Introdução aos Sistemas Operativos

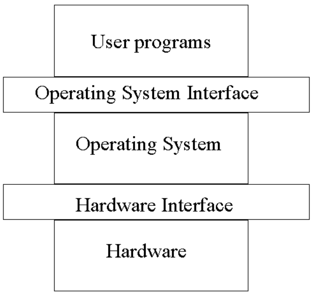
(corresponde em parte ao Chapter2 do W.Stallings)

SISTEMAS OPERATIVOS, 1º Semestre, 2004-2005

What is an Operating System?

- **Possible definitions:**
  - **the code that {Microsoft, Sun, Linus} provides**
  - **the code that you didn’t write**
  - **the code that runs in privileged mode**
  - **the code that makes things work**
  - **the code that makes things crash**
  - **etc.**

What is an Operating System?



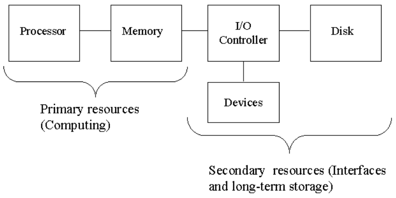
Levels in a computer System

What is an Operating System?

- A program that acts as an intermediary between a user of a computer and the underlying hardware.

- Exploits the hardware resources of one or more processors (CPU, Memory, Disk, etc).
- Allow the user to conveniently access data and programs.
- Provides a set of services to system users.
- Manages secondary memory and I/O devices.

Operating System as a Resource Manager


Processor
Memory
I/O Controller
Disk
Devices
Primary resources (Computing)
Secondary resources (Interfaces and long-term storage)

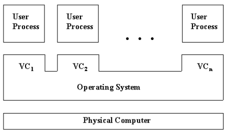
Hardware resources in a computer system

Services Provided by an Operating System

- Program development
  - – Editors and debuggers
- Program execution.
- Access to I/O devices.
- Controlled access to files.
- System access.
- Error detection and response.
- Accounting and Monitoring.

OS creates virtual computers

- multiple *processes* (simulated processors)
- multiple *address spaces* (memory for a process to run in)
- implements a *file system*



Multiple virtual computers in an operating system

Kernel

- Portion of operating system that is in main memory.
- Contains most-frequently used functions.

Evolution of Operating Systems

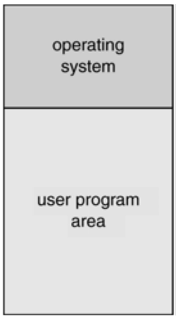
OS Strategies

- Serial Processing
- Batch processing
- Multiprogramming and Timesharing
- Multitasking and Multi-threading
- Personal computer & workstations
- Parallel Systems
- Real-time systems
- Distributed systems

Batch Processing Systems

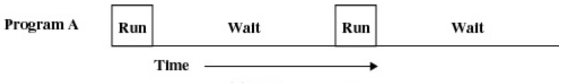
- *Job* (file of OS commands) prepared offline.
- Batch of jobs given to OS at one time.
- OS processes jobs one-after-the-other.
- No human-computer interaction.
- Try to optimize resource utilization.

Memory Layout for a Simple Batch System



Uni-programming

- Processor must wait for I/O instruction to complete before proceeding.

