CHAPTER 1: INTRODUCTION

- What is an operating system?
- Early Systems
- Simple Batch Systems
- Multiprogramming Batched Systems
- Time-Sharing Systems
- Personal-Computer Systems
- Parallel Systems
- Distributed Systems
- Real-Time Systems

Operating system – a program that acts as an intermediary between a user of a computer and the computer hardware.

Operating system goals:

- Execute user programs and make solving user problems easier.

- Make the computer system *convenient* to use.

- Use the computer hardware in an *efficient* manner.

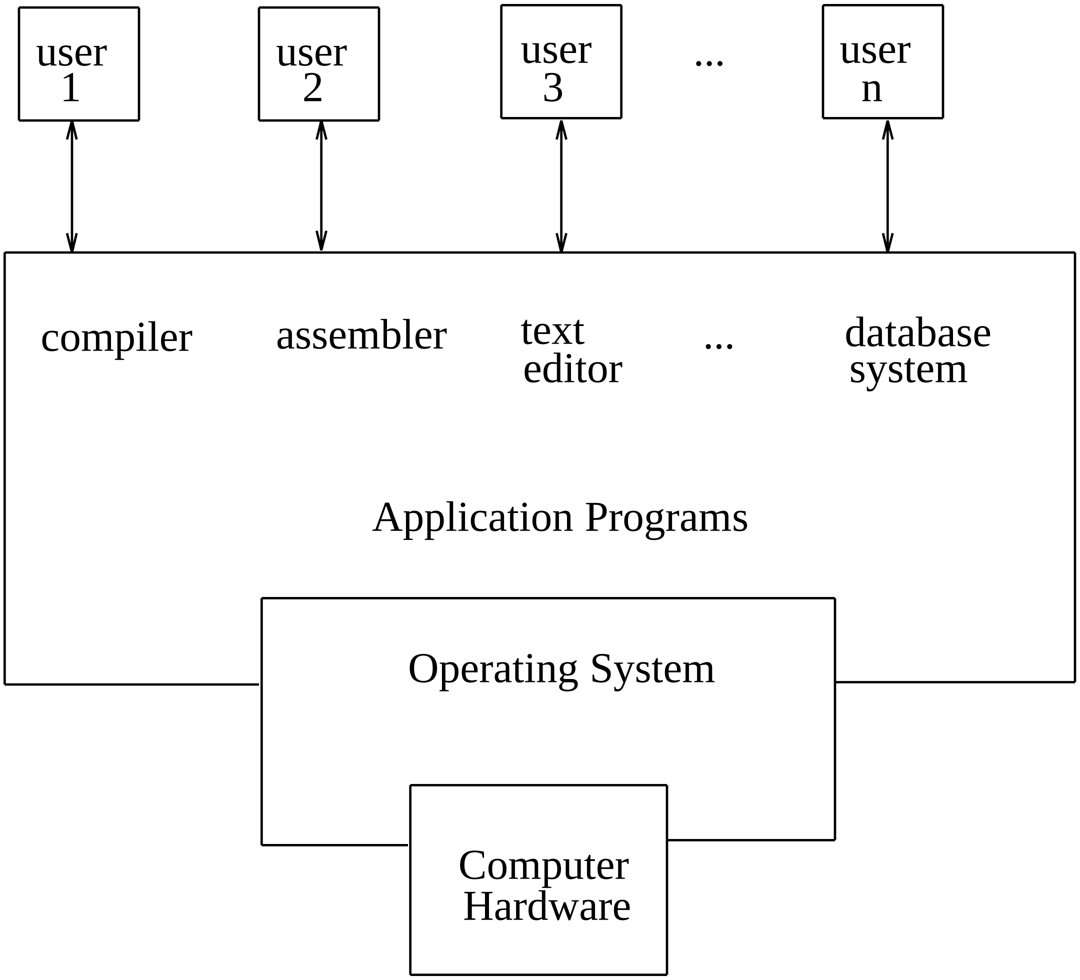
Computer System Components

1. Hardware – provides basic computing resources (CPU, memory, I/O devices).

2. Operating system – controls and coordinates the use of the hardware among the various application programs for the various users.

3. Applications programs – define the ways in which the system resources are used to solve the computing problems of the users (compilers, database systems, video games, business programs).

4. Users (people, machines, other computers).


user 1
user 2
user 3
...
user n
compiler
assembler
text editor
...
database system
Application Programs
Operating System
Computer Hardware

Operating System Definitions

- Resource allocator – manages and allocates resources.

