Operating Systems

Using Tanenbaum's
Modern Operating Systems (3rd edition)

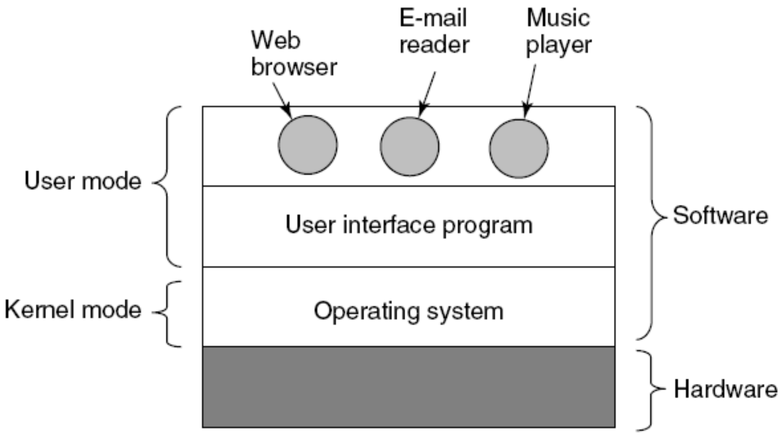
© 2008 Prentice-Hall, Inc. All rights reserved. 0-13-6006639

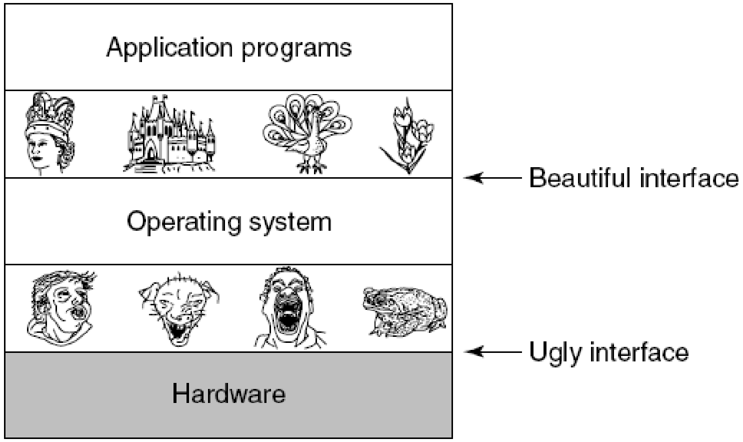
What is Operating System?

What is Operating System ?

