Basic Set of Material to Cover in Lecture on Chapter One of 9th Edition of Silberschatz.

The OS performs as an intermediary between the user of the computing system and the hardware, functioning as a virtual computer that makes the use of the hardware more convenient and/or efficient.

The first chapter
* Describes how a computing system is organized, and
* What the components are

It also explains about the variety of different kinds of computing systems (environment) and gives some examples.

SECTION 1.1 - What Operating Systems Do
* operate the hardware (minimizing misuse)
* coordinates use of the hardware among all the programs running on the system
* makes the hardware (much) easier to use
* insures that the hardware is utilized efficiently
* allocates resources (such as CPU time and memory) to running programs

Most or all programs running on a computer need to be provided with the services listed above.

It does not make sense to require every application programmer to furnish code that implements these common services.

It's better to have a single program, the OS, that provides these common services to all the other programs.

SECTION 1.2 - Computer System Organization

1.2.1 - Computer System Operation
Components
* CPU(s), primary memory, and device controllers sharing a bus
* CPUs and controllers can execute in parallel
* CPUs and controllers compete to use the primary memory
* The memory controller arbitrates competition for use of memory

Typically, computers boot through automatic execution of a program in ROM that knows how to load the OS from secondary memory and jump into it. Details vary.

The primary purpose of CPUs is to execute user program. Therefore, whenever the operating system is executing on a CPU, its job is to finish whatever it needs to do as soon as possible, and then VACATE the CPU by dispatching a user process.

This means that the OS is NOT EXECUTING much of the time. (When people say that the OS is "always running" - that's not literally true.) However, the hardware and the user processes need to have a way to summon the operating system when they need it to provide a service. The purpose of INTERRUPTS is to give the hardware and the software a way to summon the operating system. The hardware is designed and constructed in a way that allows devices and running programs to send an interrupt to a CPU, which causes the CPU to immediately save some critical information about the current process and then begin executing a service routine associated with the interrupt. This service routine is part of the operating system. This is very important. The operating system is completely dependent on interrupts. There is no other way for the OS to begin execution again after it has given the CPU to a process.

1.2.2 - Storage Structure

1.2.3 - I/O Structure
Typically an operating system contains a module called a device driver for each type of controller in the hardware system. The peculiarities of how to operate the device are sequestered in the device driver code, and the rest of the operating system can use a uniform interface to the device, which is provided by the device driver. Device drivers are complimentary to interrupts: device drivers provide communication from the the OS to the devices, while interrupts provide a means of communication from devices to the operating system.

Sometimes the style of I/O requires one interrupt for each transfer of a byte. Some devices function using direct memory access (DMA) which accomplishes the transfer of large blocks of data between primary memory and a device with only one interrupt.

SECTION 1.3 - Computer System Architecture

1.3.1 - Single-Processor Systems
In such systems there is only one general purpose CPU

1.3.2 - Multiprocessor Systems
These systems have multiple CPUs
* typically they are 'tightly-bound' CPUs that share a common bus and primary memory, as well as power supply and peripherals.
* Most commonly they are symmetric multiprocessing systems, rather than boss/worker designs. In an SMP all processors can and do perform all types of system work, including execution of operating system code.
* Multiprocessors can execute more than one program simultaneously.

* It is a challenge for the OS to keep all the processors busy.

Multicore Systems - are multiprocessor systems in which multiple CPUs share the same chip.

Blade systems are essentially multiple independent computers housed in the same chassis. Each mainboard may contain multiple cores and/or multiple processors.

1.3.3 - Clustered Systems
- variant architecture
- special OS features may be required

SECTION 1.4 - Operating System Structure
+ The ability to multiprogram is important to OS design.
It necessitates sophisticated OS software.
+ Remember that multiprogramming is a concept that is quite
different from multiprocessing.
+ However, the two have something in common - both lead to a
situation in which the computer exhibits concurrency.
+ This is especially true when the multiprogramming style is time-sharing
(aka multitasking) wherein the context switches occur very frequently.
+ Time-sharing operating systems have to be very sophisticated too -
in order to support interaction of each user with his or her jobs.
+ Examples of the demands on the OS are for online file systems,
process scheduling, memory management, swapping,
and process synchronization. These are major themes of chapters
of the textbook.

