Operating Systems (Client)

Lecture 1
Introduction to Operating Systems

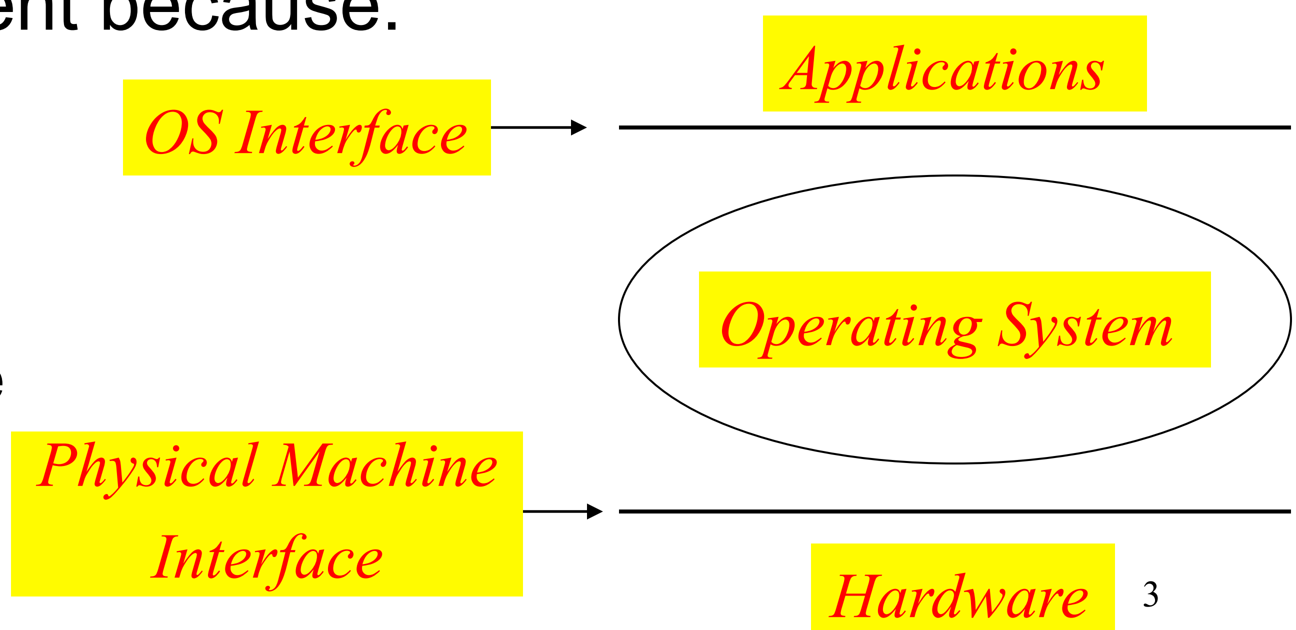
Dr. Kevin Farrell

Topics Covered in this Lecture

- What is an Operating System?
- What do Operating Systems do?
- What Resources need to be managed?
- What's in an Operating System?
- Major Issues with Operating System
- Brief History of Operating Systems

What is an Operating System?

- <u>Definition</u>: An Operating System (OS) provides a virtual machine on top of the hardware which is more convenient than the raw hardware interface
- An OS is "All of the code you didn't write"
- More convenient because:
  - Simpler
  - More reliable
  - More secure
  - More portable
  - More efficient



