Overview

- What this course is about
- How you study this course
- Why you have to take this course
- What you will learn in this course
- What you will earn in this course
What this course is about

- Understand internal structures of operating systems
  - Process management
  - Memory management
  - Storage management
- Prospect state-of-the-art OS technologies
- Explore future directions of OS technologies
Why You Have to Take This Course

- To graduate
- Just for Fun!
- To understand computer systems better
- To obtain useful design methodologies and principles for implementation of complex software
  - 13 of 41 CS Principles stem from OS research
- To make a better computer systems
  - Functionality
  - Performance / Cost
  - Reliability
  - Energy efficiency
Prerequisites

- Undergraduate operating systems
- Undergraduate computer organization
Lecture Topics

- Operating systems concept
- Operating systems design paradigms
- Process management
- Process scheduling
- Power management and energy efficiency
- Interrupt handling and interprocess communications
- Multiprocessing and synchronization
- Kernel memory management
- Virtual memory management
- Cache and buffer management
- I/O and device handling
- Virtual file system and file system concepts
- File system implementations
- Future of operating systems design
Course Components

- Lecture
  - 1.5 hours every week
  - Delivered by professor

- Paper presentation
  - Another 1.5 hours every week
  - Present key papers related to the topic of the week
  - Two papers a week, one classic and one cutting-edge
  - 25 min. presentation + 10 min. discussion
  - You are supposed to present a paper
Grade

- Class participation
  - Discussion + critique
  - 30% of total credit
  - You will submit critique on the papers of the week before the class hour via i-Campus

- Exams
  - Only final exam
  - 30% of total credit

- Research proposal
  - Writing (a short paper – (implementation + evaluation))
  - A combination of your research interests and operating system technology
  - 20% of total credit

- Paper presentation
  - 20% of total credit
Administrative Information

- **Course Code**: ECE5658
- **Class Hour**: Monday, 12:00 PM ~ 14:45 PM
- **Lecture Room**: #21121 (located on 1F in Engineering Bldg. I)
Lecturer

- Euiseong Seo
  - Associate professor, Software and Computer Eng. Dept.
  - E-Mail: euiseong (at) skku.edu
  - Office: #85564
  - Phone: (031) 299-4953
Teaching Assistants

- Cassiano Campes
  - E-Mail: cassianocampes at gmail.com
- Office at #85533
- E-mail is the preferred way to contact
- Make an appointment before you visit
Textbook

- Virtually, there are no textbooks for this class
- You may get some helps to understand Linux kernel internals from Professional Linux Kernel Architecture
  - Written by W. Maruerer
  - Wrox, 2008
References

- Operating System Concepts
  - 9th Edition
  - Written by A. Silberschatz, P. B. Galvin and G. Gagne
  - Published by Wiley
  - 2012
References

- Operating Systems: Internals and Design Principles
  - William Stallings
  - Prentice Hall

- Modern Operating Systems
  - Andrew S. Tanenbaum
  - Prentice Hall
Course Web Page

- [ ] http://csl.skku.edu/ECE5658S17
- [ ] Check the web site regularly
- [ ] Class material will be uploaded to the web page
Remarks

- Presentation is important for all of us
  - Prepare your presentation to the perfection
  - Never deprive your colleagues of their precious time
  - Never reuse slides obtained from Internet as they are

- Think, discuss and think
  - This is a graduate-level course
  - I do not feed you. I just help you to digest.
What are operating systems?
What is an Operating System?

- A program that acts as an intermediary between a user of a computer and computer hardware

- Operating system goals
  - Execute user programs
  - Make computer systems convenient to use
  - Use computer hardware in an efficient manner
Computer System Structure

- Computer system can be divided into four components
  - Hardware – provides basic computing resources
    - CPU, memory, I/O devices
  - Operating system
    - Controls and coordinates use of hardware among various applications and users
  - Application programs – define the ways in which the system resources are used to solve the computing problems of the users
    - Word processors, compilers, web browsers, database systems, video games
  - Users
    - People, machines, other computers
Four Components of a Computer System

- 
  - user 1
  - user 2
  - user 3
  - ...
  - user n

- compiler
- assembler
- text editor
- ...
- database system

- system and application programs

- operating system

- computer hardware
Operating System Definition

- **OS is a resource allocator**
  - Manages all resources
  - Decides between conflicting requests for efficient and fair resource use

- **OS is a control program**
  - Controls execution of programs to prevent errors and improper use of the computer
No universally accepted definition

“Everything a vendor ships when you order an operating system” is good approximation

