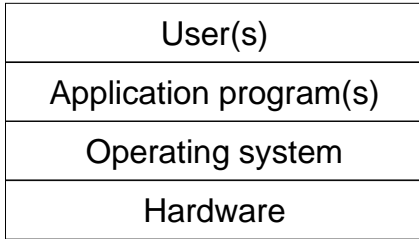


What is an OS?

Increasing levels of abstraction




---

---

---

---

---

---

---

---

OS Function Summary

- **Coordinator**
  - Make things work together
- **Illusion generator**
  - Present cleaner, higher-level interface
- **Standard library**
  - Provide commonly need facilities

M. Rosenblum  
Stanford Univ.

---

---

---

---

---

---

---

---

Coordination

- **Concurrency**
  - One/more users; multiple tasks
  - I/O concurrent with computation
- **Memory management**
- **File management**
- **Network access**

---

---

---

---

---

---

---

---

Illusion Generation

- **Cleaner abstraction**
  - High-level machine
    - Architectural details hidden
- **Multiprogramming**
  - One machine looks like more than one
- **Multiprocessing**
  - Multiple machines look like one

---

---

---

---

---

---

---

---

---

---

Standard Libraries

- **Run-time libraries**
  - Language support
  - User interface (GUI, etc.)
- **Utility programs**
  - File/directory management
  - Scripting tools, etc.

---

---

---

---

---

---

---

---

---

---

Early Computers (1) 1950's ?

- **Single user, single program**
- **Program loaded into memory**
  - Punched cards, paper tape, etc.
    - How? What controls loading process?
- **Program execution**
  - Start at address zero?

---

---

---

---

---

---

---

---

---

---

Early Computers (2)

• Common code segments

- Arithmetic (“run-time library”)
- Input/output (“device drivers”)



• Loaded along with program

- Fixed addresses or “linking loader”
- Resident part of system?

---

---

---

---

---

---

---

---

Batch Systems (1) 1960's

• Computers expensive

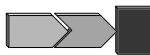
- Slow job set-up time was costly

• Resident monitor

- Permanently (?) loaded software

• Job execution

- Read program, execute, output




---

---

---

---

---

---

---

---

Batch Systems (2)

• Standard applications

- Assembler, compiler, etc.
- Which one should be run?

• Job control language

- Special input (punched cards)
- Separates code and data in stream

---

---

---

---

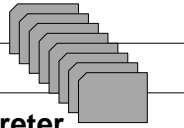
---

---

---

---

Resident Monitor



- **Control language interpreter**
  - E.g., cards with "\$" in first column
    - \$IBFTC, \$DATA, \$EOJ, \$EOD
- **Program loader**
  - System and application programs
- **Device drivers**
  - Interface to each type of I/O device

---

---

---

---

---

---

---

---

---

---

Spooling



- **I/O too slow**
  - E.g., card reader
- **Wastes expensive computer**
- **Solution**
  - Card reader data to tape or disk
  - As needed, read from faster storage
    - Overlap I/O and computation

---

---

---

---

---

---

---

---

---

---

Multiprogram Batch



- **More than one job resident**
  - System/application programs
- **Scheduler**
  - Some jobs ready; some awaiting I/O
  - Increase CPU utilization
  - Allocate devices, memory, CPU

---

---

---

---

---

---

---

---

---

---

Time Sharing Early 1970's

• Interactive computing

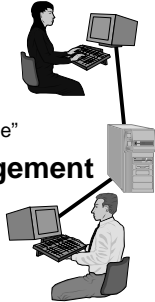
- On-line user access

• Response time important

- Not only throughput - "5 user rule"

• Secondary storage management

- User files, directories




---

---

---

---

---

---

---

---

Personal Computers (1) 1970's-80's

• Return to "early computer"?

- One user, one program; text mode

• Resident software

- E.g., BASIC in ROM

• Program loading

- Custom, DOS
- Audio cassette tape, diskette




---

---

---

---

---

---

---

---

Personal Computers (2) 1980's-90's

• Early multi-programming

- TSR's ("terminate, stay resident")

• Graphical user interface (GUI)

- Macintosh, Windows

• Enhanced multi-programming

- Scheduling, resource management

---

---

---

---

---

---

---

---

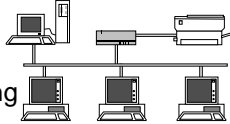
Multiple Processors Late 1970's+

• **Tightly coupled**

- Multi-processor systems
- Shared memory or busses

• **Distributed**

- Local area networks
- Cooperative processing




---

---

---

---

---

---

---

---

Real-Time Systems 1970's+

• **Embedded systems**

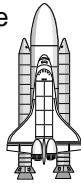
- Industrial control, avionics, etc.

• **Time constraints** (hard/soft)

- Unlike throughput or response time

• **Emphasis on OS efficiency**

- Always pressing hardware limits




---

---

---

---

---

---

---

---

Class Exercise

• **List OS's you have used**

- As programmer or end user

• **For each one**

- Hardware platform(s)
- Primary purpose
- Services provided

• **OS comparison**

- Identify common features
- Compare implementations

• **List observations**

- Principles, good & bad ideas, etc.

---

---

---

---

---

---

---

---