SECTION 1.5 - Operating System Operation
1.5.1 - Dual Mode and Multimode Operation

Dual mode operation is a method for protecting the computing system, by insuring that user processes are not able to execute dangerous operations.

The basic idea is to have a mode bit in the hardware.
The computer boots with the mode bit cleared (0).
The OS sets the bit before jumping into user code (1)
An interrupt clears the mode bit (0).
In this manner, the mode bit is assured to be 0 when the OS is executing,
and 1 when a user process is executing.
Privileged instructions are implemented in the hardware - instructions
that can be executed only in kernel mode (0). An attempt to execute a

privileged instruction while in user mode just causes a trap to the
operating system.
Examples of Privileged Instructions
* Switch to kernel mode
* I/O control
* timer management
* masking of interrupts

+ A **SYSTEM CALL** is a request made by a process for a service from the OS
+ Usually system calls are implemented by execution of an instruction that causes a trap
+ The service routine of the OS that handles the interrupt figures out
  what service is being requested (often by checking for parameters left
  by the calling process in registers, memory, or on its stack), and
  then calls the appropriate routine to perform the service.

1.5.2 - Timer
Many computer architectures incorporate one or more timers that can be set to interrupt after a specified amount of time. Such a timer can be used by the operating system to prevent a user process from failing to relinquish the CPU. The instructions that manage the timer are privileged, which prevents user processes from disabling the timer.

SECTION 1.6 - Process Management

Basically a process is a program that is executing. Many resources and data are required to maintain a process.
The OS:
* Allocates CPU time to processes
* Creates and deletes processes
* Suspends and resumes processes
* Provides synchronization mechanisms for processes
* Provides mechanisms for processes to communicate with each other

SECTION 1.7 - Memory Management

In a typical multiprogramming computing system, the OS has to
* Allocate and deallocate memory
* Keep track of which parts of memory are free and allocated
* Keep track of which processes are allocated which portions of memory
* Deal with situations in which the supply of free memory is depleted

SECTION 1.8 - Storage Management

* The OS is responsible for implementing the structure of a file system, usually with secondary storage being the main underlying physical structure.

1.8.1 - File-System Management
* The OS implements the abstract concept of a file

* The OS implements file directory structure
* The OS implements file ownership and permissions
The OS is responsible for
+ file creation and deletion
+ directory creation and deletion
+ primitives for file manipulation
+ mapping files onto secondary storage devices
+ backing up files

1.8.2 - Mass-Storage Management
* The OS has to manage
  + Free space on disk
  + Disk storage allocation and deallocation
  + The scheduling of transfers of data between disk and primary memory

Since the speed of secondary storage is often a performance bottleneck,
the efficiency of the implementation of secondary storage may be critical to operating system design.

The OS often has responsibility for tertiary storage - operations such as mounting and unmounting backup tape drives, and performing backup and restore functions.

1.8.3 - Caching
* Know what it is, the basics of how it is implemented
* Know that cache size and replacement policy are important to hit ratio.
* Be aware of the problem of keeping consistency among copies of data kept in different cache levels, or in separate caches that are at the same level.

1.8.4 - I/O Systems
The I/O subsystem provides "information hiding" between the I/O devices and higher levels of the OS. The components of the I/O system are
* memory management such as buffering, caching, and spooling
* A general interface for the OS to use to communicate with device drivers.
* device drivers for specific hardware devices

SECTION 1.9 - Protection and Security

Things that come under this heading
* authorization
* preventing access to some resources
* preventing misuse of some resources to which access IS ALLOWED

* Protection is mainly about ensuring access is authorized (checking ID).

* Security is more about monitoring usage to defend it against misuse

SECTION 1.10 - Kernel Data Structures

1.10.1 Lists, Stacks, and Queues
* Operating Systems are fundamentally programs. They are big sophisticated programs that utilize a wide variety of data structures
+ arrays
+ various kinds of lists, many of which are linked structures
+ stacks
+ queues

1.10.2 Trees
Operating systems utilize a wide variety of tree data structures, like binary trees and trees that have much greater potential "fan out"

1.10.3 Hash Functions and Maps
+ For example the OS might hash user names to lookup passwords when interacting with a user who is trying to log in to the system

1.10.4 Bit Maps
A string of n bits can be used to indicate the free/allocated status of n items. A bit value of 0 commonly means the resource is free, and 1 means it is allocated. Bit maps are very compact, so they are used for things like allocation of disk blocks.