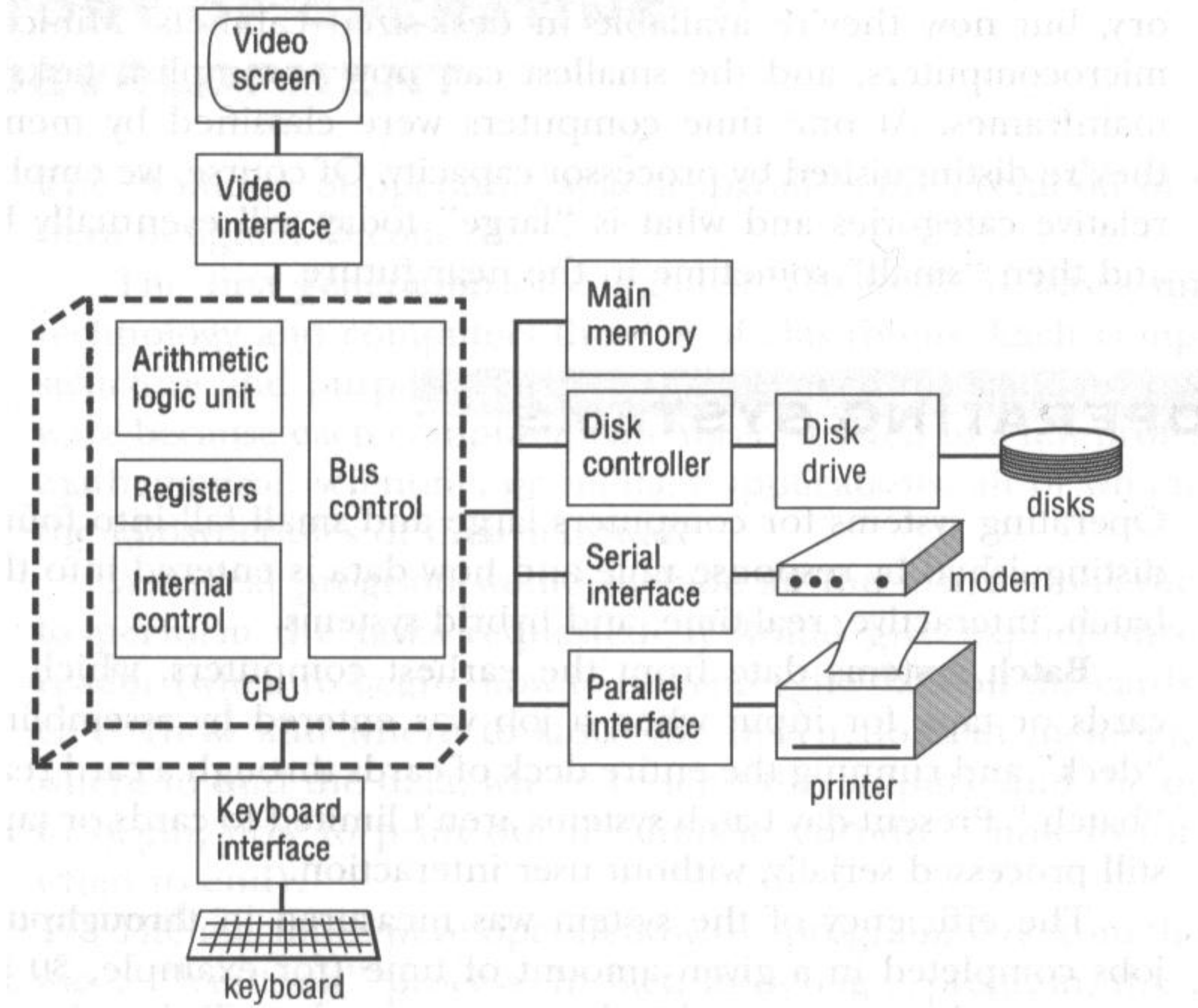
What do Operating Systems Do?

- Manage physical and virtual resources
- Provide users with a well-behaved environment
- Define a set of logical resources (objects) and a set of well-defined operations on those resources (i.e. an interface to those objects)
- Provide *mechanisms* and *policies* for the control of resources
- Control how different users and programs interact

What Resources need to be Managed?

- The CPU(s)
- Memory
- Storage Devices (disks, tapes, etc.)
- Input Devices (keyboard, mouse, cameras, etc.)
- Output Devices (printers, displays, speakers, etc.)
- Networks

What Resources need to be Managed?