- Control program – controls the execution of user programs and operation of I/O devices.

- Kernel – the one program running at all times (all else being application programs).

Early Systems – bare machine (early 1950s)

- Structure
  - Large machines run from console
  - Single user system
  - Programmer/User as operator
  - Paper tape or punched cards
- Early Software
  - Assemblers
  - Loaders
  - Linkers
  - Libraries of common subroutines
  - Compilers
  - Device drivers
- Secure
- Inefficient use of expensive resources
  - Low CPU utilization
  - Significant amount of setup time

Simple Batch Systems

- Hire an operator
- User ≠ operator
- Add a card reader
- Reduce setup time by batching similar jobs
- Automatic job sequencing – automatically transfers control from one job to another. First rudimentary operating system.
- Resident monitor
  - initial control in monitor
  - control transfers to job
  - when job completes control transfers back to monitor

Problems:

1) How does the monitor know about the nature of the job (e.g., Fortran versus Assembly) or which program to execute?

2) How does the monitor distinguish
   a) job from job?
   b) data from program?

Solution: introduce control cards

Control Cards

- Special cards that tell the resident monitor which programs to run.

  $JOB
  $FTN
  $RUN
  $DATA
  $END

- Special characters distinguish control cards from data or program cards:

  $ in column 1

  // in column 1 and 2

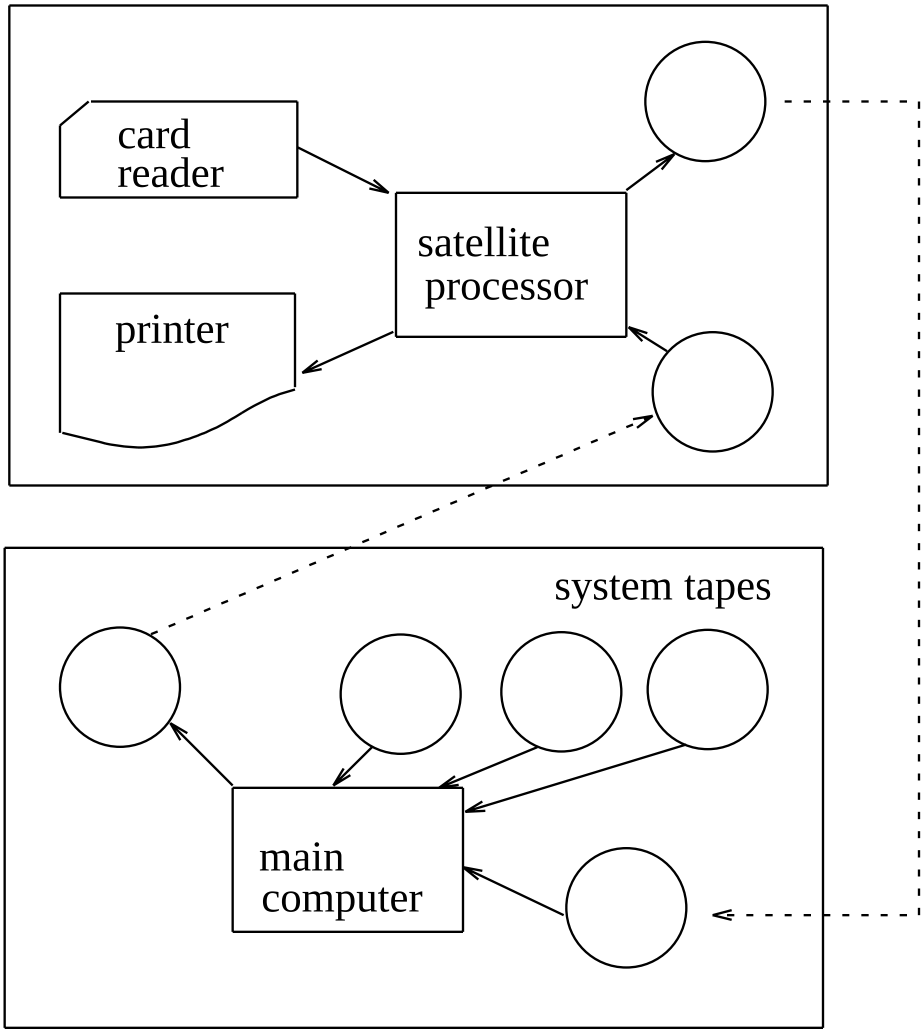
  7-9 in column 1

- Parts of resident monitor

  - Control card interpreter – responsible for reading and carrying out instructions on the cards.
  - Loader – loads systems programs and applications programs into memory.
  - Device drivers – know special characteristics and properties for each of the system's I/O devices.

