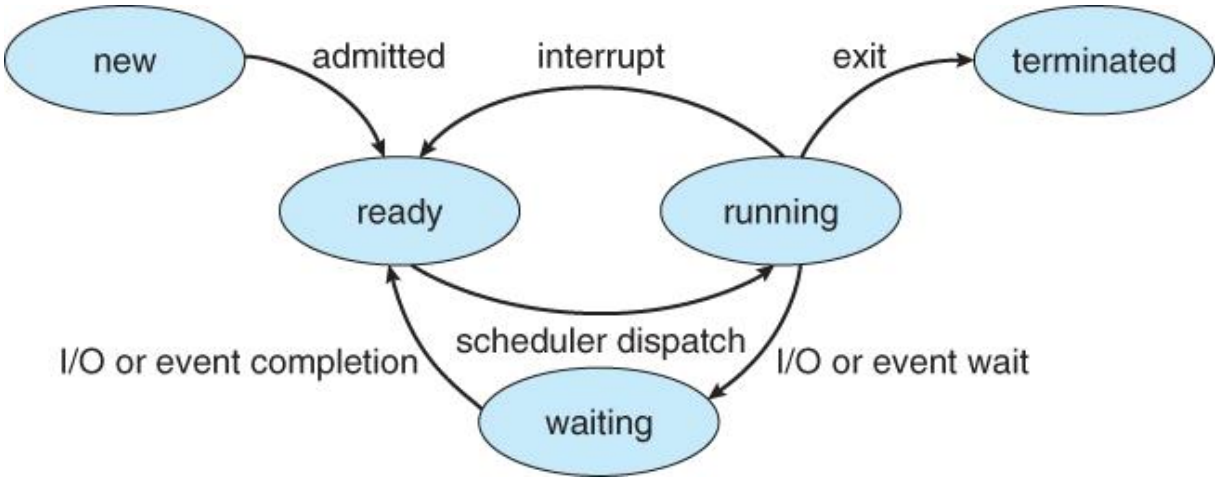
## Concepts and Terms

- Real mode – Real mode is a memory-addressing scheme and operating state for computer microprocessors. In real mode, the memory that can be accessed by a program — usually random-access memory (RAM) — is not managed or buffered in any way by the hardware, software or basic input and output services (BIOS). This means a program is able to access all reachable memory addresses, regardless of what the memory is being used for, and must manage all aspects of reading and writing to memory locations by itself. Several restrictions come with using real mode, including the fact that the amount of accessible memory is limited to 1 megabyte, because the processor in this mode allows addresses to be only 20 bits in length.
- Physical address – The physical address is the post-translated address, it's an address of some page in physical memory (main memory). It is divided into **frames**.
- Logical address – The address at which an item (memory cell, storage element, network host) appears to reside from the perspective of an executing application program. Logical address is in the virtual memory, the address that exists in the secondary memory (pre-translated). It is divided into **pages**. A logical address may be different from the physical address due to the operation of an address translator or mapping function.
- Base address – An absolute address that acts as a reference point for other addresses.
- 512 bytes – Size of the boot block.
- 0x55 0xaa – Magic signature, last two bytes that ends the boot sector.
- Bootstrap – To boot or to load a program into a computer using a much smaller initial program to load in the desired program, which is usually an OS.
- Segment selector – Special pointer that identifies a segment in memory.
- Parse – Where a string of commands – usually a program – is separated into more easily processed components, which are analyzed for correct syntax and then attached to tags that define each component. The computer can then process each program chunk and transform it into machine language.
- Padding – Bits or characters that fill up unused portions of a data structure.
- ELF (Executable and linking format) – Fundamental file formats.
    - ELF Header – Structure that contains the metadate of the file.

- Program Header – Each segment is defined by a small program header structure. All program headers are in an array directly behind each other in the ELF file. The ELF header specifies; The location of the program header table (e_phoff), The size of each program header entry (e_phentsize), The total number of entries (e_phnum)
    - Segments – Only relevant at runtime. Points to an absolute address in memory and specify how many bytes are contained in the segment. – Also specify where and how they should be loaded into virtual or physical memory. "Tells the OS how to map parts of the ELF file into memory."
        - Data segment – Initialized globals and other initialized data. Most cases the mapped data segment in memory will be larger than the segment in the ELF file. This is used to specify room for uninitialized data.
        - Code segment – Contains the executable code. The OS will load the entry point where the binary will start its execution.
        - Dynamically linked binary (not always used) – Specifies which shared libraries need to be loaded.
- Section Header – Sections are defined by a section header, that are arranged in an array in memory called the section header table. The ELF header contains the same information about the Program Header and the Section Header
    - Sections – Only relevant at link time. Points to an absolute address in memory and specify how many bytes are contained in the section. – If an ELF file contains both segments and sections, some segments will contain multiple sections

- Process - A program under execution i.e., an active program
- Threads - A thread is a lightweight process that can be managed independently by a scheduler.
- Context switching - Context Switching involves storing the context or state of a process so that it can be reloaded when required and execution can be resumed from the same point as earlier. This is a feature of a multitasking operating system and allows a single CPU to be shared by multiple processes.
- Thread root

- User Thread
- Independent threads – not state shared with other threads: deterministic -> input state determines result. Reproducible -> can recreate starting conditions I/O. Scheduling order doesn't matter.
- Cooperating threads – shared state between multiple threads. Non-deterministic. Non-reproducible, sometimes there is an error other time there is not.
- Kernel Thread
- Multiprocessing – multiple CPUs
- Multiprogramming – Multiple jobs or processes, the scheduler is free to run any thread in any order.
- Multithreading – multiple threads per process
- Process states – The different states a process can be in, and the behavior of each state.