What’s in an OS – Logical Structure

Applications

Quake | Word Processor

System Utils | Windowing & Graphics | Shells

OS Interface

Machine Independent Services

Naming | Windowing & Graphics

Networking | Virtual Memory | Access Control

Generic I/O | File System | Process Management

Machine Dependent Services

Device Drivers | Memory Management

Physical Machine Interface

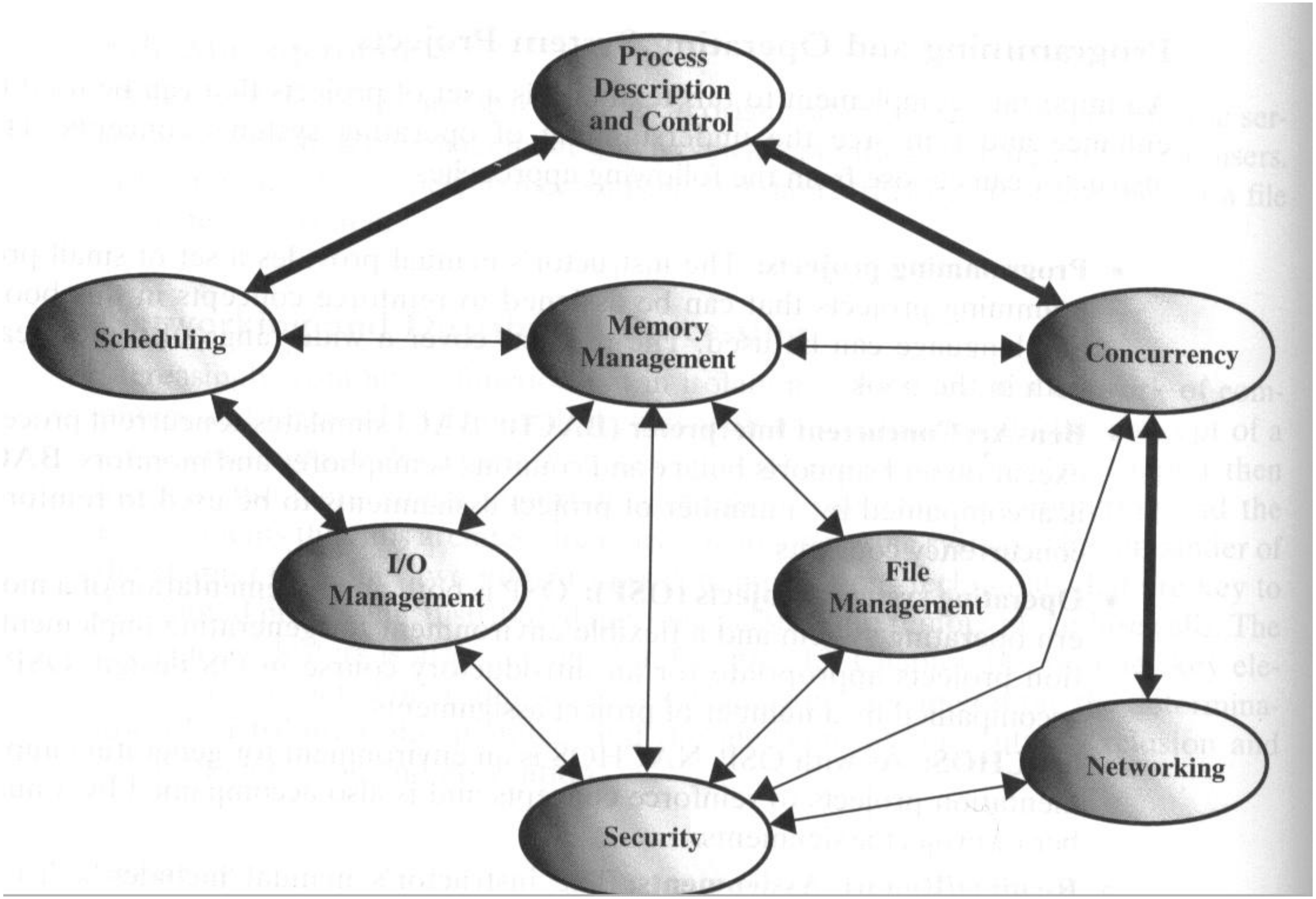
Interrupts, Cache, Physical Memory, Hardware Devices