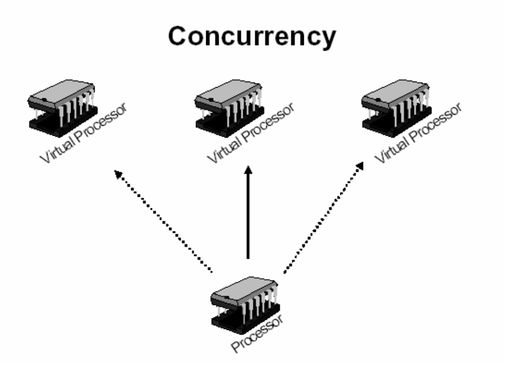


(a) Uniprogramming

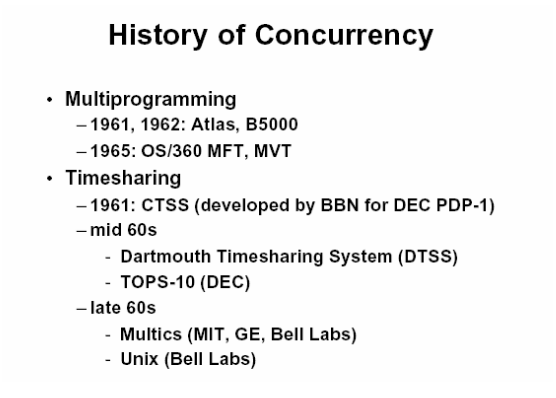
CPU Idleness...

• Read one record from a file:	0.0015 seconds
• Execute 100 instructions:	0.0001 seconds
• Write one record to file:	0.0015 seconds
Total:	0.0031 seconds

Percent CPU Utilization= 0.0001/0.0031 = 3.2%



Concurrency



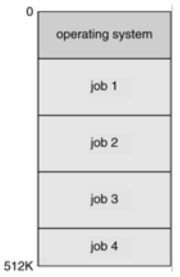
History of Concurrency

- Multiprogramming
  - 1961, 1962: Atlas, B5000
  - 1965: OS/360 MFT, MVT
- Timesharing
  - 1961: CTSS (developed by BBN for DEC PDP-1)
  - mid 60s
    - Dartmouth Timesharing System (DTSS)
    - TOPS-10 (DEC)
  - late 60s
    - Multics (MIT, GE, Bell Labs)
    - Unix (Bell Labs)

Multiprogramming

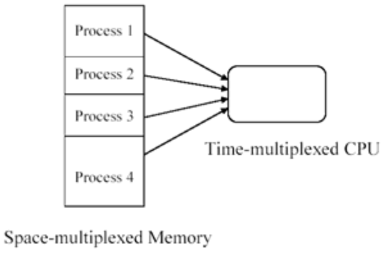
- Technique for *sharing* the CPU among *runnable* processes.
  - Process may be *blocked* on I/O.
  - Process may be *blocked* waiting for other resource.
- While one process is blocked, another should be able to run.
- Needs memory management and scheduling.
- Reduced time to run all processes.

Multiprogrammed Batch Systems



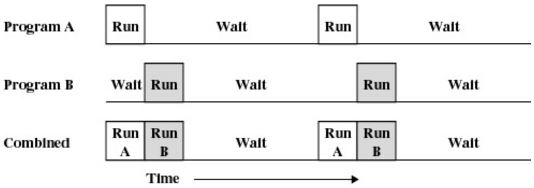
Several jobs are kept in main memory at the same time, and the CPU is multiplexed among them.