- Problem: Slow Performance – since I/O and CPU could not overlap, and card reader very slow.

- Solution: Off-line operation – speed up computation by loading jobs into memory from tapes and card reading and line printing done off-line.

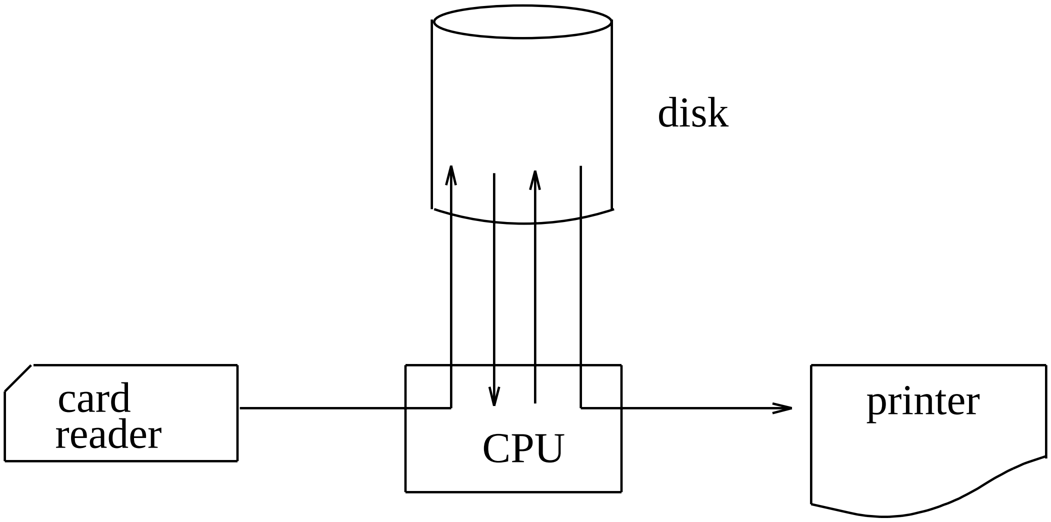


- Advantage of off-line operation – main computer not constrained by the speed of the card readers and line printers, but only by the speed of faster magnetic tape units.

- No changes need to be made to the application programs to change from direct to off-line I/O operation.

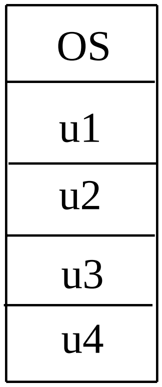
- Real gain – possibility of using multiple reader-to-tape and tape-to-printer systems for one CPU.

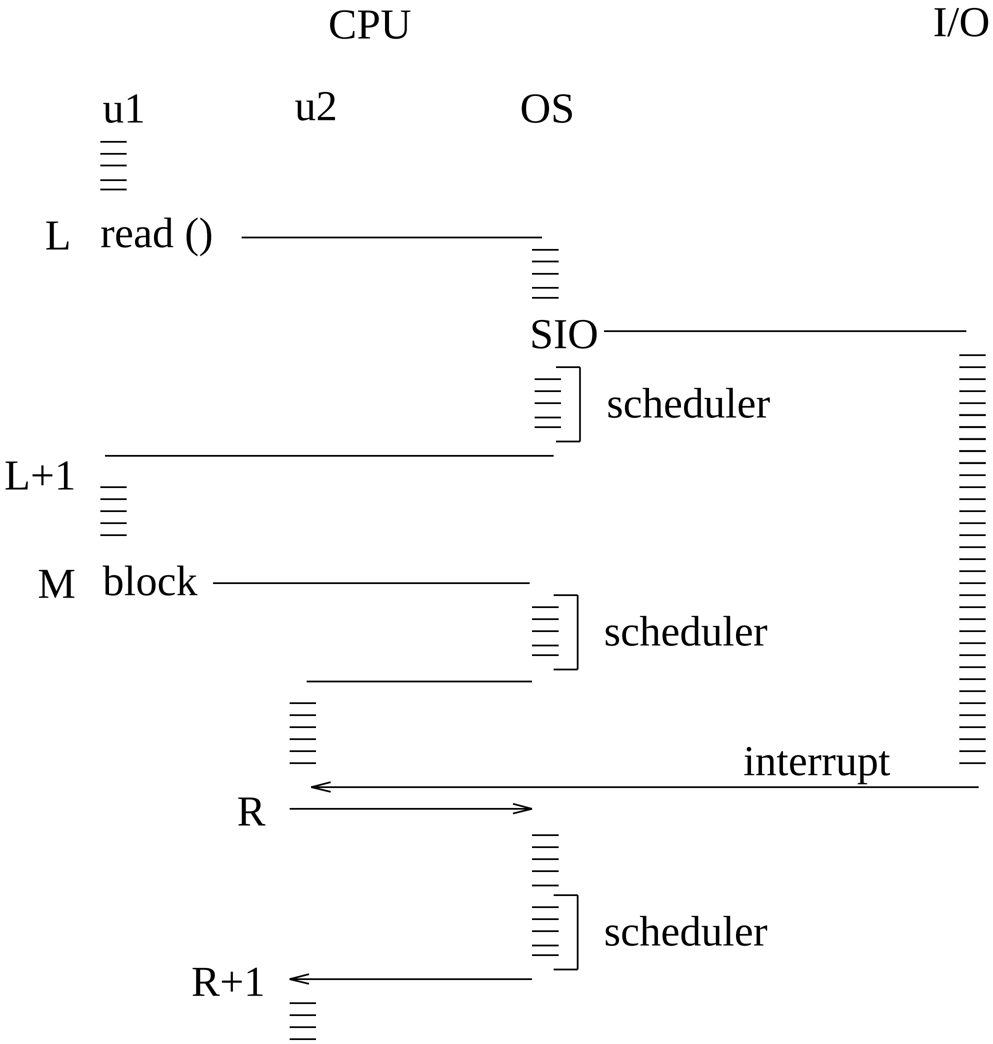
Spooling – overlap the I/O of one job with the computation of another job.