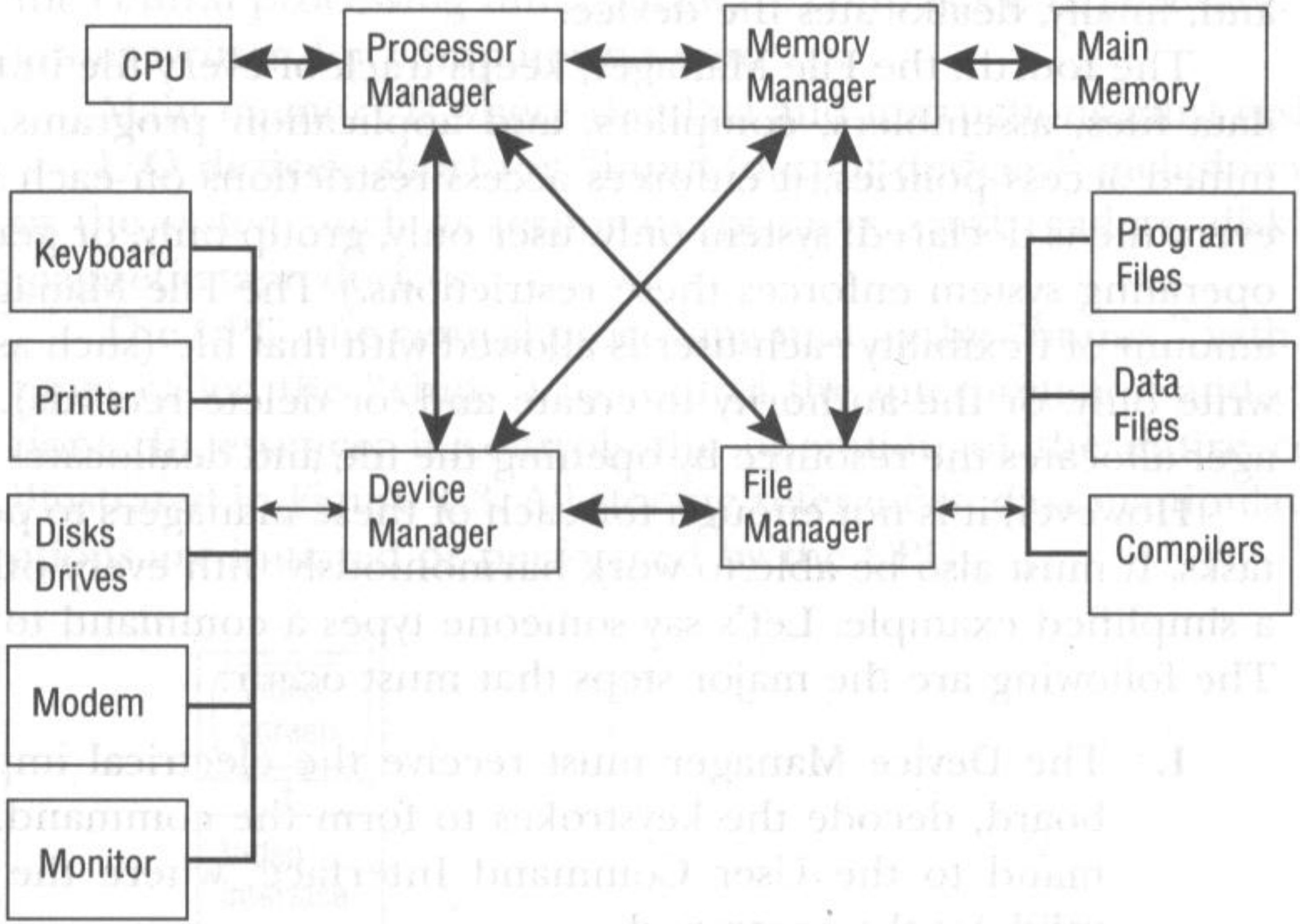
Major Issues in Operating Systems #1

- **Structure**: how is an OS organised?
- **Sharing**: how are resources shared among users?
- **Naming**: how are resources named by users or programs
- **Protection**: how is one user/program protected from another
- **Security**: how to authenticate, control access, and secure privacy
- **Performance**: why is it so slow?
- **Reliability & Fault Tolerance**: how do we deal with failures?
- **Extensibility**: how do we add new features?

Major Issues in Operating Systems #2

- **Communication**: how can we exchange information?
- **Concurrency**: how are apparently parallel activities created and controlled?
- **Scale & Growth**: what happens as demands & resources increase?
- **Persistence**: how to make data outlast the processes that created them
- **Compatibility**: can we ever do anything new?
- **Distribution**: accessing the world of information
- **Accounting**: who pays the bills, and how do we control resource usage?