Multiprogramming



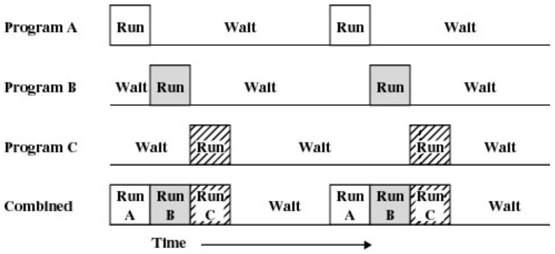
Multiprogramming

- When one job needs to wait for I/O, the processor can switch to the other job



(b) Multiprogramming with two programs

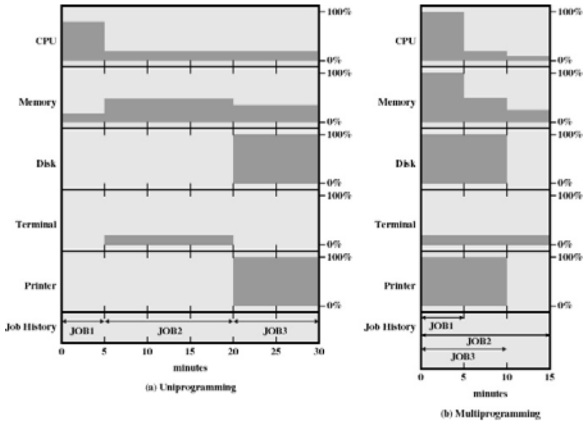
Multiprogramming



(c) Multiprogramming with three programs

Example

	JOB1	JOB2	JOB3
Type of job	Heavy compute	Heavy I/O	Heavy I/O
Duration	5 min.	15 min.	10 min.
Memory	50K	100 K	80 K
Need disk?	No	No	Yes
Need terminal	No	Yes	No
Need printer?	No	No	Yes



(a) Uniprogramming

(b) Multiprogramming

Figure 2.6 Utilization Histograms

Effects of Multiprogramming

	Uniprogramming	Multiprogramming
Processor use	22%	43%
Memory use	30%	67%
Disk use	33%	67%
Printer use	33%	67%
Elapsed time	30 min.	15 min.
Throughput rate	6 jobs/hr	12 jobs/hr
Mean response time	18 min.	10 min.

Time-Sharing

- The CPU is multiplexed among several jobs that are kept in memory and on disk.
- Using multiprogramming to handle multiple interactive jobs.
- Processor's time is shared among multiple users.
- Multiple users simultaneously access the system through terminals.
- Require a scheduling algorithm and memory Management.

Memory Mangt: Example

- Memory Requirements:
  - Job1: 15,000
  - Job2: 20,000
  - Job3: 5,000
  - Job4: 10,000

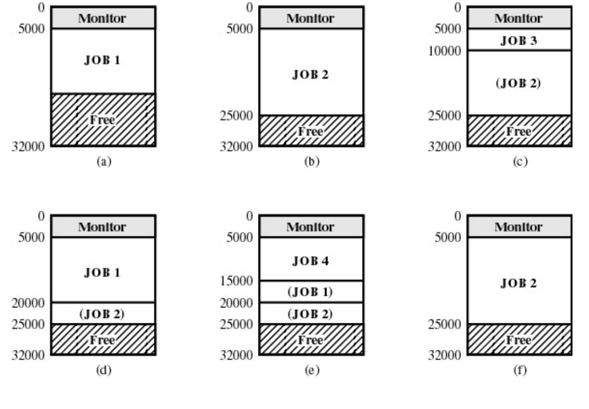


Figure 2.7 CTSS Operation

Multiprogramming vs Time-Sharing

- ***Multiprogramming*** refers to the notion of having multiple programs active at the same time so that when the current program can no longer run (for example because it's blocked waiting for I/O), another is available to run.
- ***Timesharing*** is an extension of multiprogramming in which the execution of the active programs is **time-sliced**: each program runs for a short period of time, then another is run.

Multiprogramming vs Time Sharing

	Batch Multiprogramming	Time Sharing
Principal objective	Maximize processor use	Minimize response time
Source of directives to operating system	Job control language commands provided with the job	Commands entered at the terminal

Multitasking

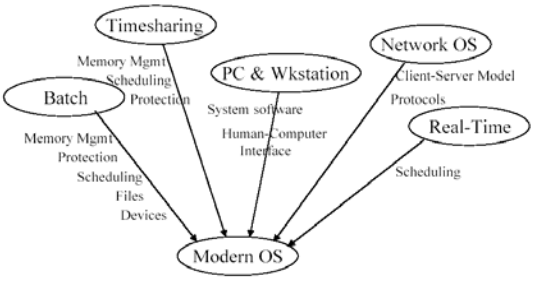
• Optimize the use of computers
— programs typically compute, perform I/O, compute, perform I/O ...
— when one program is performing I/O, another should be computing

