CSC 453 Operating Systems

Lecture 1: An Introduction

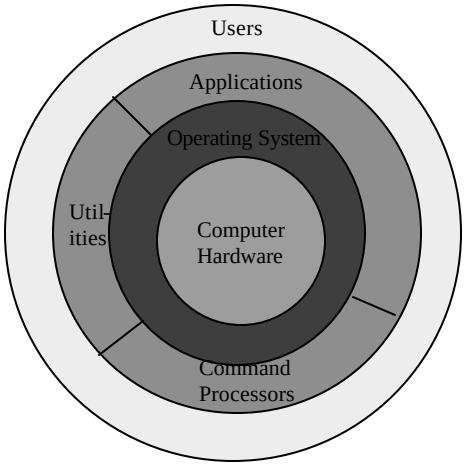
What Is an Operating System?

- An operating system is the software that serves as an interface between the user (or his/her software applications) and the computer's hardware.

What Is The Purpose Of An Operating System?

- The operating system provides an environment where a user or a group of users can use the resources of a computer easily and efficiently.

The Components Of a Computer System



The Operating System Is a Resource Allocator

- A computer system has several resources:
  - CPU time
  - Memory
  - Disk Drives and the Files Stored On Them
  - I/O Devices
- These resources have to be shared and the operating systems has to ensure that programs get the resources that they need

The Operating System Is a Control Program

- Operating systems control the execution of a program:
  - scheduling their use of CPU time
  - allocation the area of memory that they use
  - assigning their disk operations
  - performing their input and output
- This is ensure their accurate and proper use of the computer system's resources.

Design Goals

- It is far easier to define an operating system by what it ***does*** than by what it ***is***.
- An operating system has two main performance goals:
  - the convenience of the user
  - the efficient operations of the computer systems as a whole.

Operating Systems and Computer Architecture

- The design and performance of the operating system is heavily dependent on the computer's architecture.
- Similarly, the desire to employ certain features of operating system design has influenced the design of many computers.

Historical Perspective: Early Machines

- The earliest computers were large machines operated directly by programmers controlling them from their consoles.
- Programmers had to load programs manually, monitor execution and write all the necessary software to enable them to use their input/output devices.

The Start of Systems Programming

- Programmers developed software to perform commonly-used tasks such as:
  - controlling input/output devices
  - assembling, linking and loading programs.
- Programmers developed libraries of procedures that were commonly used to avoid having to rewrite procedures to perform the same jobs over and over again.

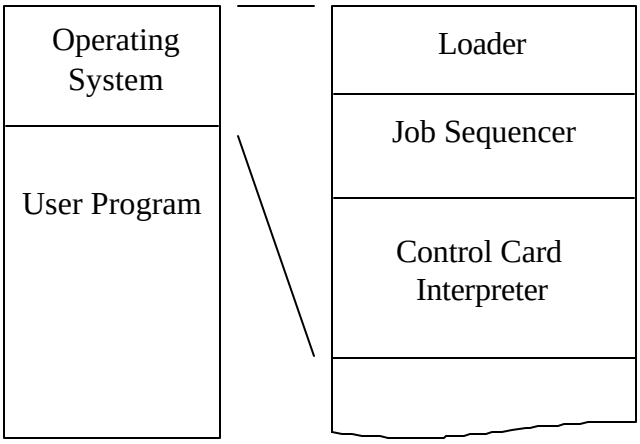
The Introduction of Compilers

- Compilers for FORTRAN, COBOL and other languages simplified programming but complicated computer operations.
- The biggest complication was increased set-up time and made the use of the computer far less efficient

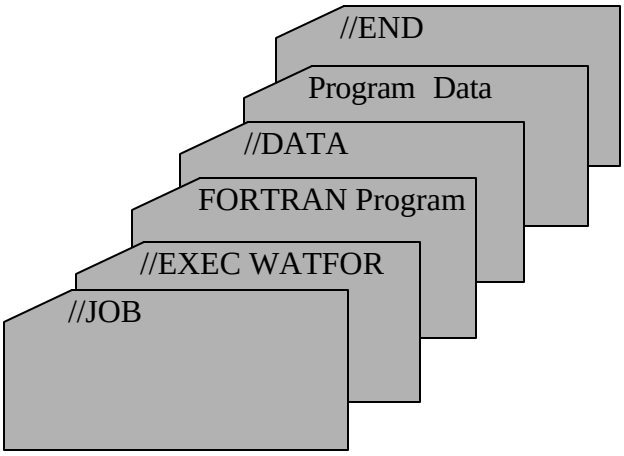
The Next Step: The Professional Operator

- IDEA - *Hire an operator!!*
- The advantage is the more efficient use of computer time. as programs are run in batches, leading to the batch system.
- Eventually a program (an ***operating system***) replaced the human operator.

