CSCI 3411: Operating Systems

Acknowledgements: Some slide material derived from Silberschatz, et al.
Administrative Fun

- See course webpage (up tomorrow)!
- http://www.seas.gwu.edu/~gparmer/courses/f13_3411
- Forget the URL:
  - google “gabe parmer” or “gparmer”...first link
  - Go to the “courses” tab
  - Click: Fall '13 3411
Administrative Fun II

- Format of Class
  - Lecture
    - Concepts
    - Written HW/tests
  - Lab: Jiguos Song (jiguos@gwu.edu) – there is lab this week!
    - Implementation: C, Linux
    - Programming assignments & final project

- Book(s)

- Grading – hw, exams, participation

- Piazza (search for csci3411) – homework: signup now!
  - Good app

- Academic Honesty
Undergraduate Education

Why are you here???

- In college?
- In CS?
- In OS?
This Semester: Perspective

- You have at least 2 years under your belt.
- Undergraduate – a unique, *limited* opportunity
  - Learning for learning's sake
  - Constant intellectual progress

- Decision time
  - Are you here to get a degree and a job?
  - Are you here to improve as a human being?
This Semester: Perspective

- Question: If you kept on doing what you're doing, *would you be happy with your undergraduate education?*

- This semester
  - challenging
  - opportunity

- *You can do more with your next two years*
  - Take responsibility for your education
“High-level”

Cars

Computers
Cars

Computers

/*
 * So far all flags should be taken in the context of the
 * actual invoking thread (they affect the thread switching
 * _from_ rather than the thread to switch _to_) in which case
 * we would want to use the sched_page flags.
 */
flags = rflags;
switch_thread_update_flags(da, &flags);

if (unlikely(flags)) {
    thd = switch_thread_slowpath(curr, flags, curr_spd, rthd_id, da, &ret_code &curr_sched_flags, &tnd_sched_flags);
    /* If we should return immediately back to this
     * thread, and its registers have been changed,
     * return without setting the return value */
    if (ret_code == CDS_SCHED_RET_SUCCESS && thd == curr) goto ret;
    if (thd == curr) goto_err(ret_err, "snoopown");
} else {
    next_thd = switch_thread_parse_data_area(da, &ret_code);
    if (unlikely(D == next_thd)) goto_err(ret_err, "data_area\n");
    thd = switch_thread_get_target(next_thd, curr, curr_spd, &ret_code);
    if (unlikely(NULL == thd)) goto_err(ret_err, "get target");
}

/*
 * If a thread is involved in a scheduling decision, we should
 * assume that any preemption chains that existed aren't valid
 * anymore. */
break_preemption_chain(curr);
...“low-level”

Wiring diagram for the power supply, starting, charging, engine, front and rear wiper, cigarette lighter, sunroof (G11), sound system and rear demister systems. G10, G11 and G11 Turbo models.
What is an Operating System!??
What is an OS: Where is it?

- Applications (excel, word, browser, ...)
- Operating Systems
- Hardware (CPU, memory, hard drive) “things you can kick”
What is an OS: Where is it?

- Applications (excel, word, browser, ...)
- Operating Systems
- Hardware (CPU, memory, hard drive) “things you can kick”
What is an OS: Analogy

- You!
- Customer$_1$
- Customer$_2$
- Customer$_n$
What is an OS: Analogy
What is an OS: Analogy

Hardware

Operating System

Applications

You!

Customer_1

Customer_2

Customer_n
Operating System as Abstraction

- "The effective exploitation of his powers of abstraction must be regarded as one of the most vital activities of a competent programmer." - Edsger W. Dijkstra

- Abstractions for resources (memory, CPU, disk)
- Environment for application execution
  - Self-centered processes

- Aside: Edsger Dijkstra - Discipline in Thought
OS as Abstraction: System Layers
An x64 processor is screaming along at billions of cycles per second to run the XNU kernel, which is frantically working through all the POSIX-specified abstraction to create the Darwin system underlying OS X, which in turn is straining itself to run Firefox and its Gecko renderer, which creates a Flash object which renders dozens of video frames every second because I wanted to see a cat jump into a box and fall over.

I am a god.
Computers as Distributed Systems

“Hardware: The parts of a computer system that can be kicked.”
- Jeff Pesis
OS as Hardware Manager

- Control a diverse set of hardware
  - Processors
  - Memory
  - Disks
  - Networking cards
  - Video cards
- Coordinates these hardware resources amongst user programs
- OS as a resource manager/multiplexer
History, or How did we get were we are now?

• Bare metal
  • Life cycle:
    – Boot up
    – Run a single application
    – Output result
    – Power down

• OS support for these systems???
History: Batch Systems

• Goal: Maximize amount work done for multiple users

• Applications run one after the other
  • One application at a time!

• Application uses all computer resources

• OS support for batch systems???
History: Batch Multiprogramming

- Multiple applications in memory
- When one waits for I/O, another executes
  - Better utilization of CPU

- *OS support for these systems???
History: Timesharing Systems

- Interactive use of computers
  - Responsiveness matters
  - Expect system to respond to keyboard input immediately
- Several users/applications can share computer
  - Share CPU, Disk, Memory...

- OS support???
Batch vs. Timesharing: Fight!

- Which is more efficient? Which gets more work done?
Batch vs. Timesharing: Fight!

- Which is more efficient? Which gets more work done?
  - Computer work: instructions processed per second
  - Human work: perform operations user requires
History: PCs, Servers, Mobile

- Iterations on timesharing systems
- PCs
  - Less emphasis initially on protection (argh!)
- Servers
  - Throughput oriented
- Mobile
  - Power consumption
iPhone vs. Android

- Original iPhone vs. Android
- Which paradigm does each fall into?
iPhone vs. Android

- Which paradigm does each fall into?
  - Iphone: single user application running at any point in time
    - Back to the 70s
  - Android: multiple applications concurrently execute
    - What happens when memory runs out?
Fundamental OS Concepts

- Abstraction
- Resource management (CPU, RAM, devices)
- Concurrency
- Parallelism
- Protection/Security
- Performance
  - Kernel doesn't do useful work, enables it
Course Objectives

- Explore core ideas in Operating Systems in two ways:
  1) understanding the concepts behind resource management, abstraction, and hardware interface
  2) practical coding experience with a real OS to understand the subtleties and challenges of systems
Why should you care about OSes!?

- Fundamentally: Understand what's going on under the hood
  - "In theory, there is no difference between theory and practice. But, in practice, there is." - Jan L. A. van de Snepscheut/Yogi Berra
- The world runs on systems
  - Microsoft, VMWare, Google (Operating systems, virtual machines)
  - Google, Yahoo, Facebook, Twitter (distributed systems)
  - Boeing, NASA, BMW (embedded/distributed systems)
  - Financial firms (have to spend stimulus money on something!)
  - AppNexus (GWU staffed NYC startup focusing on systems)
- The world is concurrent!
- Industry feedback