Computer structure

- CPU: Genral purpose processor for program/OS execution
- Memory: RAM
- Storage: Disk
- IO: Keyboard, Monitor
- Connected by "bus".
- Each IO device has a "dedicated" "internal" processing unit -- IO device controller.

Computer IO (Input Output)

- Synchronous IO: CPU is waiting for IO to complete.
  - Hw technique: Polling
- Asynchronous IO: CPU is not waiting for IO to complete (doing some other task)
  - Hw technique: Interrupts
  - OS maintains a device status table to keep track of IO devices (busy/idle) and processes waiting for those IO devices.

Interrupt Processing

- IO event is sensed by IO device controllers.
- It will be conveyed to CPU as a special signal - Interrupt.
- CPU pause current execution and execute interrupt handler.
- "Interrupt handler" will get address of "ISR" (from IVT) and execute ISR.
- When ISR is completed, execution resumes where it was paused.

Maskable vs Non-maskable interrupt

- Maskable -- Interrupts can be disabled temporarily i.e. their processing (ISR) can be delayed.
  - Lower priority (w.r.t. Non-Maskable)
  - Most of hardware interrupts are Maskable.
  - e.g. 8085/86 -- INTR pin, ARM7 -- irq/fiq
- Non-Maskable -- Interrupts cannot be disabled i.e. processing must be done immediately.
  - High priority
  - Special hardware interrupts are Non-Maskable. Usually these interrupt indicate severe failure.
  - e.g. 8085/86 -- NMI pin, ARM CM3 -- NMI

Hardware vs Software interrupt

- Hardware -- interrupts from hardware peripherals.
- Software interrupt
  - Special instructions (Assembly/Machine level) when executed, current execution is paused, interrupt handler is executed and then the paused execution resumes.
  - Arch specific:
    - 8085/86: INT
    - ARM 7: SWI
    - ARM Cortex: SVC
  - Also called as "Trap" in few architecture.

Interrupt Controller

- Convey the interrupts from various peripherals to the CPU.
- Also manage priority of the interrupt (when multiple interrupts arrives at same time).
- e.g. 8085/86 <-- 8259, Modern x86 processors (apic), ARM-7 (VIC), ARM-CM3 (NVIC), ...
- In few arch, interrupt controllers are programmable (e.g. ARM), while in few arch it is not programmable (e.g. AVR).
    - PIC can config priorities of the interrupts.
    - Non-programmable IC priorities of interrupts are fixed

CPU Scheduling

- Modern OS are time-sharing systems i.e. CPU time is allocated to each process and after that time the next process is scheduled on the CPU.
- Timer Hardware (PIT/SysTick) is used periodically to generate the interrupt.
- When interrupt is arrived, interrupt handler is executed which in turn invokes ISR.
- Then interrupt handler invokes scheduler.
- Scheduler check if time allocated to current process is completed and if completed then decide the next process to be executed (using some scheduling algorithm).
- The selected process's execution context is then restored into CPU by the dispatcher and the next process continues to execute.
- The whole process is also referred as "Context Switch".

System Calls

- Software interrupt is used to implement OS/Kernel services.
- Functions exposed by the kernel so that user programs can access kernel functionalities, are called as "System calls".
    - e.g. Process Mgmt: create process, exit process, communication, synchronization, etc.
    - e.g. File Mgmt: create file, write file, read file, close file, etc.
    - e.g. Memory Mgmt: alloc memory, release memory, etc.
    - e.g. CPU Scheduling: Change process priroty, change process CPU affinity, etc.
- System calls are specific to the OS:
    - UNIX: 64 syscalls e.g. fork(), ..
    - Linux: 300+ syscalls e.g. fork(), clone(), ...
    - Windows: 3000+ syscalls e.g. CreateProcess(), ...

Classification of OS

- OS can be categorized based on the target system (computers).
    - Mainframe systems
    - Desktop systems
    - Multi-processor (Parallel) systems
    - Distributed systems
    - Hand-held systems
    - Real-time systems

Desktop systems

- Personal computers -- desktop and laptops
- User convinence and Responsiveness
- Examples: Windows, Mac, Linux, few UNIX, ...

Handheld systems

- OS installed on handheld devices like mobiles, PDAs, iPODs, etc.
- Challenges:
  - Small screen size
  - Low end processors
  - Less RAM size
  - Battery powered
- Examples: Symbian, iOS, Linux, PalmOS, WindowsCE, etc.

Realtime systems

- The OS in which accuracy of results depends on accuracy of the computation as well as time duration in which results are produced, is called as "RTOS".
- If results are not produced within certain time (deadline), catastrophic effects may occur.
- These OS ensure that tasks will be completed in a definite time duration.
- Time from the arrival of interrupt till begin handling of the interrupt is called as "Interrupt Latency".
- RTOS have very small and fixed interrupt latencies.
- RTOS Examples: uC-OS, VxWorks, pSOS, RTLinux, FreeRTOS, etc.

