Process Management: Process States

Long Term Scheduling: when to admit a new task to the system.

Short Term Scheduling: what task gets to run next (move from Ready to Running).

Non-preemptive scheduling: task voluntarily gives up the CPU, i.e. when it needs to do I/O.
Preemptive scheduling: CPU can preempt a task when it runs longer than its time-slice.
Process Management: Process States

Short Term Scheduling methods:

**FCFS** (first come first served)
- non-pre-emptive
- response time can vary based on order tasks were started

**SJF** (shortest job first)
- provably optimal
- must estimate how long a task will run
- pre-emptive version called "shortest remaining time first"

**Round Robin**
- run task till time-slice, then move to next task

**Priority**
- may starve low priority tasks
Process Management: Process States

New → Ready (admit (Long Term)) → Running (run (Short Term)) → Waiting for I/O → Halted (done)

- time slice interrupt (pre-emptive)
- I/O device interrupt
- recoverable error (fault, exception)
- I/O complete (interrupt from device)
- I/O request (system call)
- unrecoverable error

Operating System Elements

- System Call from Process
  - System Call Handlers
  - Interrupt Handlers
  - Long Term Queue
  - Short Term Queue
  - I/O Queues

- Short Term Scheduler

Pass Control to selected Process
Process Management: Process States

Queue Model for an Operating System
**Multiprogramming / MultiTasking**: allowing multiple tasks/processes/programs to run concurrently (on one or more processors).

**Multithreading**: allowing a process to be divided into threads and independently scheduling threads (on one or more processors). A program can launch several independent threads. The scheduler must determine how and when they run.

Task context switch:
- All information about a running task needs to be saved and the information about the new task needs to be loaded.
  - PC, Registers, Page table
- Task switches are expensive, so they need to be managed efficiently.

Thread context switch:
- All the threads in a program share the same Virtual Memory, so a thread switch doesn't need to save and restore the Page Table so a Thread context switch can be more efficient.

**Multiprocessing**: a system using multiple CPUs. The Operating System now schedules tasks and threads to run on multiple CPUs. This needs to be done correctly in order to be efficient.

  - **Tightly-coupled**: CPUs share memory and I/O. Typically 16 (256) or fewer processors.
  - **Loosely-coupled**: CPUs have their own private (distributed memory) and are connected using a network, but share a common (distributed) operating system. Can be built from a **cluster** of networked PCs.

**Multi-core**: A single chip with multiple CPUs on the chip.
Operating System Services

**Process management** (managing the CPU)
- creating, scheduling and terminating processes

**Resource Management** (managing memory, virtual memory and I/O devices)
- tracks free page frames and manages page tables
- handles page faults
- includes device drivers so individual applications don’t need their own code to access I/O devices
- system calls used to access I/O devices (open, close, write, read)
- controls when processes can access an I/O device preventing processes from interfering with each other

**File System Management**
- file and directory creation, deletion, access permission, reading, writing and searching.

**Security and Protection**
- protect the system from buggy software and from malicious software
- memory protection (user / kernel)
- file protection
- I/O protection
- CPU protection (timeslice interrupt allows the O/S to prevent a program in an infinite loop from dominating the system)

**User Interface**
- command line vs GUI
Real Time Systems use an RTOS (Real Time Operating System) used for control of physical systems i.e.: manufacturing process control, robotics, avionics. In these systems we have to be able to guarantee that deadlines are met and devices get service within certain time limits.

**Hard Real Time**: all deadlines are met
**Soft Real Time**: the average response time meets a deadline

Embedded System O/S are often real-time need to have a small footprint

VxWorks
**Virtual Machine:**
a software abstraction of a real machine.
Each virtual machine sees an environment consisting of a CPU, registers, I/O devices and virtual memory as though these resources were dedicated to the exclusive use of the virtual machine.

+ Multiple OSs (same or different) can be running simultaneously on the same hardware.
+ Protection through a high degree of isolation
+ May support an ISA that is slightly different from the real machine, or emulate a completely different ISA.
  - A virtual machine is less efficient that a real machine because it doesn’t access the hardware directly (depending on the degree of emulation).
  - Multiple VMs on a single machine can interfere with each other’s performance.

Hopefully, attempts by software (running in each virtual machine) to access real hardware will trap to the VMM. Sometimes, as in x86, the original ISA does not support this, so some of the code may need to be re-written as it is loaded.

**Figure 8.1** Virtual Machine Images Running under a Control Program
Operating systems and VMMs can be divided into logical subsystems, each subsystem is somewhat isolated from the other subsystems and can be configured, managed, started and stopped independently of the others.

i.e., Windows has Win64, Win32, WIN16 and Posix, File manager and security subsystems.
Logical Partitions are more isolated. No resources are shared between partitions providing for a sandbox environment.

**Figure 8.4** Logical Partitions and Their Controlling System:
Resources cannot be shared easily between the partitions
Server Virtualization (Consolidation)
An IBM zSeries (formerly IBM 390) mainframe can support as many as 60 logical partitions. Each partition, running IBM VM, can support thousands of virtual Linux systems. So, a football sized server farm can be replaced by one zSeries machine which is slightly larger than a refrigerator.
Advantages of Server Consolidation

- Managing one large system is easier than managing a multitude of smaller ones.
- A single large system consumes less electricity than a group of smaller systems having equivalent computational power.
- With less electrical power usage, less heat is generated, which economizes on air conditioning.
- Larger systems can provide greater fail-over protection. (Hot spare disks and processors are often included in the systems.)
- A single system is easier to backup and recover.
- Single systems occupy less floor space, reducing real estate costs.
- Software licensing fees may be lower for a single large system than for a large number of small ones.
- Less labor is required for software upgrades.