• Optimize the use of people
— want several applications active at once
– interactive applications
· editing
· drawing
– background applications
· mail monitor
· printing
· file transfer

Preemptive Multitasking

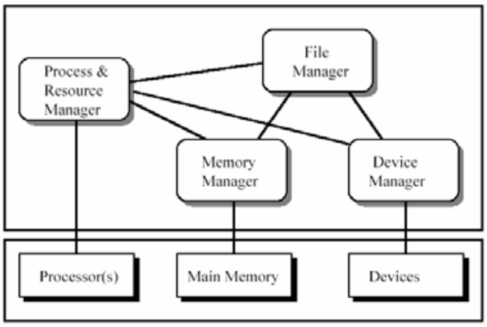
- ***Preemptive Multitasking**:* It's a multiprogramming technique in which one program with higher priority can preempt the execution of another program with lower priority for urgent execution.
- This notion was first implemented in the 1960s.
- (Note that Microsoft "innovated" this idea in 1993 with their introduction of Windows NT.)

Evolution of Modern Operating Systems

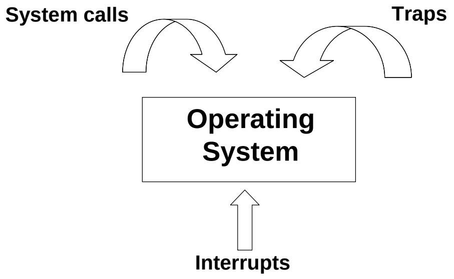


Operating System Organization

Basic OS Organization



System Calls, Traps and Interrupts


System calls

Traps

Operating System

Interrupts

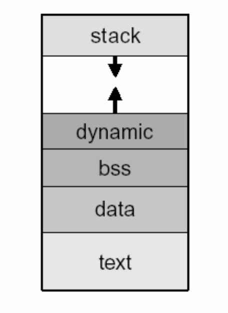
Processes

- A program in execution.
- An instance of a program running on a computer.
- A unit of activity characterized by a single sequential thread of execution, a current state, and an associated set of system resources.

Process

- Consists of three components:
  - An executable program
  - Associated data needed by the program
  - Execution context of the program
    - All information the operating system needs to manage the process

Unix Address Space



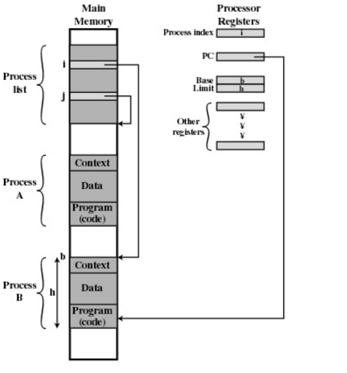
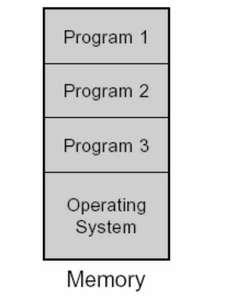
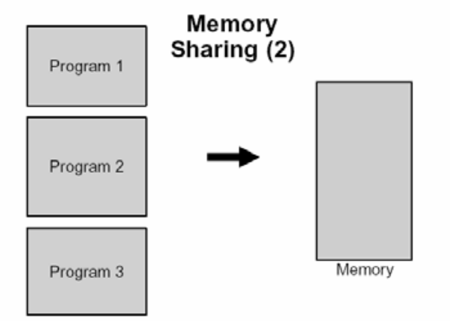


Figure 2.8 Typical Process Implementation

Memory Sharing



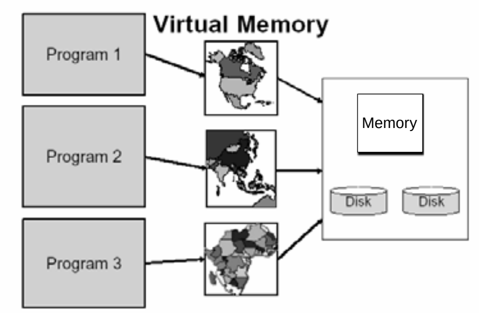
Memory Sharing



Memory Management

- Process isolation.
- Automatic allocation and management.
- Support for modular programming.
- Protection and access control.
- Long-term storage.

