Introduction
CSE 506: Graduate Operating Systems

Pradipta De
pradipta.de@sunykorea.ac.kr
What is CSE 506 about?

- Graduate Operating Systems
  - To understand OS beyond theory
  - e.g. how memory paging works? … how does it work in Linux, or other OS

- At the end of the course,
  - Good understanding of how OS works
  - Going beyond concepts to delve into OS code
Outline

• Operating System Overview
• OS kernel
  – Booting
  – Program execution
  – How devices work
• Admin and Logistics
  – Lab assignments
  – Exams, Grading, etc.
Operating System: What is it

• Most common view: a software program that manages (and hides complexity) resources and devices from the user
  – In a desktop: manages processor, memory, disk space, network, etc
  – In cellphone: manages much more … battery, screen, dialer and so on
  – In embedded systems: could be managing a specialized piece of hardware

• Collection of code to manage the resources, and libraries to provide interfaces
Types of OS

• Batch OS, Interactive OS

• From perspective of users and tasks:
  – single user, single task: DOS, PalmOS
  – single user, multiple task: Windows Personal Editions
  – Multiple user, multiple task: Linux, Windows Servers

• Real Time OS: RT Linux
## Coarse grain view

<table>
<thead>
<tr>
<th>User Mode</th>
<th>User Applications</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Standard Libs</strong> (shell, commands, system lib)</td>
</tr>
<tr>
<td>Kernel Mode</td>
<td>System call interface</td>
</tr>
<tr>
<td>Hardware</td>
<td><strong>OS Kernel</strong> (memory manager, process manager, filesystem, networking stack)</td>
</tr>
<tr>
<td></td>
<td>Kernel interface to hardware (device drivers)</td>
</tr>
<tr>
<td></td>
<td><strong>Devices</strong> (keyboard, mouse, memory Controllers)</td>
</tr>
</tbody>
</table>
Extend to Virtualization
Assembler, Linker, Loader

C program (*.c) \(\rightarrow\) Compiler \(\rightarrow\) Assembly lang prog (*.S) \(\rightarrow\) Assembler \(\rightarrow\) Machine lang prog (*.o) \\
\(\rightarrow\) Library routine/shared libs \\
\(\rightarrow\) Executable machine lang program (ELF, exe) \\
\(\rightarrow\) Linker \(\rightarrow\) Load into memory \\
\(\rightarrow\) Program(shell) \(\rightarrow\) execve() \(\rightarrow\) sys_execve() \(\rightarrow\) sys_execve() \(\rightarrow\) do_execve() \(\rightarrow\) search_binary_handler() \(\rightarrow\) load_elf_binary() \(\rightarrow\) start_thread()
Keyboard to Console

- Key pressed and released (inside keyboard)
  - Generates a scancode, and sends a (stream of) byte to the keyboard controller chip
- Scancodes assembled into keycodes
  - Keyboard controller managed by kbd driver
  - Conversion of scan code to keycode
- Keycodes converted to TTY input chars
  - Uses kernel maps (fonts are resolved)
Admin and Logistics
• JOS: A bare-bone OS kernel developed at MIT to help teaching
  – We will be using the 64-bit port of JOS implemented at SBU (Don Porter’s initiative)
  – Each lab hands out skeleton code and you implement one OS block
  – Last lab ends up with a (rudimentary) working kernel, which can boot on a real PC
Important Instructions for Lab

• Lab assignments will be emailed to you
  – Details on the webpage
• Get familiar with GIT
  – Checkpointing your VM can be helpful
    • Play with VMPlayer to understand the features
  – **MUST NOT** post any of the assignments, or the solutions handed out, online or share with others
Exams, Grading

• Midterm: 20
• Final: 30
• Lab1: 10 (use this grade as a bonus)
• Lab2: 10
• Lab3: 10
• Lab4: 10
• Lab5: 10
• Lab6: 10

• Late submission:
  – 6 bonus days can be used without penalty
  – After that, each day will be penalized at 10% of total grade.

• Course Schedule
  – Reading list: more pointers to what is presented in class
  – Lecture slides: what is relevant for this course
Other resource

• Please send emails with a subject prefix
  – [CSE 506] <subject>
  – Copy everyone in class if it is a common question
  – Treat this as our discussion group