



Operating System Practice— Introduction

Che-Wei Chang

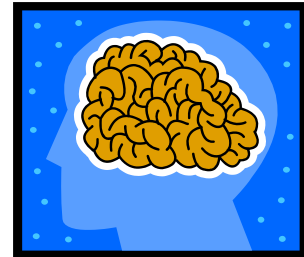
chewei@mail.cgu.edu.tw

Department of Computer Science and Information
Engineering, Chang Gung University

Course Roadmap

Advanced Operating System Concepts

- Concepts and Implementation of File System
- Storage Management and I/O Devices
- System Protection and Security



Exercises on PC and Emulators

- Concepts of the Linux Kernel
- Real-Time System Knowledge
- Android Programming on Android Emulator

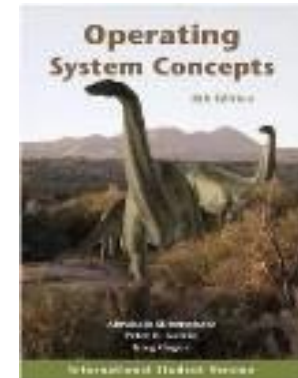
Embedded System Exercises

- Introduction to Embedded System
- Tools and Techniques to Build Embedded System
- Implementation on Embedded System Evaluation Boards



Advanced Operating System Concepts

- ▶ Cover some contents of the textbook
- ▶ Show you some advanced OS techniques
- ▶ Have some quizzes
- ▶ Use midterm and final exam to evaluate your study



Exercises on PC and Emulators

- ▶ Provide some basic knowledge for the Linux kernel
- ▶ Conduct some implementation on virtual machines
- ▶ Understand the Android framework with the Android emulator
- ▶ Have a real case study of real-time embedded systems



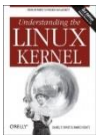
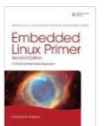
Embedded System Exercises

- ▶ Let you use an evaluation board
- ▶ Let you know the common issues and bugs of using embedded systems
- ▶ Let you use tools for developing an embedded system
- ▶ You should provide two reports for each exercise
- ▶ You can test your ideas on the evaluation board



Syllabus

- ▶ Instructor: **Che-Wei Chang 張哲維**
- ▶ Classroom: **CSIE Seminar Room (6)**
資工系研討室 (6)
- ▶ Class Time: **Thursday 9:10-12:00**
- ▶ TA:
 - 黃聖文 unlimited20r@gmail.com
 - 張哲嚴 tim900430@gmail.com
- ▶ Reference Books:
 - Silberschatz, Galvin, and Gagne, “Operating System Principles,” 10th Edition, John Wiley & Sons, 2018.
 - Christopher Hallinan, “Embedded Linux Primer,” 2nd Edition, Prentice Hall, 2011.
 - Daniel P. Bovet and Marco Cesati, “Understanding the Linux Kernel”, 3rd Edition, O’Reilly, 2005.



Grading and Resources

- ▶ Midterm: 20%
- ▶ Lab Exercise or Homework: 20%
- ▶ Quizzes and Attendance: 20%
- ▶ Final Exam: 20%
- ▶ Final Project: 20%

- ▶ Office Hours: **Thursday 13:30-20:30**
- ▶ Course Website:
- ▶ <https://icechewei.github.io/webpage/teaching.html>



Rules and Requirements

- ▶ It is better to complete the course “Operating System” before you take this course
- ▶ We have closed-book midterm and final exams
- ▶ Teamwork is required for lab exercises and a final project
- ▶ No adjustment for the final grading results
- ▶ We will give the grading results after 18 weeks

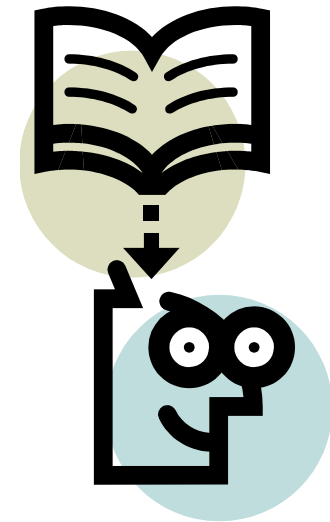




Course Overview

Advanced Operating System Concepts

- ▶ Chapter 10: File System
- ▶ Chapter 11: Implementing File-Systems
- ▶ Chapter 12: Mass-Storage Structure
- ▶ Chapter 13: I/O Systems
- ▶ Chapter 14: System Protection
- ▶ Chapter 15: System Security