- CPU Scheduler – Whenever the CPU becomes idle, the operating system must select one of the processes in the ready queue to be executed.
- Dispatcher – Is the module that gives control to the CPU to the process selected by the Scheduler. The time it takes for the dispatcher to stop one process and start another running is known as the Dispatch latency.
- CPU-scheduling may take place under the following four circumstances:
    1. When a process switches from the running state to the waiting state
    2. When a process switches from the running state to the ready state (for example, when an interrupt occurs)
    3. When a process switches from the waiting state to the ready state (for example, at completion of I/O)

4. When a process is terminated
    - For situations 1 and 4 (non-preemptive), there is no choice in terms of scheduling. A new process (if one exists in ready queue) must be selected for execution. However, there is a choice for situations 2 and 3 (preemptive).
- Scheduling criteria –
    - CPU utilization – minimize the time the CPU is spent in idle.
    - Throughput – processes completed for a time unit.
    - Turnaround time – the time it takes for a process to complete as seen from the process point of view.
    - Waiting time – The sum of the periods spent waiting in the ready queue.
    - Response time – The time it takes for the user to notice a completed process (opening a video file, it takes some time for the file to open, but is theoretically opened instantly in the CPU)
- Process Control Block – Each process is represented in the operating system by a PCB. A PCB can contain things like:
    - Process ID – unique ID that identifies that process.
    - Process State – The state a process can be in at that moment.
    - Program counter – The address for the next instruction that must be executed for that process.
    - CPU Registers – The registers that are used by a process.
    - CPU Scheduling information – Has the priority of the processes, pointer to the scheduling queue, and other scheduling parameters
    - Memory management information – the memory that is being used by a process.
    - Accounting information – resources that are being used by a process.
    - I/O status information – I/O devices that are being assigned to a process.
- Interrupt descriptor table (IDT) – The IDT is used by the processor to determine the correct response to interrupts and exceptions. Use of the IDT is triggered by three types of events: hardware interrupts, software interrupts, and processor exceptions, which together are referred to as *interrupts*.
- Deadlock – Is a situation where a set of processes are blocked because each process is holding a resource and waiting for another resource acquired by some other process.

- Race conditions – A race condition occurs in the software when the computer program depends on the threads or processes of a program. In software, we cannot debug the race condition because the final result is non-deterministic and based on the timing of multiple threads.
- Atomic operation – Program operations that run completely independently of any other processes.
- Locks – Lock is either taken or not. If lock is not taken, claim the lock and proceed. If lock is taken, wait/block until lock is available. (binary semaphore?)
- Semaphores – Semaphores provide two operations: wait (P) and signal (V). The wait operation decrements the value of the semaphore, and the signal operation increments the value of the semaphore. When the value of the semaphore is zero, any process that performs a wait operation will be blocked until another process performs a signal operation.
    - Binary semaphores (0 and 1), Counting semaphores (more than 0 and 1)
- Condition variables – A conditional variable is like a queue. A thread stops its execution and enters the queue if the specified condition is not met. Once another thread makes that condition true, it sends a signal to the leading thread in the queue to continue its execution.
    - We use the *wait* instruction in a thread if we want to halt the execution of that thread till a certain condition is met.
    - We use the *signal* instruction if we want to continue executing the leading thread in the waiting queue.
    - We use the *broadcast* signal to run all waiting threads.
- Barriers – A barrier for a group of threads or processes in the source code means any thread/process must stop at this point and cannot proceed until all other threads/processes reach this barrier.

- Message passing – Provides a mechanism to allow processes to communicate and to synchronize their action without sharing the same address space.
    - Direct or indirect communication – There must exist a **link** between processes to make communication possible.
        - Direct communication – Only one link between two processes
            - Symmetric variant:
                - Processes must specify who they are sending and receiving messages from.
                  send(P, message): a process sends a **message** to process **P**.
                  receive(Q, message): a process receives a **message** from process **Q**.
            - Asymmetric variant:
                - Sender must specify who they are sending to, receiver can receive from any process to which they have an communication link with
                  send(P, message): a process sends a **message** to process **P**.
                  receive(*id*, message): a process receives a **message** from any process to which they have an **link** with.
        - Indirect communication – A link can exist between more than two processes. There may be more than one link between a communicating process, but each link correspond to exactly one mailbox
            - Messages can be sent and receives from mailboxes. Each mailbox must have a unique identification. Processes can communicate only if the processes have a shared mailbox.
              send(A, message): a process sends a **message** to mailbox **A**.
              receive(A, message): a process receives a **message** from mailbox **A**.

## Abbreviations

- ROM – Read Only Memory
- BIOS – Basic Input/Output System
- POST – Power-on Self-test
- CMOS – Memory on a motherboard that stores the BIOS settings. A small battery, called a *CMOS* battery, keeps it powered.
- CHS 0,0,1 – Cylinder 0, Head 0, Sector 1
- MSB – Most significant bit