Distributed systems

- Multiple computers connected together in a close network is called as "distributed system".
- Its advantages are high availability (24x7), high scalability (many clients, huge data), fault tolerance (any computer may fail).
- The requests are redirected to the computer having less load using "load balancing" techniques.
- The set of computers connected together for a certain task is called as "cluster". Examples: Linux.

Mainframe systems

Resident Monitor

- Early (oldest) OS resides in memory and monitor execution of the programs. If it fails, error is reported.
- OS provides hardware interfacing that can be reused by all the programs.

Batch Systems

- The batch/group of similar programs is loaded in the computer, from which OS loads one program in the memory and execute it. The programs are executed one after another.
- In this case, if any process is performing IO, CPU will wait for that process and hence not utilized efficiently.

Multi-Programming

- In multi-programming systems, multiple program can be loaded in the memory.
- The number of program that can be loaded in the memory at the same time, is called as "degree of multi-programming".
- In these systems, if one of the process is performing IO, CPU can continue execution of another program. This will increase CPU utilization.
- Each process will spend some time for CPU computation (CPU burst) and some time for IO (IO burst).
    - If CPU burst > IO burst, then process is called as "CPU bound".
    - If IO burst > CPU burst, then process is called as "IO bound".
- To efficiently utilize CPU, a good mix of CPU bound and IO bound processes should be loaded into memory. This task is performed by an unit of OS called as "Job scheduler" OR "Long term scheduler".
- If multiple programs are loaded into the RAM by job scheduler, then one of process need to be executed (dispatched) on the CPU. This selection is done by another unit of OS called as "CPU scheduler" OR "Short term scheduler".

Multi-tasking OR time-sharing

- CPU time is shared amoung multiple processes in the main memory is called as "multi-tasking".
- In such system, a small amount of CPU time is given to each process repeatedly, so that response time for any process < 1 sec.
- With this mechanism, multiple tasks (ready for execution) can execute concurrently.
- There are two types of multi-tasking:
    - Process based multitasking: Multiple independent processes are executing concurrently. Processes running on multiple processors called as "multi-processing".
    - Thread based multi-tasking OR multi-threading: Multiple parts/functions in a process are executing concurrently.

Process

- Process is program in execution.
- Process has multiple sections i.e. text, data, rodata, heap, stack. ... into user space and its metadata is stored into kernel space in form of PCB struct.
- PCB contains
    - id, exit status,
    - scheduling info (state, priority, time left, scheduling policy, ...),
    - files info (current directory, root directory, open file descriptor table, ...),
    - memory information (base & limit, segment table, or page table),
    - ipc information (signals, ...),
    - execution context, kernel stack, ...

Thread

- Threads are used to execute multiple tasks concurrently in the same program/process.
- Thread is a light-weight process.
    - For each thread new control block and stack is created. Other sections (text, data, heap, ...) are shared with the parent process.
    - Inter-thread communication is much faster than inter-process communication.
    - Context switch between two threads in the same process is faster.

- Thread stack is used to create function activation records of the functions called/executed by the thread.

Multi-user

- Multiple users can execute multiple tasks concurrently on the same systems. e.g. IBM 360, UNIX, Windows Servers, etc.
- Each user can access system via different terminal.
- There are many UNIX commands to track users and terminals.
    - tty, who, who am i, whoami, w

Multiprocessor systems

- The systems in which multiple processors are connected in a close circuit is called as "multiprocessor computer".
- The programs/OS take advantage of multiple processors in the computer are called as "Multiprocssing" programs/OS.
    - Windows Vista: First Windows OS designed for multi-processing.
    - Linux 2.5+: Linux started supporting multi-processing.
- terminal> lscpu
    - sockets = 1 (single processor chip on motherboard)
    - cores per socket = 4 (number of physical cores -- CU + ALU + Registers)
    - threads per core = 2 (number of hardware threads -- simultaneous multi-threading(SMT) or hyper-threading(HT))
    - logical cores = number of sockets * physical cores * threads per core = 1 * 4 * 2 = 8
- Modern PC architectures are multi-core arch i.e. multiple CPUs on single chip.
- Since multiple tasks can be exeuted on these processors simultaneously, such systems are also called as "parallel systems".
- Parallel systems have more throughput (Number of tasks done in unit time).
- There are two types of multiprocessor systems:
    - Asymmetric Multi-processing
    - Symmetric Multi-processing

Asymmetric Multi-processing

- OS treats one of the processor as master processor and schedule task for it. The task is in turn divided into smaller tasks and get them done from other processors.

Symmetric Multi-processing

- OS considers all processors at same level and schedule tasks on each processor individually.
- All modern desktop systems are SMP.