# Overview of Operating System Concepts

## 1. Introduction

- Modern Computer Hardware Organization
  - Processor(s)
  - Main memory
  - Mass storage
  - Various input/output devices
- What is Operating Systems
  - An Operating System is a program that acts as an intermediary or interface between a user of a computer and the computer hardware.
    - Convenient abstraction of complex hardware devices
    - Protection of access to shared resources
    - Security and Authentication
    - Communication amongst logical entities
  - An Operating System goals
    - Control/execute user or application programs
    - Make the computer system convenient to use
    - Ease the solving of user problems
    - Use the computer hardware in an efficient manner
- Computer System Components
  - Hardware
  - Operating system
  - Application programs (Application systems)
  - Users
- Classical views of Operating Systems
  - Resource manager
  - Control program
  - Command executer

## 2. Computer System Organization and Architecture

- Computer Hardware Architecture
  - *von Neumann* architecture
  - Harvard Architecture
  - Data representation
  - Addressing mode
  - Registers
  - Instruction system
  - Memory system
  - Interrupt controller
  - Input/output controller

  - ✧ Information protecting mechanism
- ■ Computer System Organizations
  - ✧ CPU
  - ✧ Main memory
  - ✧ Bus
  - ✧ Device controller
  - ✧ Device driver
  - ✧ Device
- ■ Interrupt
  - ✧ The basic interrupt mechanism
  - ✧ Asynchronous processing
  - ✧ Interrupt request line
  - ✧ Interrupt handler routine
  - ✧ Interrupt vector
  - ✧ Interrupt controller
  - ✧ Maskable and non-maskable interrupts
  - ✧ Interrupt chaining
  - ✧ Interrupt priority
  - ✧ Interrupt types
    - Traps (Exceptions)
    - External interrupts
    - System calls
  - ✧ Various interrupt attributes.
    - Asynchronous vs. Synchronous.
    - External/Hardware vs. Internal/Software.
    - Implicit vs. Explicit.
- ■ Storage Structures
  - ✧ Registers
  - ✧ Cache
  - ✧ Main memory
  - ✧ Secondary storage (NVM)
  - ✧ Hard disks
- ■ Main Memory Management
  - ✧ Memory management dynamics
  - ✧ Dual-mode operation
  - ✧ Memory Protection
- ■ I/O Structure
- ■ Multiprogrammed batch systems
- ■ Time-Sharing Systems
- ■ Real-Time Systems
- ■ Personal/Desktop Computers
- ■ Multiprocessor Systems
  - ✧ SMP
  - ✧ Multicore

- Clustered Systems
- Networked Systems
- Distributed Systems
- Web-based Systems
- Handheld & Mobile Systems

## 3. Functional Views of Operating Systems

- Computer Dynamics
  - Instruction cycle with interrupts
  - External interrupt
  - Interrupt handler
  - Interrupt-driven I/O
  - Interrupt timeline of CPU and I/O device
  - Synchronous and asynchronous I/O methods
  - Direct memory access (DMA)
  - System calls
- Hardware Protection
  - Dual-mode operation
    - Monitor mode/Kernel mode
    - User mode
    - Privileged instructions
    - User mode to kernel mode transfer
      - System call
      - Interrupt
      - Trap
  - I/O protection
    - System calls for I/O protection
  - Memory protection
    - Base register (relocation)
    - Limit register (bounded)
  - CPU protection
    - Timer
- Fundamental OS Concepts
  - Thread
  - Address space
  - Process
  - Dual-mode operation
- Resource Management
  - Process management
  - Memory management
  - File management
  - Mass-storage management
  - Cache management
  - I/O management

- Free and Open Source Operating Systems
  - ✧ GNU project
  - ✧ GNU/Linux
  - ✧ BSD UNIX

## 4. Structures of Operating Systems (1)

- Operating System Services
  - ✧ A user oriented view
  - ✧ User interface
  - ✧ Program execution
  - ✧ I/O operations
  - ✧ File systems
  - ✧ Communication
  - ✧ Error detection
  - ✧ Resource allocation
  - ✧ Accounting
  - ✧ Protection and security
- Common Operating System Components
  - ✧ A system oriented view
  - ✧ Process management
  - ✧ Main memory management
  - ✧ File management
  - ✧ Mass-storage management
  - ✧ I/O management
  - ✧ Networking
  - ✧ Command interpreter
  - ✧ GUI
  - ✧ Protection and security
  - ✧ Error detection and response
  - ✧ Accounting
- System Calls and APIs
  - ✧ Relationship of API, system call and OS
  - ✧ Standard C library and POSIX
- System Programs
- Operating System Design and Implementation