Virtual Memory

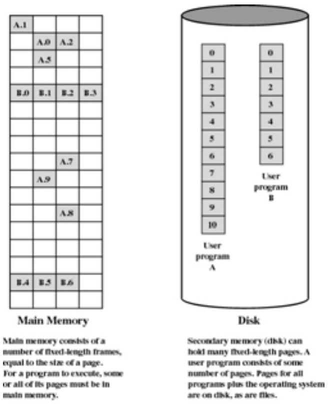


Virtual Memory

- Allows programmers to address memory from a logical point of view.
- While one process is written out to secondary store and the next process is read in, there should be no waiting time.

Paging

- Allows process to be comprised of a number of fixed-size blocks, called pages.
- Virtual address is a page number and an offset within the page.
- Each page may be located any where in main memory.
- Real address or physical address in main memory.



Main memory consists of a number of fixed-length frames, equal to the size of a page. For a program to execute, some or all of its pages must be in main memory.

Secondary memory (disk) can hold many fixed-length pages. A user program consists of some number of pages. Pages for all programs plus the operating system are on disk, as are files.

Figure 2.9 Virtual Memory Concepts

Virtual Memory Addressing

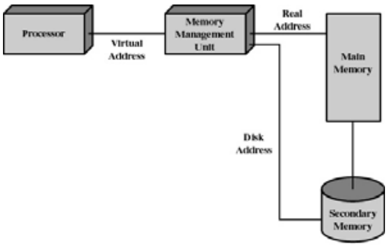


Figure 2.10 Virtual Memory Addressing

Major Elements of an Operating System

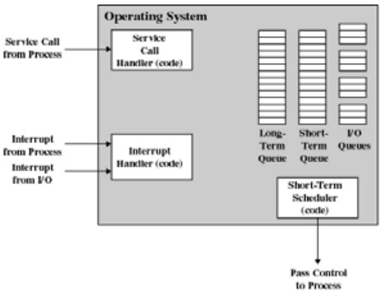


Figure 2.11 Key Elements of an Operating System for Multiprogramming

Characteristics of Modern Operating Systems

- **Microkernel** architecture
  - – assigns only a few essential functions to the kernel
    - address space
    - interprocess communication (IPC)
    - basic scheduling

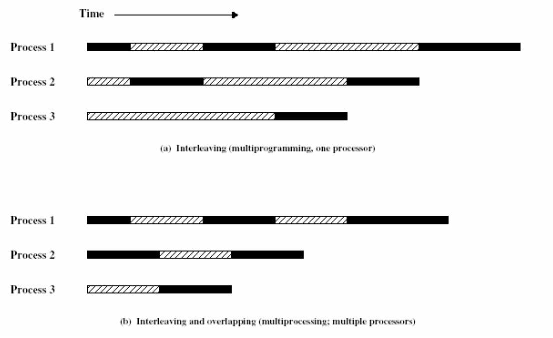
Characteristics of Modern Operating Systems

- **Multithreading**
  - – process is divided into threads that can run simultaneously
- **Thread**
  - – dispatchable unit of work
  - – executes sequentially and is interruptable
- **Process** is a collection of one or more threads.

Characteristics of Modern Operating Systems

- **Symmetric multiprocessing**
  - – there are multiple processors
  - – these processors share same main memory and I/O facilities
  - – All processors can perform the same functions

Interleaving and Overlapping



(a) Interleaving (multiprogramming, one processor)

(b) Interleaving and overlapping (multiprocessing; multiple processors)