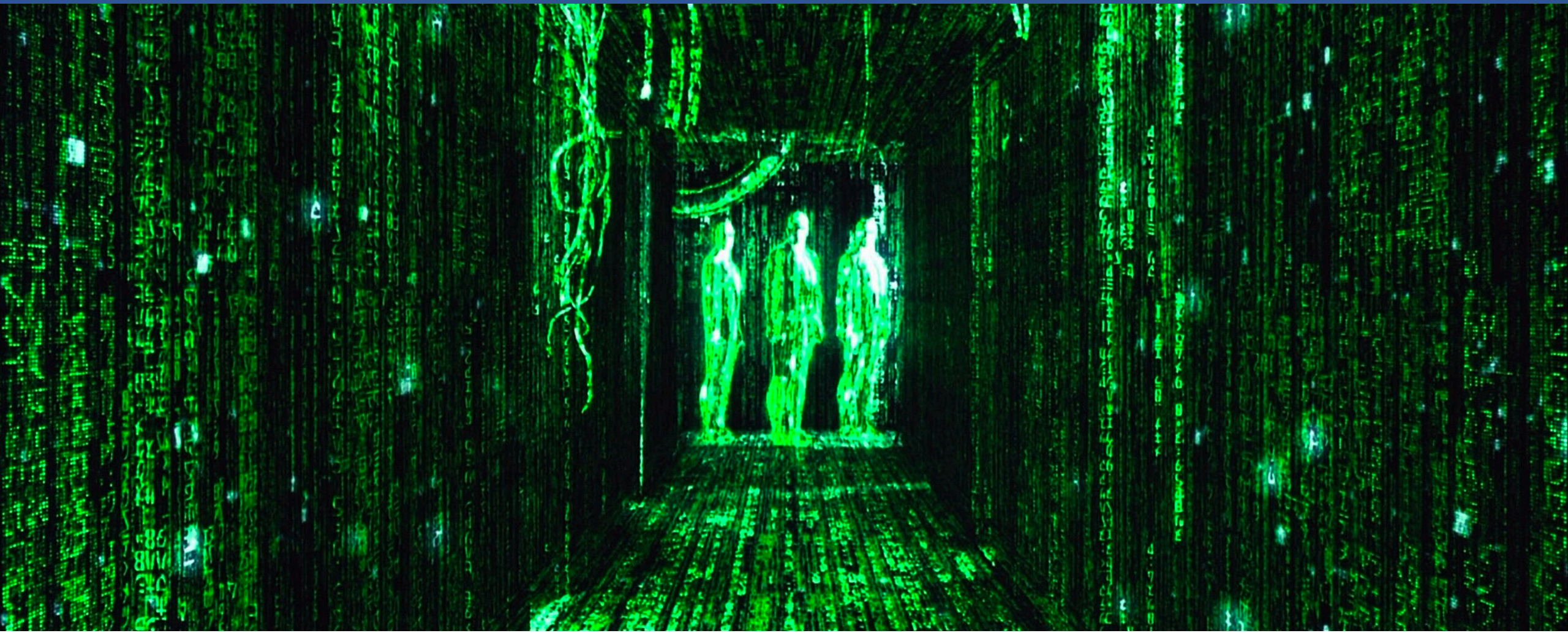
Concluding Remarks

- It's a challenging course
- worthwhile

Concluding Remarks

- **It's a challenging course**
  - It takes courage and hard core to carry through
- **But I hope you found it worthwhile**
  - that it helps improve your system programming skills into the next level
  - … and that you look at (& appreciate) OSes in a new way

Concluding Remarks



Acknowledgement

- **Thanks for sticking with the course to the end**
- **Special thanks to students who regularly attend the class**
  - …and turn on the videos
  - It makes a huge difference when I can see your faces ☺
- **Appreciate the help from Yuzhuo, Haoze, Gongqi and Stephen**
  - This is a challenging course for both students and the CAs
  - They spread the much of the load to make the course possible

Four Take-Away Messages

1. **The devil is in the detail**
   - building systems needs elegant ideas, but just having ideas is far from enough
2. **Never underestimate the power of abstraction & indirection**
   - “All problems in computer science can be solved by another level of indirection”

Four Take-Away Messages

1. **The devil is in the detail**
   - building systems needs elegant ideas, but just having ideas is far from enough
2. **Never underestimate the power of abstraction & indirection**
   - "All problems in computer science can be solved by another level of indirection"
3. **Dare to inspect, modify and rewrite any "mysterious" software**
   - You've gone through the dark side, and few software is as complex as the OS
   - Hack like a champion

Four Take-Away Messages

1. **The devil is in the detail**
   - building systems needs elegant ideas, but just having ideas is far from enough
2. **Never underestimate the power of abstraction & indirection**
   - “All problems in computer science can be solved by another level of indirection”
3. **Dare to inspect, modify and rewrite any “mysterious” software**
   - You’ve gone through the dark side, and few software is as complex as the OS
   - Hack like a champion
   - “Every good work of software starts by scratching a developer's personal itch.”
4. **System thinking**
   - Even if you forget how OS works, I hope you develop the habit of system thinking

Good luck and take care!


2020
1
0