Process Description and Control
Scheduling
Memory Management
Concurrency
I/O Management
File Management
Networking
Security


CPU
Processor Manager
Memory Manager
Main Memory
Keyboard
Printer
Disks Drives
Modem
Monitor
Device Manager
File Manager
Program Files
Data Files
Compilers

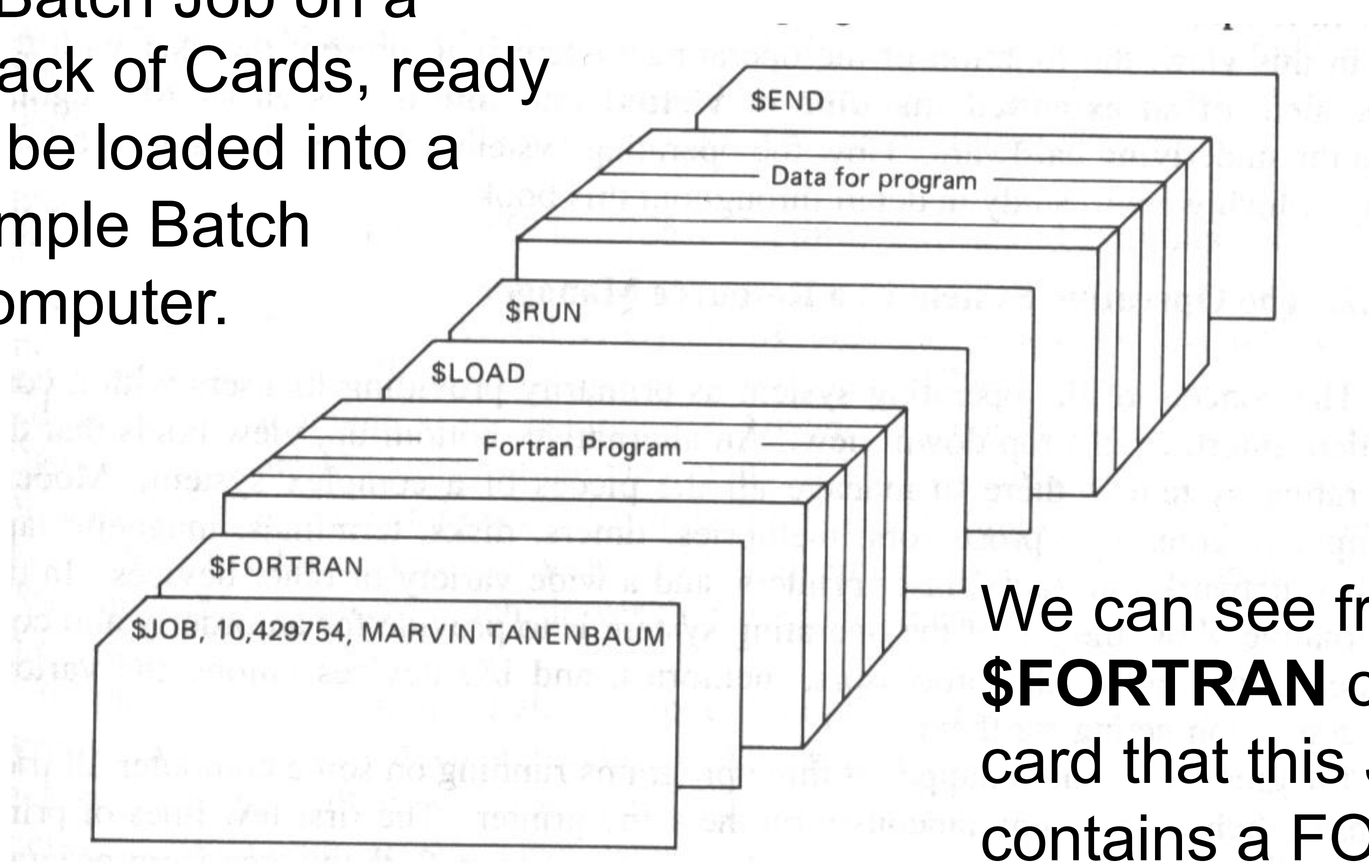
A Brief History of Operating Systems

- Initially, the OS was just a run-time library
  - You linked your application with the OS, loaded the whole program into memory, and ran it
  - But, how do you get it into the computer?
    - Answer: Through the control panel!