But varies wildly

“The one program running at all times on the computer” is the kernel. Everything else is either a system program (ships with the operating system) or an application program.
Operating-System Operations

- Interrupt driven by hardware
- Software error or request creates *exception* or *trap*
  - Division by zero, request for operating system service
- Other process problems include infinite loop, processes modifying each other or the operating system
- **Dual-mode** operation allows OS to protect itself and other system components
  - **User mode** and **kernel mode**
  - **Mode bit** provided by hardware
    - Provides ability to distinguish when system is running user code or kernel code
    - Some instructions designated as *privileged*, only executable in kernel mode
    - System call changes mode to kernel, return from call resets it to user mode
Privilege Levels

- 4 levels in x86-32 and x86-64 architecture
- User process cannot directly access I/O devices
System Calls

- Programming interface to control I/O or other restricted resources
- Provided by kernel (OS)
- Implemented by interrupting or trapping instructions
System Calls

Application processes

System call functions

Other kernel functions

Hardware
System Call Handling in User-Level

Diagram:
- **Application**
  - `gettimeofday()` → **Kernel**
  - `strcmp()` → **C Library**

Legend:
- `strcmp()`: String comparison function
- `gettimeofday()`: Get time of day
- **Kernel**: Operating system kernel
- **C Library**: C programming language library

Note: The diagram illustrates the flow of system calls from the application layer to the kernel and the library components.
System Call Handling in User-Level

- What is actually implemented...
Categories of System Calls

- Process control
  - Create, terminate, wait, load, etc.
- File management
  - Create, delete, open, close, read, write, reposition, etc.
- Device management
  - Request, release, read, write, etc.
- Information management
  - Get/set time/date, get process/file/device attributes, etc.
- Communications
  - Create/delete connection, send/receive messages, etc.
- Etc.
Execution Mode

- **User mode**
  - Processes in user mode can access its own instructions and data (but, not kernel instructions and data)

- **Kernel mode**
  - Processes in kernel mode can access kernel and user addresses, and also can execute privileged instructions

- **Kernel distinguishes among processes**

- **Hardware distinguishes mode of execution**

<table>
<thead>
<tr>
<th>Kernel Mode</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>K</td>
<td></td>
<td></td>
<td></td>
<td>K</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>User Mode</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>U</td>
<td>U</td>
<td>U</td>
<td>K</td>
<td></td>
</tr>
</tbody>
</table>
Privileged Instructions

- Machine instructions that can be executed only in kernel mode

Examples
- I/O controls
- Timer management
- Interrupt management
- Switching to user mode
- Etc.

- Manipulate memory management register
  - LGDT, LLDT, LTR, LIDT, SGDT, SLDT, SIDT, STR
- Load and store control registers
  - MOV (CR0~CR4)
- Invalidate cache and TLB
  - INVD, WBINVD, INVLPG
- Performance monitoring
  - RDPMC, RDTSC, RDTSCP
- Fast System Call
  - SYSEXIT, (SYSENTER)
System Call and Execution Mode

User process

- User process executing
- Calls system call
- Return from system call

Kernel

- Trap mode bit = 0
- Execute system call
- Return mode bit = 1

User mode (mode bit = 1)

Kernel mode (mode bit = 0)
Interrupts and Exceptions

- OS is basically interrupt-driven
  - Hardware interrupt
    - Timer
    - I/O
  - Software interrupt
    - System call
    - Exception
Interrupts and Exceptions

- **Interrupts (external events)**
  - Considered to happen between execution of two instr.
  - When kernel receives an interrupt
    - Saves its current context
    - Determines cause of interrupt
    - Services interrupt
    - Restores interrupted context and proceeds as if nothing had happened
Interrupts and Exceptions

- Exceptions
  - Unexpected events caused by a process
  - Happens in the middle of execution of an instruction
- Examples
  - Addressing illegal memory addresses, executing privileged instructions in non-privileged mode, dividing by zero
Operating System Components

- **Kernel**
  - Arch-dependent kernel code

- **System Call Interface**

- **C Library (libc)**

- **User Application**

- **Hardware Platform**

User space

Kernel space
## Operating System Components

### Hardware Components
- **System Call Interface**
  - **File System Management**
  - **Memory Management**
  - **Process Management**
    - scheduler
    - IPC
    - synchronization
  - **I/O Management** (device drivers)
  - **Hardware Control** (Interrupt handling, etc.)