## 5. Structures of Operating Systems (2)

- Structure/Organization/Layout of OS
  - ✧ Monolithic (one unstructured program)
    - Traditional UNIX system structure
    - Linux system structure
    - Google's Android
  - ✧ Layered
    - Win3.1

- OS/2
- ✧ Microkernel
  - Architecture of a typical microkernel
  - Mach 3.0
  - MAC OS
  - MINIX
  - Windows NT, XP, 7.0
  - QNX Neutrino RTOS
- ✧ LKMS
- ✧ Virtual Machines

## 6. Introduction to Process (1)

- Basic Concepts
  - ✧ Segments in a process
    - Text, Data, Heap and Stack
  - ✧ Context of a process
  - ✧ Process virtual memory (PVM) in IA-32
  - ✧ Attributes of process
    - Process ID
    - Parent process ID
    - User ID
    - Process state
    - Process priority
    - Program counter
    - CPU registers
    - Memory management information
    - I/O status information
    - Access control
    - Accounting information
- Process Table and Process Control Block
  - ✧ Process table and its contents
  - ✧ PCB and its contents
- Process States and Transitions
  - ✧ Running, ready and waiting/blocked states
  - ✧ Process states transitions
  - ✧ Five-states process model
- Operations on Process
  - ✧ Process creation
    - Details in creating a process
    - Programming: forking a separate process
    - Programming: vforking a sharing space process
  - ✧ Process termination
    - Why terminating a process
    - Details in terminating a process

        - Zombie process
        - Orphan process
        - Programming: forking without waiting child termination
        - Programming: forking with waiting child termination
        - Programming: vfork, execv and wait
        - Programming: vfork, execv without wait

## 7. Introduction to Process (2)

- Unix and Linux Examples
    - UNIX SVR4 process states model
    - Linux process representation
    - Linux process state
- Process Scheduling
    - What is process scheduling
    - Scheduling queues
        - Job queue
        - Ready queue
        - Device/Waiting queues
    - Types of process schedulers
        - Long-term scheduler
        - Medium-term scheduler
        - Short-term scheduler
    - Process swapping
    - Process scheduling activities
    - Process scheduling queues
- Process Switching
    - Context switching (steps)
    - Mode switching

## 8. Inter-process Communication (1)

- Overview
    - Independent process and cooperating process
    - Why cooperating
        - Information sharing
        - Computation speedup
        - Modularity
        - Convenience
    - Two fundamental models of IPC
        - Shared memory and message passing
- Shared-memory Systems
    - Producer-consumer problem with shared-memory
        - A producer and a consumer
        - Concurrency
        - Unbounded and bounded buffer

- Linux IPCs Limits
- Linux shared memory
  - System calls
  - Programming: Single-writer-single-reader problem
- POSIX shared memory
  - POSIX APIs for memory sharing
  - Programming: Producer-consumer problem

## 9. Inter-process Communication (2)

- Message-passing Systems
  - Primitives for message-passing
  - Variable and fixed message size
  - Message format
    - Linux message structure
  - Logical communication link
    - Direct or indirect communication
    - Synchronous or asynchronous communication
    - Automatic or explicit buffering
  - Synchronization in Message-passing Systems
    - Blocking and unblocking
  - Buffering
  - Linux message passing
    - System calls
    - Programming: Sending & receiving process demo
  - POSIX message passing
    - POSIX APIs for memory sharing
    - Lab work: Sending & receiving process demo

## 10. Inter-process Communication (3)

- Pipes
  - Ordinary pipes
  - Naked pipes
  - Linux ordinary pipes
    - System calls
  - Linux named pipes
    - System calls
    - Programming: single pipe buffer
    - Programming: total pipes capacity
    - Programming: ordinary pipe
    - Programming: named pipe between parent and children
    - Programming: named pipe between two arbitrary Processes

## 11. Inter-process Communication (4)

- Communications in Client-Server Systems

- Sockets
- Linux socket programming
  - Socket APIs
  - Programming: socket-server-BBS
- Remote Procedure Calls

## 12. Threads (1)

- Overview
  - Tasks/Processes vs. Threads
  - Benefits of multithreaded programming
    - Responsiveness
    - Resource sharing
    - Economy
    - Scalability
  - Thread states
    - Running
    - Ready
    - Blocked
    - No suspend state
  - Terminating a thread
- Multicore Programming
  - Parallelism and concurrency
  - Multithreading
  - Programming challengs
    - Balance
    - Data splitting
    - Data dependency
    - Testing and debugging
    - Identifying tasks
  - Data parallelism and task parallelism
- User and Kernel Level Threads
  - Thread libraries
    - POSIX Pthreads
  - User level threads (ULT)
    - Kernel activity for ULTs
    - Advantages and inconveniences of ULTs
  - Kernel level threads (KLT)
    - Linux threads
    - Advantages and inconveniences of KLTs
  - Hybrid ULT/KLT Approaches
- Multithreading Models
  - Many-to-one model,
  - One-to-one model, and
  - Many-to-many model.

