Review

CS503: Operating systems, Spring 2019

Pedro Fonseca
Department of Computer Science
Purdue University

Admin

- Exam: **Thu 05/02, 8:00a - 10:00a @ FRNY B124**
- Course survey: make sure to submit your feedback!

Previous lecture

- DS:
  - Concurrency, fault-tolerance
  - Safety and liveness
- Systems research

Final exam: preparation and style

- Essential:
  - Slides
  - Xinu book chapters
  - Xinu code and labs
- Recommended: Reading the other two recommended books and the suggested material (see slides)
- Expect questions similar to midterm in terms of style

Final exam: scope

- Covers all material covered in the course
  - Will focus more on the material not covered in the midterm, i.e., the memory management lectures and subsequent lectures
- The second part of the course material is mostly discussed in Xinu Chapters 9-15
  - Note that some of the material is not covered in the Xinu book (see prev. slide)

Quick overview

Memory management

- High-level memory manager provides services to other
- Memory is divided into pools of buffers:
  - Enables some level of isolation between services
- Memory allocated to pools is fixed
- Buffers are the same size within each pool
- Using a single pointer to identify the buffers requires some bookkeeping

Virtual memory 1/3

- Paging divides the memory into pages/page frames
- Enables mapping logical addresses into physical addresses at the page granularity
  - High-degree of flexibility for OSs
  - Causes more memory accesses
- Relies on hardware and in-memory configuration structure (page table)
- More on paging next lecture

Virtual memory 2/3

- Virtual memory mechanisms:
  - Paging
  - Segmentation
- Page tables, multi-level paging, PDEs/PTEs, etc.
- TLB
- Memory protection
- Memory layout with virtual memory

Virtual memory 3/3

- On demand paging:
  - Lazy loading of executables
  - Simulating/virtualizing more memory than the available physical RAM
- Intercepting page accesses using the valid bit
- Trapping on the page faults

Device management

- Interrupt vectors
- Scheduling during interrupts
- Device interface; device switch table
- Tty device
- API for devices
- Device independent IO
- Xinu primitives
- Driver examples

Clock and timer

- Device management
- Clock and timer management
  - Timestamp counter and real time counter
  - Timed events: preemption and sleep
  - Operation timeout

High-level message passing

- Xinu offer low-level message passing
  - Only one message outstanding per process
- Xinu high-level message passing
  - Dynamically created ports
  - Number of messages and size fixed when port created
  - Arbitrary senders an receivers
  - Synchronous interface
- Port reset/deletion is tricky because of concurrency
  - Unblocked processes, new processes
  - Use three techniques to handle transition
- Linux IPC

FS 1/2

- Storage terminology
- File system terminology
- Design space
- inodes and metadata
- Typically FS interface and basic steps
  - Mounting file system
  - Open
  - Read/Write
  - Delete
  - Finding files

FS 2/2

- Terminology
- FS design space
- Basic FS:
  - Inodes and metadata
  - Typically FS interface and basic steps
- Block allocation
- Journalling

Virtualization

- VM-level virtualization:
  - Virtualization paper
- Challenge in multiplexing
- Para-virtualization
- Advanced OS features
  - Checkpoint and restart
  - Migration

Virtualization and TEEs

- Hypervisors
  - Type 1 vs. type 2
  - Advantages
- OS-level / process-level virtualization
  - Container abstraction
  - Light virtualization
  - Performance, isolation, security
- Trusted execution environments

Linux 1/2

- History of Linux
- Scale
- Kernel structure
- Process management
- Debugging tools:
  - Debugger, strace, /proc/

Linux 2/2

- Clone system call
- Sys file system
- Virtual memory
- Buddy allocator
- Slab allocator
- BPF
- Micro-kernel vs. monolithic kernel

DS

- DS:
  - Concurrency, fault-tolerance
  - Safety and liveness
- Systems research (not covered)

True false questions

- Q. Each file system has multiple (different) superblocks
- Q. Demand paging enlarges virtual address space
- Q. Paravirtualization requires no modification of guest OSes

General questions

- Q: Describe a key difference between a return instruction (i.e., ret in x86) and an interrupt return instruction (i.e., iret in x86)

- Q: The CPU architecture ABC is designed for low-end IoT devices, so it does not support MMU --- everything runs in physical address space and nothing can be run in virtual address space
  - 1. What should be the limitation of ABC from OS developer's perspective? (elaborate more than simply stating "missing virtual address space")
  - 2. Describe how to support virtual address space for this architecture

More questions

- If a machine has 128 MB physical memory and a 32-bit virtual address space and 4KB pages, what is the size of the page table?

- How many child processes are created if a program executes the following code:

  - ```
    void lots_of_forks(){
       fork ();
       fork ();
       fork ();
    }
    ```
    ```

Summary

- Office hours on Tuesday before the exam
- Good luck with exam!