estratto da:

W. Stallings, Operating Systems Internal and Design Principles, Pearson International Edition

Operating System Overview (chapter 2)

OPERATING SYSTEM OBJECTIVE AND FUNCTIONS

Operating System

- a program that controls the execution of application programs
- an interface between applications and the computer hardware

OS objectives

- convenience: make the computer more convenient to use
- efficiency: allows computer system resources to be used in an efficient manner
- ability to evolve: permits the effective development, testing, and introduction of new system functions without interfering with service

Reasons why an OS evolves

- hardware upgrade plus new types of hardware
- new services
- fixes

Service provided by the OS

- program execution
- access I/O devices
- controlled access to files
- system access (controls access to the system and to specific system resources)
- memory management
- error detection and response (hardware errors, such as memory error, device failure, malfunction, and software errors, such as division by zero, access forbidden memory locations)
- accounting (collect usage statistics, monitor performance)
- program development (utility programs - editors and debuggers - to assist the programmer in creating programs)
- network support

EVOLUTION OF OPERATING SYSTEM

Serial Processing (from the late 1940s to the mid-1950s)

the programmer interacted directly with the computer hardware; there was no OS; the computers were run from a console with display lights, toggle switches, input device – e.g. a card reader - and printer; the users had access to the computer in series.

Simple Batch Systems (from the mid-1950s)

a software, known as the **monitor**, controls the sequence of events; the user does not have direct access to processor, but submits the job on cards or tape to a computer operator who batches the jobs together and places the entire batch on an input device, for use by the monitor; each job branches back to the monitor when it completes. With each job, instructions are included in a primitive form of *job control language*, used to provide instructions to the monitor (what compiler to use, what data to use, etc.).

New hardware features:

- **timer**: a single job must not monopolize the system;
- **interrupts:** OS can relinquish control to and regain control from user programs;
- **privileged instructions**: certain machine level instructions can be executed only by the monitor;
- **memory protection**: a user program must not alter memory area containing the monitor.

New concept of modes of operations:

- user program executes in **user mode:** certain areas of memory are protected, certain instructions may not be executed;
- monitor executes in **system mode**: privileged instructions may be executed, protected areas of memory may be accessed.

With *uniprogramming,* the processor is often idle: the problem is that I/O devices are slow compared to the processor.

Multiprogrammed Batch Systems

the central theme of modern OS is **multitasking**: if there is enough memory to hold the OS and two, three or more user programs, when one job needs to wait for I/O, the processor can switch to the other job.

New hardware features:

- **I/O interrupts**
- **DMA** (direct memory access)

New concepts:

- **memory management**
- **algorithm for scheduling**

Time-Sharing Systems

multiprogramming is used to handle multiple batch jobs and multiple interactive jobs. The processor time is shared among multiple users, which simultaneously access the system through terminals.