Simple Batch Systems

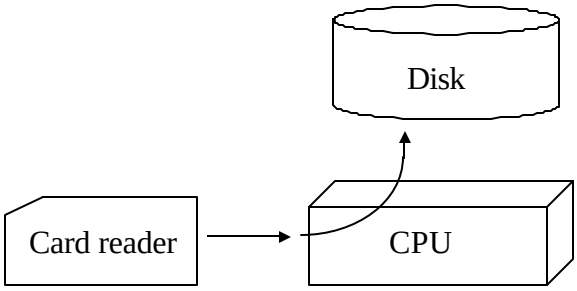


Card Deck For a Batch Program

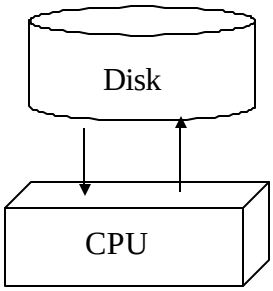


Offline Operation

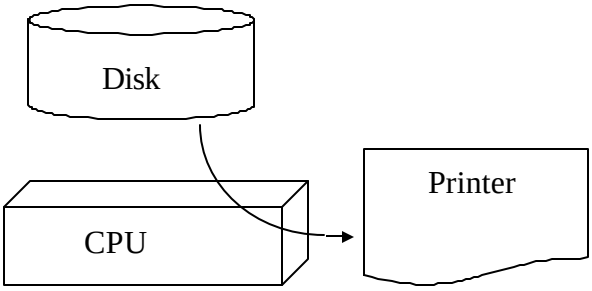
- SPOOLing - ***S***imultaneous ***P***eripheral ***O***peration ***O***n-***L***ine



Executing The SPOOLed Program



SPOOLed Output



Multiprogrammed Batched Systems

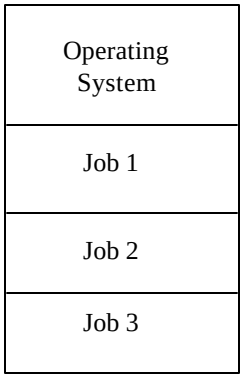
Input/Output is S L O W.....
but.....
The CPU is *fast*.....

How do we avoid having the CPU sit idle?

Multiprogrammed Batched Systems

- ***IDEA!!*** Have the CPU start working on another job!!

Memory Layout in a Multiprogrammed System



Historical Perspective: Atlas

- Atlas was designed at the University of Manchester in the 1950s.
- Atlas used a core memory as a cache to implement demand paging.

Historical Perspective: THE

- THE was designed at the *Technische Hogeschool* at *Eindhoven*.
- THE was a batch system which used a layered structure and synchronized jobs ***concurrently*** using ***semaphores***.
- THE employed the ***banker's algorithm*** to avoid ***deadlock***.

Time-Sharing Systems

- Time-sharing systems allowed multiple users to work interactively on a computer.
- Time-sharing systems are more complex and expensive than batch systems and did not become common until the 1970s.
- Time-sharing systems take advantage of the great difference between CPU speed and interactive I/O speed.

CTSS

- CTSS (Compatible Time-Sharing System) was designed at MIT in the early 1960s.
- It was implemented on an IBM 7090 and could support up to 32 interactive users.

MULTICS

- MULTICS was designed at MIT as an extension of the CTSS project.
- The goal was to create a time-sharing utility that could run continuously and use a large file system of shared programs and data.
- MULTICS was a jointed program of MIT, GE and AT&T Bell Labs running on a GE645.

UNIX

- AT&T dropped out of the MULTICS project in 1969.
- Ken Thompson and Dennis Ritchie designed UNIX to contain many features of MULTICS's that they liked while making changes and enhancements.
- UNIX has influences most interactive operating systems created since then.

Personal Computers

- As hardware declined in price, it became practical to build computers for individual users. These became known as PCs, first appearing in the 1970s.
- PC operating systems were initially simple systems, lacking protection features of larger computers.

PC Operating Systems

- As PCs became more powerful, their operating systems became more powerful as well.
- MS-DOS (a single tasking system) gave way to OS/2 and Windows.

Workstations

- The more powerful desktop machines had computing power equal to minicomputers and became known as ***workstations***.
- These computers frequently used a version of UNIX as their operating system.

Parallel Systems

- Most computers have one processors, but ***multiprocessor systems*** are becoming more common.
  - These systems have their processors in close communications sharing the computer's bus, memory, clock and peripheral devices. These are called ***tightly-coupled systems***.

Why Build Multiprocessor Systems?

- Increased throughput
- Their processors can share peripherals, cabinets, and power supplies.
- Increased realiability
  - ***Graceful degradability*** – providing service proportional to the surviving hardware
  - ***Fault-tolerant*** - systems designed for graceful degradability

Symmetric vs. Asymmetric Multiprocessing

- ***Symmetric multiprocessing*** – Each processor runs the same copy of the operating system
- ***Asymmetric multiprocessing*** – Each processor has a specialized task.

Distributed Systems

- Another recent trend is toward systems where processors do not share memory or a clock. These are called ***loosely-coupled systems*** or ***distributed systems***.

Why Distributed Systems?

- Resource sharing
- Computation speedup through ***load sharing***
- Reliability
- Communication

Real-Time Systems

- Real-time systems are computer systems that have rigid time restrictions for completing many or all of their tasks.
- Examples includes:
  - Automobile fuel injection systems
  - Medical imaging systems
  - Robots

Hard vs. Soft Real-Time Systems

- Hard real-time systems must guarantee that critical tasks be completed on time.
- In soft real-time systems, critical real-time tasks get priority over other tasks and still needed to be bounded (they cannot wait indefinitely).