- ✧ Two-level model
- ✧ Light-weight process (LWP)

## 13. Threads (2)

- Threads Libraries
  - ✧ POSIX Pthreads
    - POSIX Pthreads APIs
    - Programming: pthread creating
    - Programming: pthread memory sharing
    - Programming: pthread setting stack
- Implicit Threading
  - ✧ Threads pools
    - Lab work: Implementation of a threads pool
  - ✧ OpenMP
    - Programming: OpenMp demo
    - Programming: OpenMP matrix adding
- Threading Issues
  - ✧ fork() and execv()
    - Programming: pthread forking demo
  - ✧ Signal handling
    - Linux system calls
    - Programming: sigaction-demo
  - ✧ Thread cancelation
    - Asynchronous cancellation
    - Deferred cancellation

## 14. Threads (3)

  - ✧ Thread local storage (TLS)
    - Programming: TLS implemented by __thread in language level
    - Programming: TLS implemented by pthread_key_create
    - Programming: tls_key and bounding data structure
- Linux clone()
  - ✧ Constants with flags in clone()
    - Programming: Linux clone() demo
    - Lab work: Programming with Linux clone()
  - ✧ Windows Threads

## 15. Cooperating Processes

- Introduction
  - ✧ Process cooperation
  - ✧ Data consistency
- Race Condition
  - ✧ Race condition

- Critical section
- Atomic operations
- The Critical-Section Problem
  - The critical-section problem
  - Three essential criteria to solve the critical-section problem
    - Mutual exclusion
    - Progress
    - Bounded waiting
  - Preemptive and Non-preemptive Kernels
  - Types of solutions to critical-section problem
    - Software-based solutions
    - Hardware-based solutions
    - Operating system solutions
    - Programming language solutions
  - Software solutions to critical-section problem
    - Algorithm 1-5
    - *Peterson*’s algorithm
    - *Peterson*’s solution for n processes
    - *Lamport*'s Bakery algorithm for n processes
    - *Eisenberg-McGuire*'s algorithm for n processes
    - Lab work: *Peterson*’s algorithm

## 16. Process Synchronization (1)

- Synchronization Hardware
  - Atomic (non-interruptible) hardware instructions
    - test_and_set()
    - compare_and_swap()
  - Mutual exclusion with hardware instructions
- Mutex Lock
  - Concept of mutex lock
- Semaphores
  - Concepts
    - Semaphore
    - Semaphore operations
      - wait(S) and signal(S)
      - P and V operations
    - Solution to critical-section problem with semaphores
  - Implementation of semaphore
    - Binary Semaphore Implementation
    - Counting Semaphore Implementing
  - Deadlock and starvation with semaphore
  - Incorrect use of semaphores
- Solution to Classical Problems with Semaphores
  - Classical Problems

- Bounded-buffer problem
- Readers-writers problem
- Dining-philosophers problem

## 17. Process Synchronization (2)

- Monitors
  - The monitor type
  - Monitor usage
  - Monitor condition type
  - Monitor condition variables
  - Monitor condition operations
    - cwait(x) and csignal(x)
  - Solution to the dining-philosophers problem with monitor
  - Monitor implementation using semaphores
- Deadlock
  - What is deadlock (system model)
  - Four necessary conditions of deadlock
  - Resource allocation graph
  - Methods for handling deadlock
    - Preventing and Avoiding
    - Detecting and Recovering
    - Ignoring
  - The Banker's algorithm
    - Safety algorithm
    - Resource-request algorithm
    - Limitations of Deadlock Avoidance
    - Programming: the Banker's algorithm

## 18. Process Synchronization (3)

- Synchronization Examples
  - Linux gcc __sync atomic family
    - Programming: demo for Linux __sync_** type
  - POSIX mutex locks
    - Programming: demo for POSIX mutex lock
  - POSIX unnamed semaphore
    - Programming: demo for POSIX unnamed semaphores
  - POSIX named semaphore
    - Programming: demo for POSIX named semaphores
  - POSIX Synchronization
    - Programming: Solution to multi-producer-multi-consumer problem with POSIX semaphores and shared memory