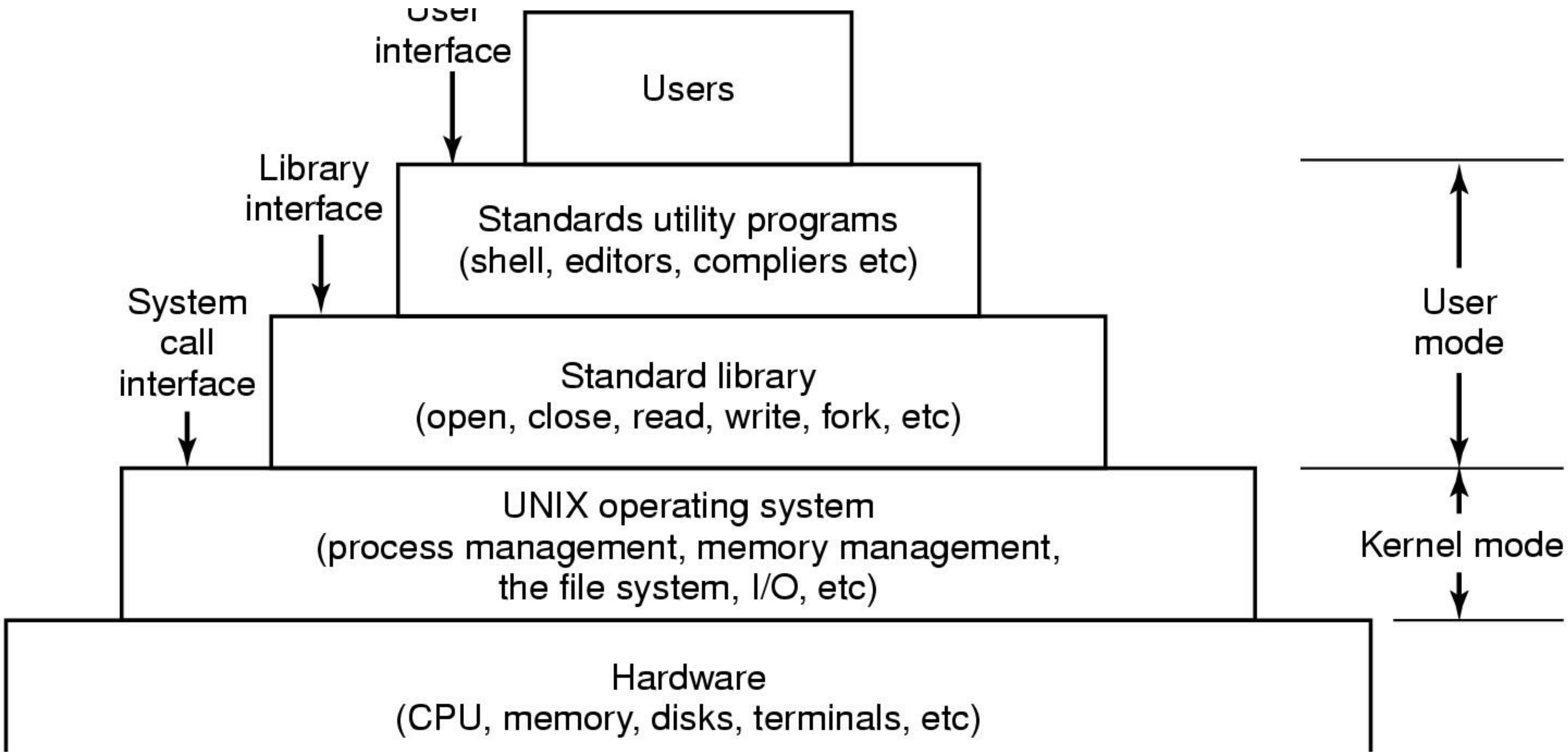
- OS as an **Extended machine** (“Top-down” view) – hides the hardware and presents a nice, clean, elegant and consistent interface (abstraction):
  - Define and implement abstractions.
  - Use these abstractions to solve problems.
- OS as a **Resource manager** (“Bottom-up” view) –
  - Manage all the components of the system: Processors, Memories, I/O devices, etc...
  - Share resource time & space

Operating System (I)





Operating System (II)



User interface

Users

Library interface

Standards utility programs
(shell, editors, compliers etc)

System call interface

Standard library
(open, close, read, write, fork, etc)

UNIX operating system
(process management, memory management,
the file system, I/O, etc)

Hardware
(CPU, memory, disks, terminals, etc)

User mode

Kernel mode

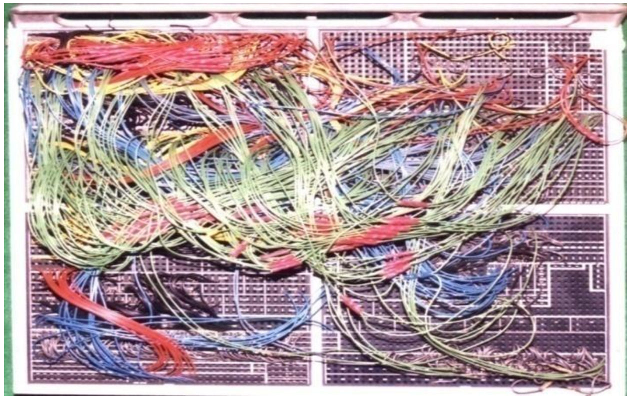
Standard API

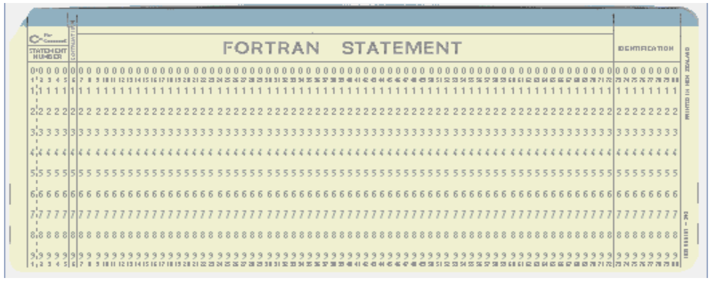
- API – Application Programmer’s Interface
- Standard API enables a program or a project to be written to any HW.
- The OS supplies standard API across multiple HW platforms (i.e. system calls, system utilities, GUI).