- While executing one job, the operating system:
  - reads the next job from the card reader into a storage area on the disk (job queue).
  - outputs the printout of previous job from disk to the line printer.
- *Job pool* – data structure that allows the operating system to select which job to run next, in order to increase CPU utilization.

Multiprogrammed Batch Systems – several jobs are kept in main memory at the same time, and the CPU is multiplied among them.





OS Features Needed for Multiprogramming

- I/O routine supplied by the system.
- Memory management – the system must allocate the memory to several jobs.
- CPU scheduling – the system must choose among several jobs ready to run.
- Allocation of devices.

Time-Sharing Systems– Interactive Computing

- The CPU is multiplied among several jobs that are kept in memory and on disk (the CPU is allocated to a job only if the job is in memory).

- A job is swapped in and out of memory to the disk.

- On-line communication between the user and the system is provided; when the operating system finishes the execution of one command, it seeks the next ‘‘control statement’’ not from a card reader, but rather from the user’s keyboard.

- On-line file system must be available for users to access data and code.

Personal-Computer Systems

- *Personal computers* – computer system dedicated to a single user.

- I/O devices – keyboards, mice, display screens, small printers.

- User convenience and responsiveness.

- Can adopt technology developed for larger operating systems; often individuals have sole use of computer and do not need advanced CPU utilization or protection features.

Parallel Systems – multiprocessor systems with more than one CPU in close communication.

- *Tightly coupled* system – processors share memory and a clock; communication usually takes place through the shared memory.

- Advantages of parallel systems:
  - Increased *throughput*
  - Economical
  - Increased reliability
    - *graceful degradation*
    - *fail-soft* systems

- *Symmetric multiprocessing*
  - Each processor runs an identical copy of the operating system.
  - Many processes can run at once without performance deterioration.
- *Asymmetric multiprocessing*
  - Each processor is assigned a specific task; master processor schedules and allocates work to slave processors.
  - More common in extremely large systems.

Distributed Systems – distribute the computation among several physical processors.

- *Loosely coupled* system – each processor has its own local memory; processors communicate with one another through various communication lines, such as high-speed buses or telephone lines.
- Advantages of distributed systems:
  - Resource sharing
  - Computation speed up – load sharing
  - Reliability
  - Communication

Real-Time Systems

- Often used as a control device in a dedicated application such as controlling scientific experiments, medical imaging systems, industrial control systems, and some display systems.

- Well-defined fixed-time constraints.

- *Hard real-time* system.
  - Secondary storage limited or absent; data stored in short-term memory, or read-only memory (ROM).
  - Conflicts with time-sharing systems; not supported by general-purpose operating systems.

- *Soft real-time* system.
  - Limited utility in industrial control or robotics.
  - Useful in applications (multimedia, virtual reality) requiring advanced operating-system features.