File System

- ▶ The Basic Concepts of File System
 - File Concept and Access Methods
 - Disk and Directory Structure
 - File-System Mounting
 - File Sharing and Protection
- ▶ Implementing File Systems
 - File-System Structure
 - Directory Implementation
 - Allocation Methods
 - Free-Space Management
 - Efficiency and Performance



Storage and I/O Systems

▶ Mass-Storage Systems

- Disk Structure
- Disk Scheduling
- Disk Management
- Swap-Space Management
- RAID Structure

▶ I/O Systems

- I/O Hardware
- Application I/O Interface
- Kernel I/O Subsystem
- Transforming I/O Requests to Hardware Operations



System Protection and Security

- ▶ System Protection
 - Principles of Protection
 - Domain of Protection
 - Access Control
- ▶ System Security
 - Security Problems
 - System and Network Threats
 - Cryptography as a Security Tool
 - User Authentication
 - Implementing Security Defenses
 - Firewalling to Protect Systems and Networks
 - Computer-Security Classifications



Flexible Embedded Systems

- ▶ Features of Embedded Systems
 - Customized hardware with high scalability
 - Heterogeneous devices with unified interface
 - Application-aware designs for energy saving

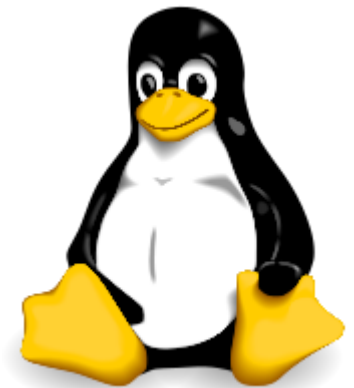


Integrated Hardware and System Software

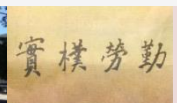
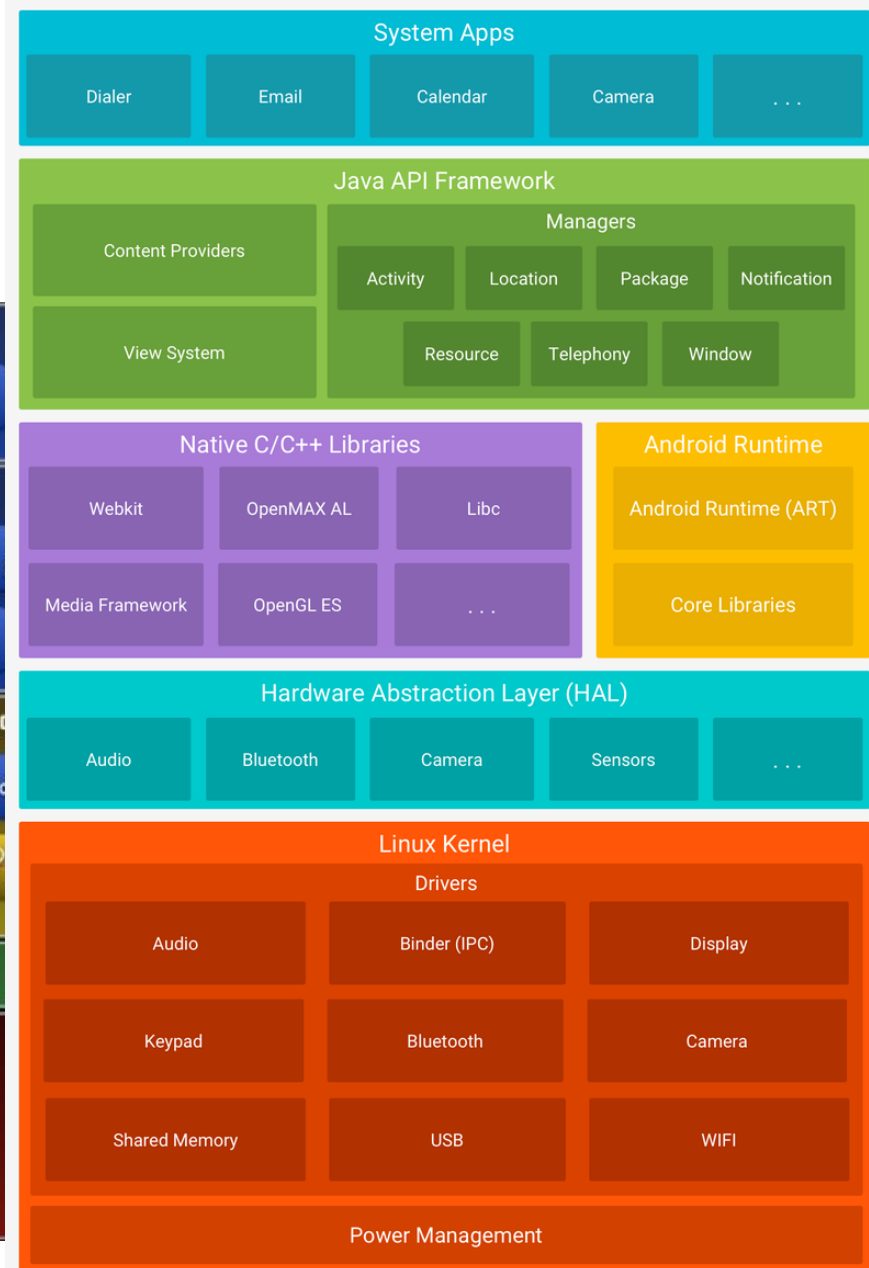
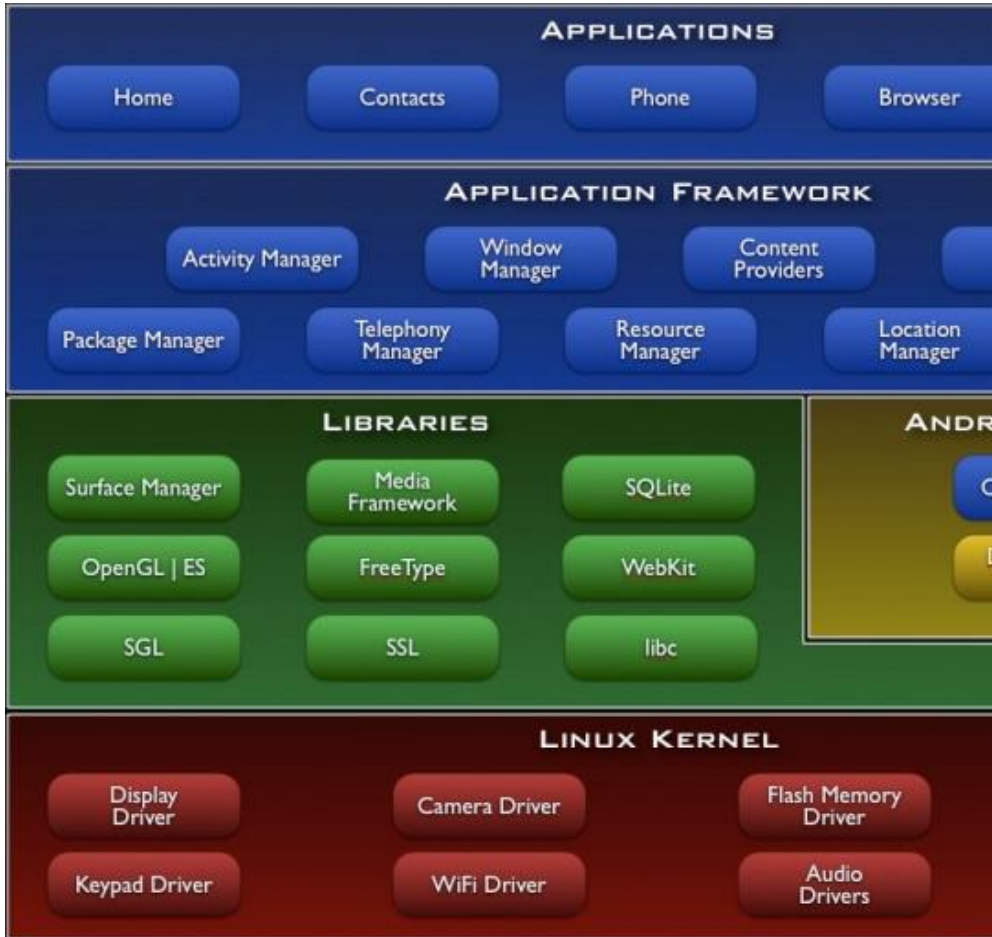