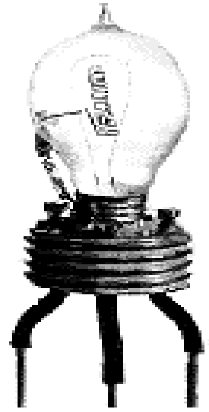
OS History Overview

1st generation (1945-1955)

- During World War II, the first vacuum tube computers were built in American and European universities.
- ***Von Neumann*** (1940) created basic modern computer architecture (HW).
- Programming was first hard wiring of plug boards, and later (1950s) used (Hollerith) punch cards.

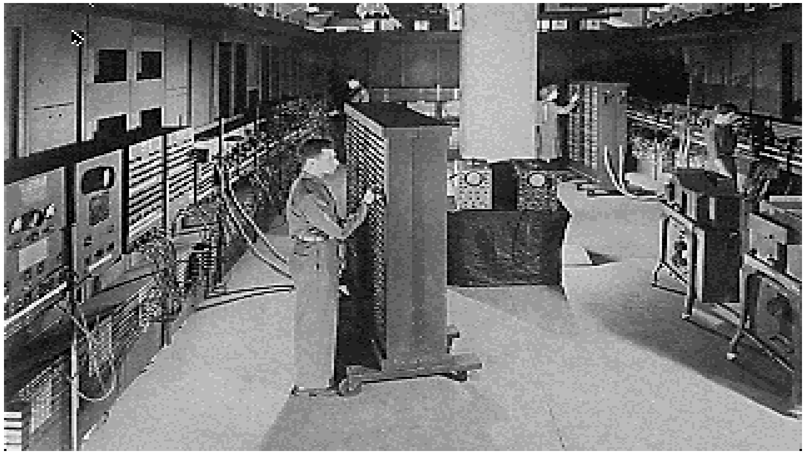




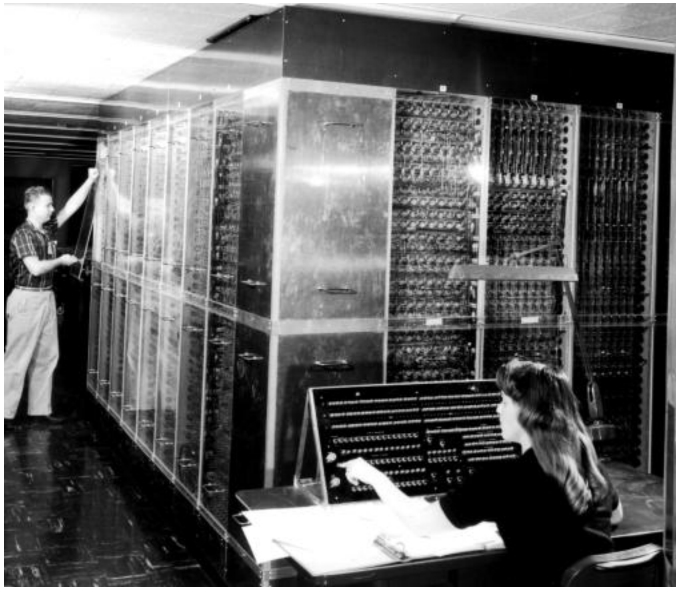


vacuum tube from the early 1900's

First generation



Electronic **N**umerical **I**ntegrator **A**nd **C**omputer (ENIAC)



2nd generation (1955-1965)

- Computers became reliable enough to be manufactured and sold.
  - New jobs created: designers, operators, programmers, maintainers...
  - The new multi-million dollar machines are called **mainframes**
  - A programmer would punch the code (ASM/Fortran) on cards, creating **a job**, hand it to the operator, who was feeding it to machine.

- **Batch-system –** running a collection of jobs.
  - Bugs where a waste of everyone's time.
  - Actual computing took only small percent of the time
  - Used mostly for scientific & engineering calculations.

