Operating Systems Notes

Marie Burer

16-1-24

Contents

Course Outline

Information:

- CS:3620 Operating Systems
- Instructor: Guanpeng Li
- Location: 110 MacLean Hall
- Hours: Tuesday/Thursday, 8:00 AM - 9:15 AM
- Prerequesite: CS:2210, CS:2230, and CS:2630 with a minimum grade of C-
- TA: A K M Muhitul Islam

Modality:

- Lectures in person
- Office Hours: Tuesday/Thursday, 2:30 PM - 4:30 PM
- Location: IATL 340

Grading:

- Homework: 40%
  - 6-8 assignments
  - Programming tasks

- In-class activities: 20%
  - Quizzes:
    - Multiple-choice
    - Close book
  - Exercises:
    - Open questions
    - Individual or group
    - BYO laptop
- Final Exam: 40%
  - Technical interview
  - Written exam

Policy:

- 3 days late policy
- Cannot make up quizzes or exercises
- Can discuss homework with colleagues
- Cannot share code
- Cannot copy from internet
- If you copy code from StackOverflow, credit it
- When in doubt, ask the instructor/TA

1 Introduction to OS

What is an operating system?

An operating system (OS) is system software that manages computer hardware and software resources and provides common services for computer programs. (Wikipedia)

- Definition has changed over years
  - Originally, very bare bones
  - Evolved to include more
- Operating System (OS)
  - Interface between the user and the architecture
  - Implements a virtual machine that is easier to program than raw hardware
- Middleware between user programs and system hardware
- Manages hardware: CPU, main memory, I/O devices

OS: Traditional View:

- Interface between user and architecture
  - Hides architectural details
- Implements Virtual machine:

  - Easier to program

- Illusionist
  - Bigger, faster, reliable
- Government
  - Divides resources
  - "Taxes" = overhead

New Developments in OS:

- Operating systems: active field of research
  - Demands on OS's growing
  - New application spaces (Web, Grid, Cloud)
  - Rapidly evolving hardware
- Open-source operating systems
  - Linux etc.
  - You can contribute to and develop OS's
  - Excellent research platform

OS: Salient Features

- Services: The OS provides standard services which the hardware implements

- Coordination: The OS coordinates multiple applications and users to achieve fairness and efficiency
- Design an OS so that the machine is convenient to use and efficient

Why study OS?

- Abstraction: How to get the OS to give users an illusion of infinite resources
- System Design: How to make tradeoffs between performance, convenience, simplicity, and functionality
- Basic Understanding: The OS provides the services that allow application programs to work at all
- System Intersection Point: The OS is the point where hardware and software meet

Not many operating systems are under development, so you are unlikely to get a job building an OS. However, understanding operating systems will enable you to use your computer more efficiently

Build Large Computer Systems

- OS as an example of large system design
- Goals:

2 OS Structure and Architecture

Modern OS Functionality:

- Concurrency
- I/O Devices
- Memory management
- Files
- Distributed systems and networks

OS Principles:

- OS as juggler
- OS as government
- OS as complex system
- OS as history teacher

Protection:

- Kernel mode vs User Mode

  To protect the system from aberrant users, some instructions are restricted to use only by the OS

  Users may not

  - address I/O directly
  - use instructions that manipulate the state of memory
  - set the mode bits that determine user or kernel mode
  - disable and enable interrupts
  - halt the machine

  In kernel mode, the OS can do all of these things

- Hardware needs to support at least both kernel and user

Crossing Protection Boundaries:

System call: OS procedure that executes privileged instructions

- Causes a trap
- The trap handler uses the parameter to jump
- The handler saves caller's state so it can restore control
- The architecture must permit this

Memory Protection:

- Architecture must provide support to the OS to protect user programs and itself
- The simplest technique is to use base and limit registers
- Instantiated by the OS before starting a program

- The CPU checks user reference ensuring it falls between base and limit values

Memory Hierarcy:

Higher = small, fast, more $, less latency

- registers (1 cycle)
- L1 (2 cycle)
- L2 (7 cycle)
- RAM (100 cycle)
- Disk (40,000,000 cycle)
- Network (200,000,000+ cycle)

Process Layout in Memory

- Stack
- Gap
- Data
- Text

Caches

- Access to main memory is expensive

- Caches are small, fast, inexpensive
  - Different sizes and locations:
    * L1: on-chip, smallish
    * L2: next to chip, larger
    * L3: pretty large, on bus

Traps

- Traps: special conditions detected by architecture
- On detecting a trap, the hardware
  - Saves the state of the process
  - Transfers control to appropriate trap handler
    * The CPU indexes the memory-mapped trap vector with the trap number,
    * then jumps to the address given in the vector, and
    * starts to execute at that address
    * On completion, the OS resumes execution of the process

I/O Control

- Each I/O device has a little processor that enable autonomous function
- CPU issues commands to I/O devices
- When an I/O device complete, it issues an interrupt

- CPU stops and processes the interrupt

Memory Mapped I/O

- Enable direct access to I/O controller
- PCs reserve a part of the memory and put the device manager in that memory
- Access becomes very fast

Interrupt-based asynchronous I/O

- Device controller has its own small processor
- Puts an interrupt on the bus when done
- CPU gets interrupt
  - Save critical state
  - Disable interrupts
  - Save state that interrupt handler will modify
  - Invoke interrupt handler using the interrupt vector
  - Restore software state
  - Enable interrupts
  - Restore hardware state