## 19. CPU Scheduling (1)

- Basic Concepts

- CPU-I/O burst cycle
- CPU burst time
- When to make CPU scheduling
- preemptive scheduling
  - during system call
  - during interruption
- Dispatcher and dispatcher latency

■ CPU Scheduling Criteria
- Scheduling goals of different systems
  - All systems
  - Batch systems
  - Interactive systems
  - Real-time systems
- Scheduling criteria
  - CPU utilization
  - Throughput
  - Turnaround time
  - Waiting time
  - Response time

■ Simple Scheduling Algorithms
- First-come, first-served scheduling
- Shortest-job-first scheduling
- Priority scheduling

■ Advanced Scheduling Algorithms
- Round-robin scheduling
- Multiple-priority queues scheduling
- Multilevel feedback queue scheduling
- Thread scheduling

## 20. CPU Scheduling (2)

■ Multiple-Processor Scheduling
- Processor affinity
  - Soft affinity
  - Hard affinity
- Load balancing
  - Push migration
  - Pull migration
  - Load balancing vs. processor affinity
- Multicore processors
  - Memory stall

■ Real-Time CPU Scheduling
- Soft real-time and hard real-time systems
- Minimizing Latency
  - Event Latency

- Interrupt latency
- Dispatch latency
- ✧ Priority-based scheduling
- ✧ Rate-monotonic scheduling
- ✧ Earliest-deadline-first scheduling
- ✧ Least Laxity First algorithm
- ✧ Proportional share scheduling
- ✧ POSIX real-time scheduling
  - Programming: demo for POSIX real-time scheduling

- Linux Scheduling
  - ✧ Completely Fair Scheduler
  - ✧ RTLinux
- Algorithms Evaluation
  - ✧ Deterministic modeling
  - ✧ Queuing models
  - ✧ Simulations
  - ✧ Lab work: CPU scheduling simulations

## 21. Introduction to Memory Management

- Basic Concepts
  - ✧ Hardware address protection
  - ✧ Address binding
  - ✧ Logical address space and physical address space
  - ✧ Memory-mapped unit (MMU)
  - ✧ Relocation register
  - ✧ Hardware support for relocation and limit registers
  - ✧ Hardware translation of address
  - ✧ Dynamic linking
  - ✧ Swapping
  - ✧ Program size vs. memory size
- Memory Management Requirements
  - ✧ Relocation
  - ✧ Protection
  - ✧ Sharing
  - ✧ Logical organization
  - ✧ Physical organization
- Memory Partitioning
  - ✧ Contiguous allocation
  - ✧ Real memory management techniques
  - ✧ Fixed partitioning
    - Dynamics of fixed partitioning
    - Fragmentation
  - ✧ Variable partitioning
    - Internal fragmentation

        - External fragmentation
        - Compaction
        - Basic placement algorithms: First-fit, Next-fit, Best-fit and Worst-fit
    - ✧ Buddy system

## 22. Memory Segmentation and Paging

- ■ Simple Segmentation
    - ✧ User view of a program
    - ✧ Dynamics of simple segmentation
        - Segment table
        - Base & limit
        - Segment-table base register (STBR)
        - Segment-table length register (STLR)
    - ✧ Address translation
    - ✧ Protection
    - ✧ Sharing
- ■ Simple Paging
    - ✧ Non-contiguous to avoid external fragmentation
    - ✧ Page and page frame
    - ✧ Free-frame list
    - ✧ Logical address structure
    - ✧ Page table
    - ✧ Page-table base register (PTBR)
    - ✧ Page-table length register (PTLR)
    - ✧ Translation Look-aside Buffer (TLB)
    - ✧ Address translation
    - ✧ Paging hardware with TLB
    - ✧ Protection
    - ✧ Sharing
    - ✧ Structure of page tables
        - Hierarchical page tables
        - Hashed page tables
        - Inverted page tables
- ■ Examples

## 23. Virtual Memory and Demand Paging (1)

- ■ Virtual Memory
    - ✧ Concept of virtual memory
    - ✧ Virtual memory addressing
    - ✧ Virtual address space
    - ✧ Shared libraries with virtual memory
    - ✧ Process execution with virtual memory
- ■ Demand Paging