An early Batch system

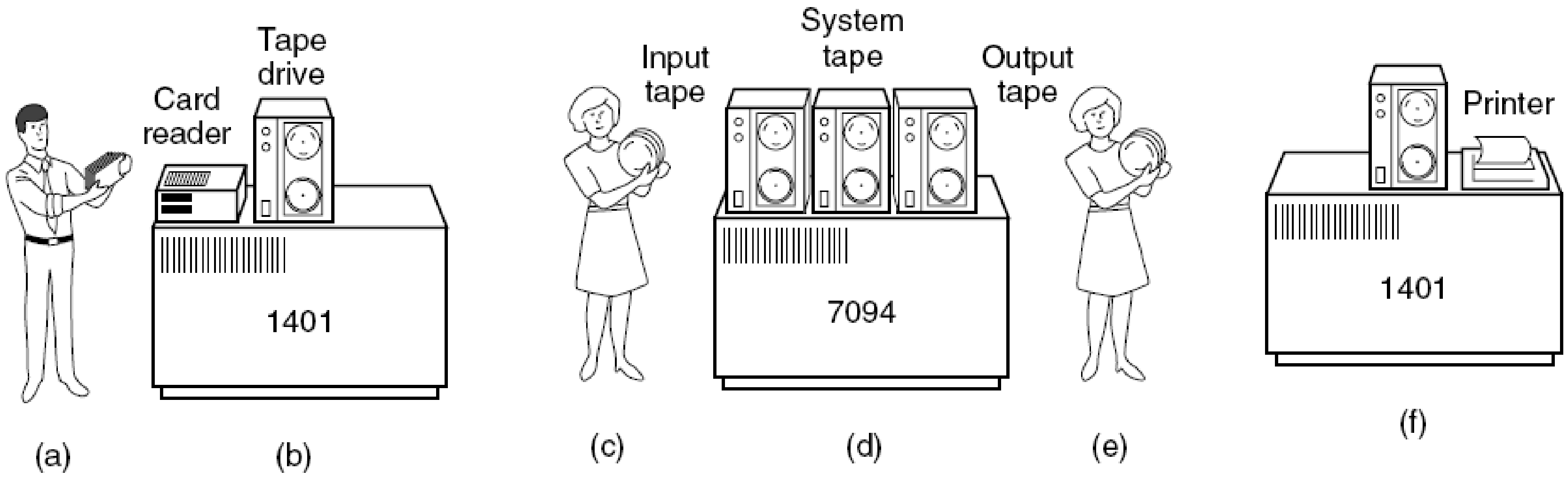


Figure 1-3. An early batch system.

(a) Programmers bring cards to 1401.
(b) 1401 reads batch of jobs onto tape
(c) Operator carries input tape to 7094
(d) 7094 does computing
(e) Operator carries output tape to 1401
(f) 1401 prints output.