Linux Kernel

- ▶ Open Source: GPL
- ▶ Preemptive Multitasking
- ▶ Virtual Memory System
- ▶ Shared Libraries
- ▶ Demand Paging
- ▶ Dynamic Kernel Modules
- ▶ Shared Copy-on-Write Executables
- ▶ TCP/IP networking
- ▶ SMP Support



Android OS



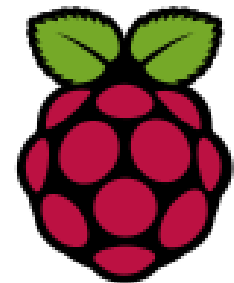
Prototyping Platforms

▶ Arduino

- Is a single-board microcontroller
- Has pre-programmed boot loader
- Is defined as do-it-yourself kits
- Is from Italy: Arduino (*Ar Du Wee No*)

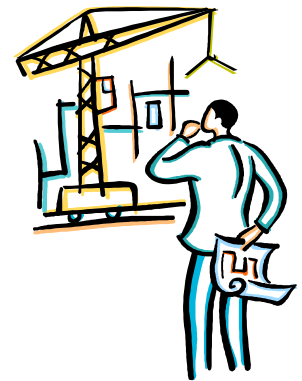
▶ Raspberry Pi

- Is a credit-card-sized single-board computer
- Is promoted for teaching
- Is based on the Linux kernel
- Is from UK



Exercises on Evaluation Boards

- ▶ Link Setup
 - RS-232 on UART for debugging and control interface
 - Ethernet for TFTP and NFS
 - JTAG interface for debugging information
- ▶ System Startup
 - Bootloader
 - Kernel
 - Init process
- ▶ Development with Cross-Platform Toolchains
 - Binary utilities, gcc, glibc
 - Kernel headers setup
 - Binary utility setup



Design Challenge— Optimizing Performance Metrics

- ▶ Obvious Design Goal
 - Construct an implementation with desired functionality
- ▶ Performance Metrics
 - Performance metrics are the measurable features of a system's implementation
 - Simultaneously optimizing numerous design metrics is a challenging issue



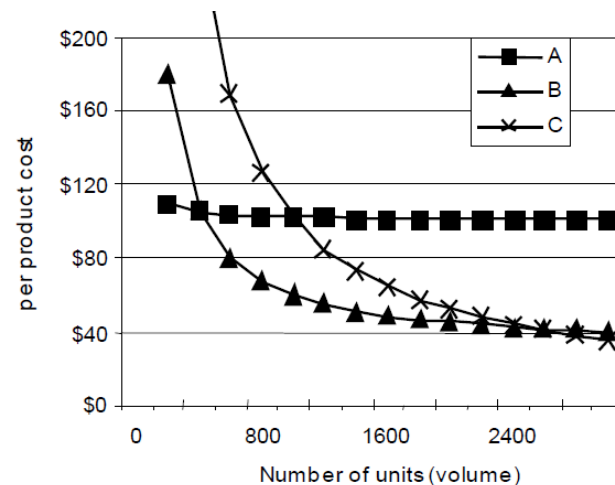
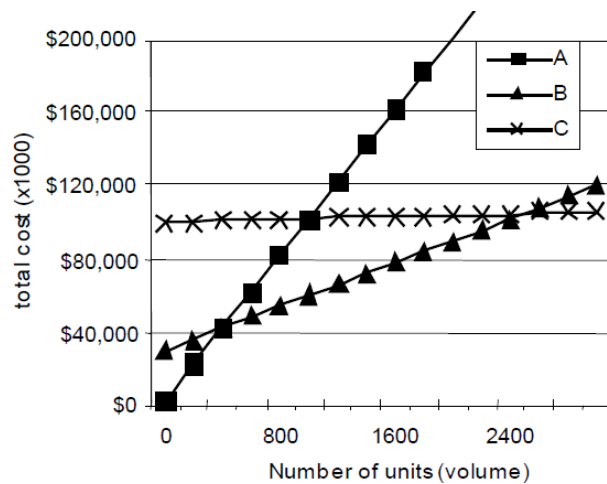
Common Performance Metrics