- ✧ Concepts of demand paging
- ✧ Page table
- ✧ Present-bit and modified-bit
- ✧ Page fault
    - Steps in handling a page fault
- ✧ Page replacement
    - Steps in handling a page replacement
- ✧ Demand paging with TLB
- ✧ Stages in demand paging (worse case)
- ✧ Performance of demand paging

■ Demand Paging Considerations
- ✧ Locality
- ✧ Thrashing
- ✧ Memory-mapped files
- ✧ Buddy system
- ✧ Slab system
- ✧ Copy-on-write
- ✧ Other issues

## 24. Virtual Memory and Demand Paging (2)

■ Page Replacement Algorithms
- ✧ Basic page replacement
- ✧ FIFO algorithm
- ✧ Optimal page replacement
- ✧ Least recently used (LRU) algorithm
    - Counter implementation
    - Stack implementation
    - Hardware matrix LRU implementation
- ✧ LRU approximation algorithms
    - Reference bit and reference byte
    - Second-chance algorithm (CLOCK algorithm)
    - Enhanced second-chance algorithm
- ✧ Cleaning policy
- ✧ Lab work: page replacement algorithms

■ Combined Segmentation and Paging

■ Virtual Memory Policies

## 25. Mass Storage Systems

■ Introduction
- ✧ Hard disk Structure
    - Disk Head and head crash
    - Track, section and cylinder
    - Host controller and Hard disk controller
    - Transfer rate and positioning time

    - Hard disk performance
    - Logical block
  - Hard disk attachment
- Disk Scheduling
  - Data access time
  - I/O request queues
  - First-come first-serve (FCFS)
    - Zigzag effects
  - Shortest seek time first (SSTF)
  - Elevator algorithms
    - SCAN scheduling and C-SCAN scheduling
    - LOOK scheduling and C-LOOK scheduling
  - Lab work: disk scheduling algorithms
- Disk Management
  - Swap-space Management
  - RAID structure
- Stable-Storage Implementation

## 26. File System Interface

- Basic Concepts
  - Files
  - File attributes
  - File type extensions
  - File operations
  - Access methods
    - Sequential access
    - Direct access
  - Types of file systems
- File Directories
  - Elements
  - Operations
  - Single-level directory
  - Two-level directory
  - Tree-structure directories
  - DAG directories
  - General graph directory
  - File system mounting
- File Sharing and Protection

## 27. File System Implementation

- File-System Structure
  - File control block (FCB)
  - File system layers
    - I/O control level

        - Basic file system
        - File organization module
        - Logical file system
    - On-storage structures
        - Boot control block, volume control block, directory structure, FCB
    - In-memory structures
        - Mount table, directory-structure cache, system-wide open-file table, open-file table, buffers
    - Directory Implementation
        - Linear list
        - Hash table
- Allocation Methods
    - Contiguous allocation
    - Linked allocation/Chained allocation
        - File allocation table (FAT)
    - Indexed allocation
        - One block scheme
        - Linked scheme
        - Two-level index scheme
    - Combined schemes
    - Performance analysis
- Free-Space Management
    - Bitmap with n blocks
    - Free-block list (Linked list)
    - Grouping and counting
    - Space maps
- Efficiency and Performance
- Recovery
- NFS
- Example: WAFL File System

## 28. I/O Systems

- Overview
- I/O Hardware
    - Port, Bus, PCI, PCIe, Expansion bus, Controller (host adapter)
    - Typical PC bus structure
    - Memory mapped I/O (MMIO)
    - I/O device-control registers
    - Polling (steps)
    - Interrupt
        - Interrupt-request line
        - Interrupt handler routine
        - Maskable and non-maskable interrupts

        - Interrupt vector
        - Difference among interrupt, fault, trap and exception
    - ✧ Direct memory access (DMA)
        - Cycle stealing
        - DMA steps
- ■ Application I/O Interface
    - ✧ Kernel I/O structure
    - ✧ I/O devices vary in many dimensions
        - Data transfer mode
        - Access mode
        - Transfer schedule
        - Sharing
        - Speed of operation
        - I/O direction
- ■ Kernel I/O Subsystem
    - ✧ Services provided by kernel's I/O subsystem
        - Scheduling
        - Buffering
        - Caching
        - Spooling
        - Device reservation
        - Error handling.
    - ✧ I/O protection
    - ✧ I/O kernel structures
- ■ Power management
- ■ Transforming I/O Requests to Hardware Operations
    - ✧ 10 Steps
- ■ **STREAMS
- ■ Performance Analysis

== The End