A typical Batch job

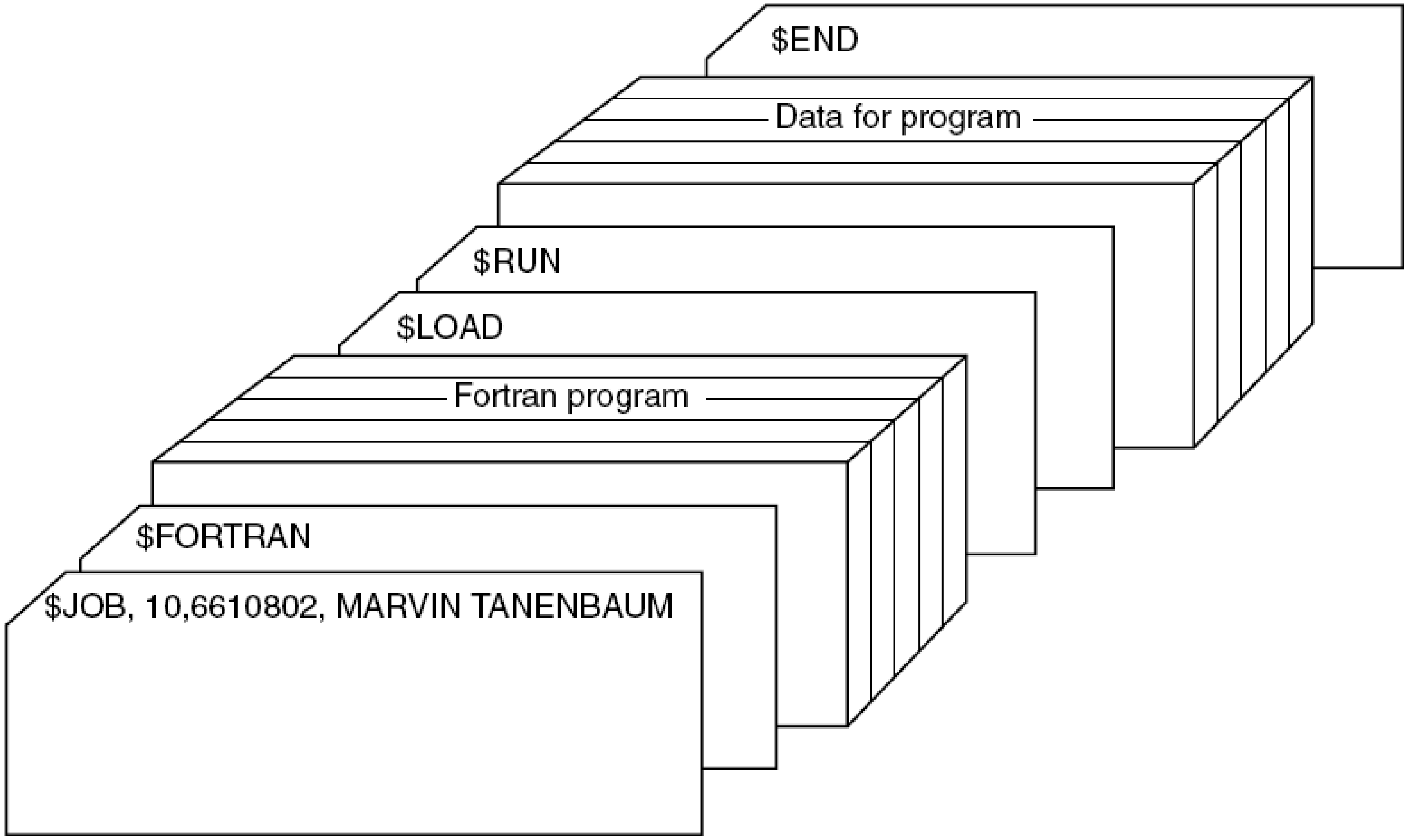


Figure 1-4. Structure of a typical batch job.

3rd generation (1965-1980)

- IBM System/360 (which used ICs):
  - **Spooling**
  - **Multiprogramming**
  - **Timesharing**
- Growth of minicomputers:
  - PDP-1 ('61) – 4K memory, 18 bits, $120K per machine.
  - PDP-11(’70) – First mini-computer time-sharing and real-time systems, running different OS’s
  - VAX/VMS (’78) – First virtual memory system
- Birth of Unix (mid-70’s) in AT&T Bell Labs

Spooling

(Simultaneous Peripheral Operation On-Line)

The Problem:

- The fast CPU was blocked most of the time on a slow printer, and on even slower human loading cards.

The Solution:

- The cards could be cashed (from card to magnetic tape or disk) using a device that works in parallel with the CPU.
- Today:
  - Buffer I/O.
  - CPU writes to a fast memory buffer, then the DMA performs transfer of the data to/from a slow device.

Multi-programming

- Obtaining a resource may take much more than the work.
- **CPU utilization** could be improved, if the CPU could run another task when one is blocked.
- Memory could be segmented to fit more than one task at a time.
- Special HW kept one program from interfering with other.

Multiprogramming Issues

- Resource and memory sharing:
  - Protect (HW guaranteed) one task from another task overwriting its memory or compromising its security.
  - Use of resources that need exclusive access (i.e. printer).

- Quality of service:
  - Serve many tasks with various needs and priorities.
  - Deadlock and starvation.

- Simulate multi programming using a single processor.
  - Time-slicing considerations.

Time-Sharing

Why?

- Interactive systems need rapid response to user command.

How?

- By letting task run only for a **quantum** of time and not until blocking/termination.

Birth of UNIX

- The **MULTICS** system by MIT lead the way to development of the PDP-11 machine and UNIX (70’s):
  - **System-V** from AT&T.
  - **BSD UNIX** from Berkeley.
- IEEE developed a standard known as POSIX, defining a minimal set of system calls, that UNIX should support
- **MINIX** (1987 A.S.Tanenbaum) was the base to Linux.