- ▶ **Unit Cost**: the monetary cost of manufacturing each copy of the system
- ▶ **NRE Cost** (Non-Recurring Engineering cost): the one-time monetary cost of designing the system
- ▶ **Size**: the physical space required by the system
- ▶ **Performance**: the execution time or throughput of the system
- ▶ **Power**: the amount of power consumed by the system
- ▶ **Flexibility**: the ability to change the functionality of the system without incurring heavy NRE cost
- ▶ **Time-to-Market**: the time required to develop a system to the point that it can be released and sold to customers
- ▶ **Maintainability**: the ability to modify the system after its initial release



NRE and Unit Cost

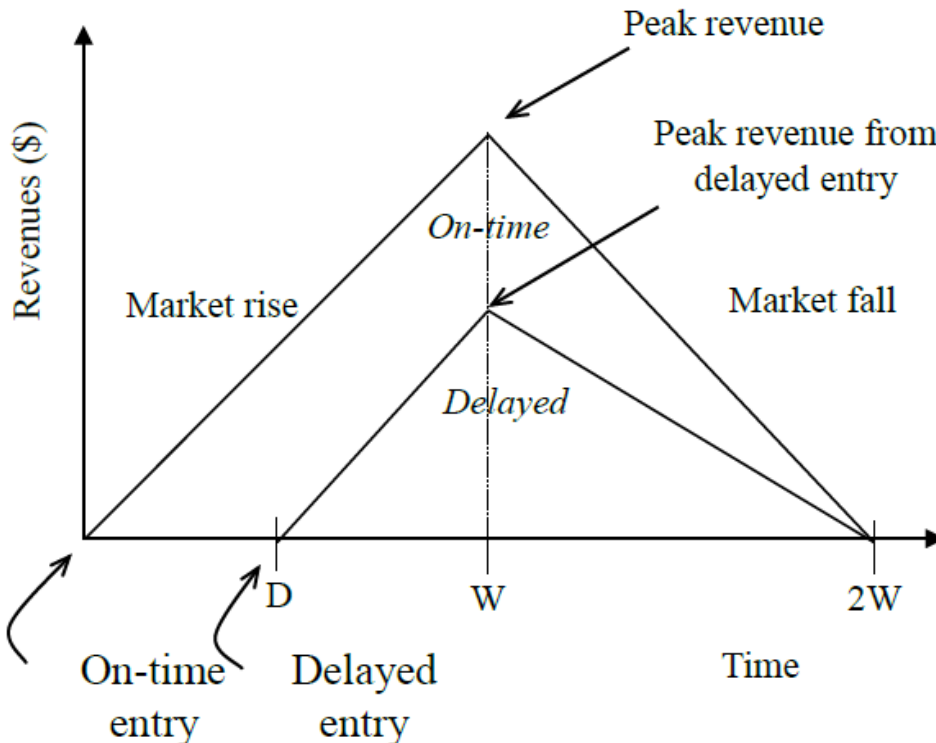
- ▶ Compare Technologies by Costs—the best solution depends on quantity of the product
 - Technology A: NRE=\$2,000, unit=\$100
 - Technology B: NRE=\$30,000, unit=\$30
 - Technology C: NRE=\$100,000, unit=\$2



- We must also consider **time-to-market**



Delayed Market Entry



- ▶ A Simplified Revenue Model
 - Product life = $2W$, peak at W
 - The time of market entry defines a triangle, representing the market penetration
 - The triangle area represents the revenue
- ▶ Loss
 - The difference between the on-time and delayed triangle areas

A Case Study: μ C/OS-II

- ▶ The name is from micro-controller operating system, version 2
- ▶ μ C/OS-II is certified in an avionics product by FAA in July 2000 and is also used in the Mars Curiosity Rover
- ▶ It is a very small real-time kernel
 - Memory footprint is about 20KB for a fully functional kernel
 - Source code is about 5,500 lines, mostly in ANSI C
 - It's source is open but not free for commercial usages
- ▶ Preemptive priority-driven real-time scheduling
 - 64 priority levels (max 64 tasks)
 - 8 reserved for μ C/OS-II
 - Each task is an infinite loop





Any Question?