### User Components
- **shell**
- **ls**
- **shell**
- **ps**

### Kernel Components
- **User space**
- **Kernel space**

### Protection

---

**Operating System Components**

- **Hardware**
- **System Call Interface**
- **Process Management**
- **I/O Management** (device drivers)
- **File System Management**
- **Hardware Control** (Interrupt handling, etc.)

**Diagram:**

- **User space**
- **Kernel space**
- **System Call Interface**
- **Protection**
Multiprogramming

- **Batch system**
  - Job must wait for the preceding job to finish
  - One by one

- **Multiprogramming needed for efficiency**
  - Single user cannot keep CPU and I/O devices busy at all times
  - Multiprogramming organizes jobs (code and data) so CPU always has one to execute
  - A subset of total jobs in system is kept in memory
  - One job selected and run via job scheduling
  - When it has to wait (for I/O for example), OS switches to another job

- **Timesharing (multitasking)**
  - CPU switches jobs so frequently that users can interact with each job while it is running, creating interactive computing
  - Response time should be < 1 second
Memory Protection

- Requirements
  - OS must protect user programs from each other
    - Malicious users
  - OS must also protect itself from user programs
    - Integrity and security
Memory Protection

- Simplest scheme
  - Use base and limit registers
  - Base and limit registers are loaded by OS before starting a program
Memory Protection

- **MMU (memory management unit)**
  - Memory management hardware provides more sophisticated memory protection mechanisms
    - Base and limit registers
    - page table pointers, page protection, TLBs
    - virtual memory
    - segmentation
  - Manipulation of memory management hardware are protected (privileged) operations
Process Management

- A process is a program in execution
  - Program is a passive entity
  - Process is an active entity
- Process needs resources to accomplish its task
- Process termination requires reclaim of any reusable resources
- Single-threaded process has one program counter specifying location of next instruction to execute
- Multi-threaded process has one program counter per thread
- Typically system has many processes, some user, some operating system running concurrently on one or more CPUs
Synchronization

- Problems
  - Interrupt can occur at any time and may interfere with interrupted code
  - OS must be able to synchronize concurrent processes

- Synchronization
  - Turn off/on interrupts
  - Use a special atomic instructions
    - read-modify-write (e.g., INC, DEC)
    - test-and-set
    - LOCK prefix in IA32
    - LL (Load Locked) & SC (Store Conditional) in MIPS
I/O Subsystems

- One purpose of OS is to hide peculiarities of hardware devices from the user
- General device-driver interface
- Drivers for specific hardware devices
- I/O subsystem includes
  - Buffering (storing data temporarily while it is being transferred)
  - Caching (storing parts of data in faster storage for performance)
  - Spooling (the overlapping of output of one job with input of other jobs)
Storage Management

- registers
- cache
- main memory
- solid-state disk
- hard disk
- optical disk
- magnetic tapes
Storage Management

- OS provides uniform, logical view of information storage
  - Abstracts physical properties to logical storage unit - file
  - Each medium is controlled by device (i.e., disk drive, tape drive)
  - Varying properties include access speed, capacity, data-transfer rate, access method (sequential or random)

- File-System management
  - Files usually organized into directories
  - Access control on most systems to determine who can access what
  - OS activities include
    - Creating and deleting files and directories
    - Primitives to manipulate files and directories
    - Mapping files onto secondary storage
    - Backup files onto stable (non-volatile) storage media
Design and Implementation of OS not “solvable”
- Some approaches have proven successful

Internal structure of different OSs can vary widely

Start design by defining goals and specifications

Affected by choice of hardware, type of system

User goals and System goals
- User goals – operating system should be convenient to use, easy to learn, reliable, safe, and fast
- System goals – operating system should be easy to design, implement, and maintain, as well as flexible, reliable, error-free, and efficient
OS Design and Implementation

- Important principle to separate
  - Policy: What will be done?
  - Mechanism: How to do it?

- Mechanisms determine how to do something, policies decide what will be done

- Separation of policy from mechanism
  - Allows maximum flexibility if policy decisions are to be changed later

- Specifying and designing an OS is highly creative task of software engineering
- MS-DOS – written to provide the most functionality in the least space
  - Not divided into modules
  - Although MS-DOS has some structure, its interfaces and levels of functionality are not well separated
Operating System Structures

- UNIX System

**Kernel**

- system-call interface to the kernel
  - signals terminal handling
  - character I/O system
  - terminal drivers
  - file system
  - swapping block I/O system
  - disk and tape drivers
  - CPU scheduling
  - page replacement
  - demand paging
  - virtual memory

**Kernel interface to the hardware**

- terminal controllers
- terminals

- device controllers
- disks and tapes

- memory controllers
- physical memory