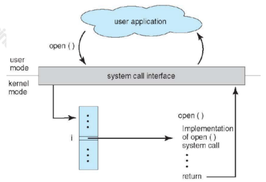
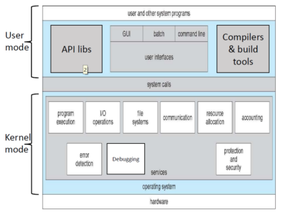
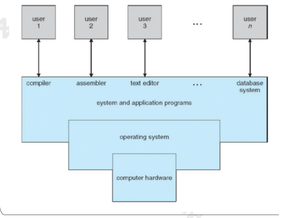
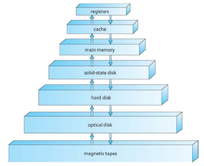
- Intro to OS
  - OS design
    - highest level, design is affected by hardware and type of system
      - distributed? realtime? batch?
      - users goal and system goal
        - OS timer for preemptive processes is a mechanism, the exact tick time is policy
        - mechanism & policy: mehcanism: how to do it, policy: what specially will be done
  - system program: provide a convenient environment for program execution, interface to system calls
    - file management
    - status info
    - file modification
    - programming language support
      - compilers, assemblers, debuggers and interpreters
    - prgram loading and execution
    - communication
    - application program
      - dont pertain to system
  - system call
    - programming interface to the services provided by OS, accessd by high level API libraries rather than direct system call
    - 

    - a number associated with each system call, system call interface maintains an indexed table according to these numbers, caller dont know how system call implemented and details, just need to obey API and the result call
    - parameter passing
      - 1. simplest, pass parameters in reg, but regs are limited and parameters may be more than reg
      - 2. parameters stored in a block in memory and addr of block passed as a parameter in a reg
      - 3. parameters pushed onto the stack by program and popped off the stack by OS
    - types of system calls
      - process control
        - create, end, load, wait for time/event...!! debugger, locks
      - file management
      - device managment
      - information
        - get time, date, system data
      - communications
        - in message passing model, processes send and receive message to host name/ process name(pid), typically for short message
        - in shared memory model, processes can access to memory regions owned by other processes, for large message
      - protection
  - user OS interface
    - commands are built into shell
      - CLI
      - command are name of other programs that execute the command
      - GUI
      - touchscreen
  - OS services
    - API library, compilers and build tools, user interface, program execution, I/O operations
      - communication: via shared memory or through message passing(packets or messages moved by OS)
      - debugging
      - error detection
      - file-system manipulation
    - provide environment and services to programs and users
      - 

    - provide functions to eusure the efficient and secure operation of system
      - resource allocating, including CPU cycles, main memory, file storage, I/O device
      - accounting:keep track of which user/process use how much resources
      - protection and security, protection: ensuring all access to system resources is controlled, security: requires authentication from outside of system, defending external I/O devices
  - OS operations: OS is invoked via interrupts and thus is interrupt driven
    - hardware interrupt: timer interrupt caused by an on-chip timer, used to preempt app and invoke OS kernel on regular intervals(OS tick, 1-50ms)
    - software interrupt: exception, OS services are requested by trap instruction
    - duel mode operation(user mode and kernel mode)
      - mode bit to distinguish user mode or kernel mode
      - some instructions designated as priviledged and some memory location may be accessible in kernel mode
      - system call(using trap instruction) change mode to kernel mode
        - kernel sets timer to prevent infinit loop, for the next interrupt before scheduling a process
      - return from a system call by reseting mode bit back to user mode
  - OS structure
    - multiprogramming batch system: a bit historical, a job selected and run via job scheduling, none preemptive=cooperative multitasking
    - timesharing/interactive system: logical extension in which CPU switches jobs that users interact with each job -> interactive computing, preemptive
    - layered approach: OS is divided into a number of layers/levels, each built on top of lower layers, layers are selected such that each uses functions/operations and services of lower level layers
    - microkernel
    - modules
      - kernel has a set of core components linked to additional services via modules
      - modules talk to each other over known interfaces
      - loaded as needed within kernel, preferred
      - similar to layers but with more flexibility
      - linux has loadable modules primarily for device drivers and file systems
- Computer system structure
  - hardware
  - app program
  - OS
    - program as an intermediary between app/system programs and the computer hardware
      - commad-line interface(CLI) and graphic user interface(GUI) may be considered system program
      - app and system programs may be refered to as user programs(run in user/unpriviledged mode), while OS run in priviledged mode(kernel mode)
    - excute user programs(easier to use) and allocate hardware resource(efficient and secure manner)
      - user view: convenient usage and good responsiveness, dont care much about resource utilization
      - system view: OS is a resource allocator: manage resources and decide when conflicts for efficient and fair resource use
      - control program: prevent errors and volatile command
      - 

    - Def: no universally def, the program running all time is the kernel, kernel runs while the CPU is in privileged mod(access all memory location and all hardware resource), else runs CPU in user mode is either app program or system program
  - 3 components of computer system organization(Von Neumann model)
    - CPU(s)
      - control unit, ALU, CPU register(RISCvs CICS)
      - machine code -> assembly code -> c source code
    - memory
      - program and jobs/processes: a process is a program loaded into the system memory(machine code+data)
      - main memory:control unit only fetch instructions from main memory /access main memory directly: RAM, ROM secondary memory: CPU access this memory indirectly via I/O controllers
      - storage hierarchy
      - 

      - caching: information use is copied from slower to faster storage temporarily
      - direct memory access structure: used for high speed I/O device(transmit info close to memory speed), transfer data without CPU intervention
    - device controllers=I/O devices/controllers
      - I/O operation:I/O devices and CPU can execute concurrently, each I/O device in charge of a particular device and have local buffer
      - CPU communicate with I/O by instructing to control reg, reading status reg, move data from main memory to local buffer and vise versa.
      - CPU waits till either of two operations, polling: read continuouosly the status till operation completes, interrupt CPU goes off and exectues other tasks
      - device driver for each I/O controllers to manage I/O, provide uniform interface between controller and kernel
      - interrupts, exception and traps
        - interrupt caused by I/O controller(hardware interrupts)
        - exception caused usually by illegal operation(/0), unaccessible memory and invalid code, usually called trap(software-generated interrupts)
        - when exception occurs, control transferred to the corresponding interrupt service routine(ISR)
        - interrupt vector table
        -

        - interrupt vector may contain branch of instruction to the location of ISR or the entry point for ISR
        - interrupt handling ISR resides within the OS's kernel and perserve the state of CPU reg: saving program counter(PC) reg, and usually have hardware support
    - computer startup
      - bootstrap program: excute immediately after power-up or reboot
      - stored in ROM(ROM+FLASH common today)
      - initialize necessary aspect of system, load OS kernel from non-volatile storage
    - computer system architecture
      - single purpose processor(embedded system)
      - multiprocessor(PC, server)

中心主题