4th generation (1980-present)

- Intel come out with the first general purpose micro-CPU – 8080 (‘74).
  - Digital Research wrote an OS called **CP/M** (‘77) running on 8080.
- IBM designed the IBM PC and found Bill Gates as the only one willing to write an operating system for it.
  - Bill Gates founded a small garage company to write the **MS-DOS** operating system for the new computer, today known as Microsoft (‘80).
- **IBM PC/AT** – x86 (‘83).

4th generation (II)

<u>User friendly GUI</u>

  – Apple Macintosh.
  – Windows (85’-95’) – GUI on top of MS-DOS.

- Windows 95, 98 : Stand-alone OS, 16-bit based.
- Windows NT : rewrite from scratch, 32-bit based, many ideas from other OS (VAX/VMS).
- Windows 2000: WIN-NT Ver. 5.0 (‘99).
- Windows XP: WIN 2000 upgrade (’01) – the first really stable version of Windows
- Windows Vista, 7: New GUI, many user programs (‘07)
- Windows 8 : going for the touch interface
- Windows 10: unify PC & Tablet

Types of OS

Typical Environments

- Batch systems:
  - A task is running without any interactive user action.
  - What is optimized here ?
    - The computer time => maximum throughput.
- Interactive systems:
  - What is optimized here ?
    - The user interaction=> minimum response time.
- Real-Time systems:
  - Meeting the deadlines.

Basic Terms

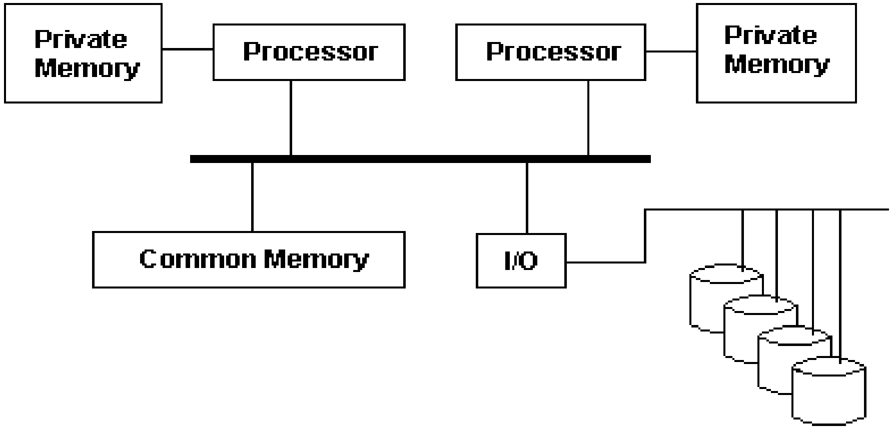
Inside OS

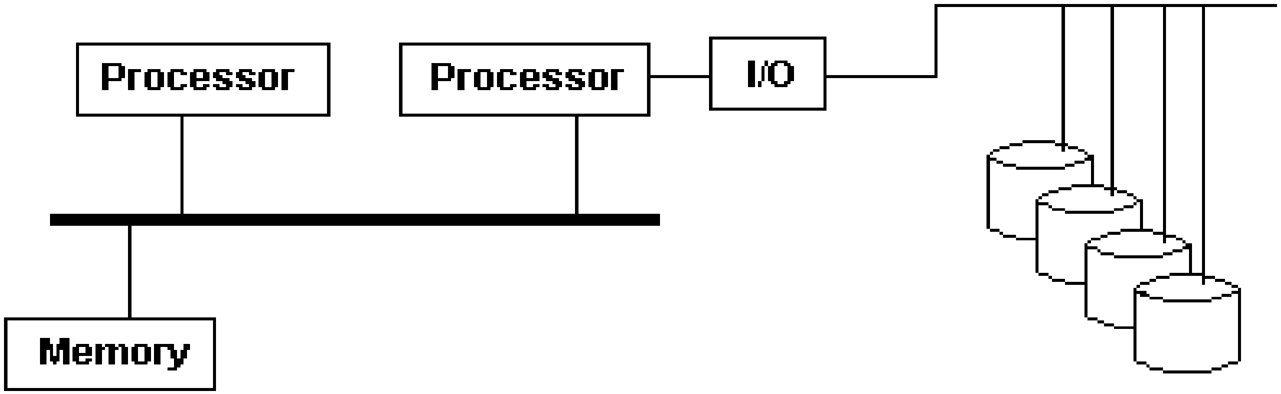
- Process:
  - A program in execution.
  - Has its own address space: code + data.
  - PCB: PC, SP, PSW, etc….
- Kernel:
  - Main part of the OS.
  - Management: Interrupt, Scheduling, Memory.
- Shell:
  - The UNIX command interpreter - OS interface.
- System calls:
  - Process request for OS service: open(),read()…