SECTION 1.11 - COMPUTING ENVIRONMENTS

1.11.1 - Traditional Computing
This may consist of PCs connected to a network, servers, and remote access with laptop computers. Some systems are still using terminals attached to main frames

Now web portals are more common, and "thin clients" - machines that have just enough power to utilize web-based services. More connections are using wireless and cellular networks.

People now have faster internet connections at home, with firewalls, and often home networks.

Batch and time-sharing systems are increasingly rare. Time sharing technology is still prevalent, but mostly used to support the multiple processes all running for the benefit of a single user.

1.11.2 - Mobile Computing
Use of mobile computing has exploded - e-mail, messaging, browsing, navigation and other location-based services, augmented reality that sometimes involves overlaying real scenes with information. Hardware for mobile systems is currently limited in speed and capacity.

1.11.3 - Distributed Systems
+ Separate machines - heterogeneous
+ Network operating system generally means one that runs separately on each host, but which is "network aware"
+ A truly distributed operating system makes the

network of computers take on the nature of a single computer with a single operating system.

1.11.4 - Client-Server Computing
+ There are compute-server systems
+ There are file-server systems

1.11.5 - Peer-to-peer Computing
+ The ability of a host to act as a server or as a client is exploited in peer-to-peer systems.
+ There can be a multiplicity of servers, which can reduce server bottleneck problems.
+ However clients in peer-to-peer networks often utilize a centralized lookup service to locate a server.
+ An alternative is for clients to broadcast to all hosts in the peer-to-peer network to ask for a service.
+ Skype is also peer-to-peer networking.

1.11.6 - Virtualization
* This is technology that allows an operating system to run as applications within another operating system.
* Virtualization is independent of emulation, since both the virtualized and native OSs may be designed to execute on the same CPU - whereas emulation denotes interpretation of the instructions of one CPU as a set of instructions on a different CPU.
 + Virtualization allows users to install multiple OSs on a single machine, and thereby provide a convenient means of running more applications - since some applications are not available on some Operating System platforms.
 + People who are trying to port software can test versions for different operating systems on just one machine that has all the operating systems installed.

1.11.7 - Cloud Computing
+ Cloud computing is compute-serving, file/storage serving, and application-serving.
+ Users pay per month for what they use.
+ Clouds can be public, private, or a hybrid.
+ There is SaS -software as a service, PaaS - platform as a service (e.g. a database server), IaaS - infrastructure as a service (e.g. storage for backups)

1.11.8 - Real Time Embedded Systems
* These ubiquitous systems are the computers in devices like cars, microwave ovens, and so on. They mainly monitor and manage devices, seldom have extensive user interfaces, and are not used as general purpose computers (although the hardware may be identical to the hardware used for some general purpose computers).

* Some embedded systems use application specific integrated circuits (ASICs) that can provide desired functionality without having an operating system.
* The OS used in an embedded system is often REAL TIME.

SECTION 1.12 - Open-Source Operating Systems

* These are operating systems for which one can get the source code, not just the compiled version of the program.
* Often open source code is easier to maintain, improve, experiment on, learn from, and so forth.

1.12.1 - History
Early in the history of CS, programmers shared source code commonly. Later it became common to withhold it, and sell only the object code. Some of the reasons were to protect profits, and keep secrets needed for the efficacy of media copy protection.
* Richard Stallman's Free Software Foundation encourages the sharing of source code.

1.12.2 - Linux
+ Linus Torvalds put GNU Unix-compatible tools together with a Minix-inspired kernel and then people worldwide refined it into the Linux operating system.
+ There are numerous distributions of Linux

1.12.3 - BSD Unix
+ BSD Unix has an interesting history.
+ Apple's Darwin kernel is based on BSD Unix

1.12.4 - Solaris
+ Sun Microsystem Unix
+ First based on BSD, then AT&T Unix
+ Sun sold out to Oracle
+ Development of Solaris continues by third party Illumos

1.12.5 - Open Source Systems as Learning Tools

SECTION 1.13 - Summary