Types of OS

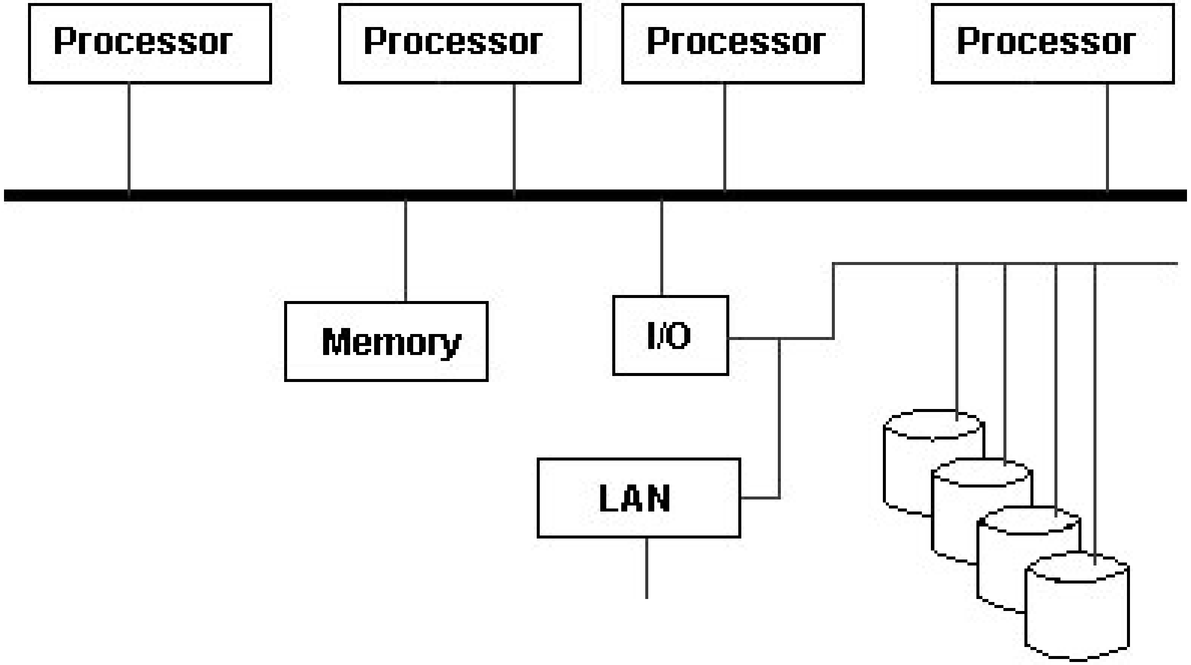
- Multi-Processor (multi-core) :
  - Tightly-coupled – same clock, shared memory.
  - SMP – OS may run on each core.
  - ASMP – OS may run on the master, other cores are slaves.
- Personal Computers OS:
  - Single user services: Windows, Linux, Macintosh.
- Server OS
  - Without graphical user interface
- Real-Time OS:
  - Hard R/T – Military applications.
  - Soft R/T – Digital audio, MM: RMX.
- Embedded OS:
  - Optimized for small HW and needs: VxWorks, Windows Embedded,…

ASMP – Asymmetric Multi-Processing





SMP – Symmetric Multi-Processing



Homework + Interview Questions

1. What are the roles & responsibilities of the Operating System?

2. Explain the following terms. What kind of problems each is trying to solve:
   - Spooling
   - Time Sharing
   - Multi-Programming

3. Explain the 3 types of OS:
   - Batch
   - Interactive
   - Real-Time