- **Simple batch systems**
  - Permanently resident OS in primary memory
  - It loaded a single job from card reader, ran it, and loaded the next job...
  - *Control cards* in the input file told the OS what to do
  - *Spooling* allowed jobs to be read ahead of time onto tape/disk or into memory

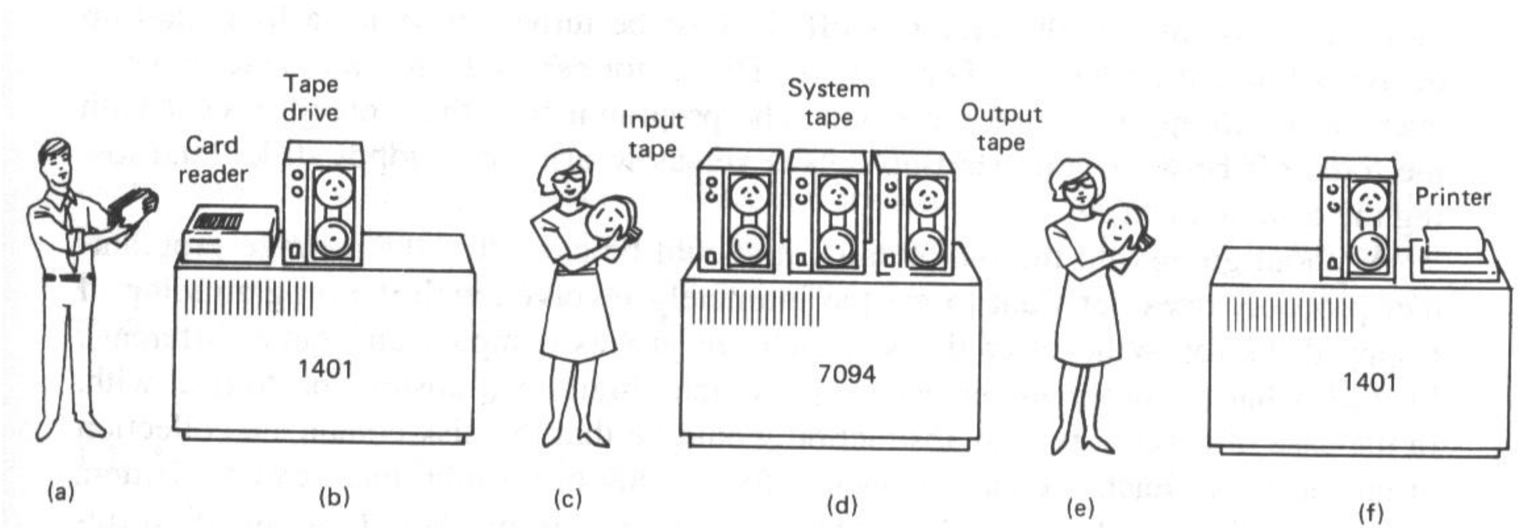
Simple Batch System Cards

A Batch Job on a Stack of Cards, ready to be loaded into a Simple Batch Computer.



We can see from the **$FORTRAN** control card that this Job contains a FORTRAN program

Simple Batch System: Spooling onto Tape



Spool = Simultaneous Peripheral Operations On-Line

Multiprogrammed Batch Systems

- *Multiprogramming Batch* systems:
  - Keeps multiple runnable jobs loaded in memory
  - Overlaps I/O processing of a job with computation of another
- Provided increased utilization:
  - Benefits from I/O devices that can operate asynchronously
  - Requires the use of interrupts and DMA (Direct Memory Access)
  - Optimizes for *throughput* at the cost of *response time*

Interactive Systems - Time-Sharing

- *Timesharing* supported *interactive* computer use
  - Each user connects to a central machine through a cheap terminal; feels as if she has the entire machine
  - Based on time-slicing - dividing CPU equally among the users
  - Permitted active viewing, editing, debugging, participation of users in the execution process
  - Security mechanisms required to isolate users from each other
  - Requires memory protection hardware for isolation
  - Optimizes for *response time* at the cost of *throughput*

Hybrid Systems

- These are a combination of batch and interactive systems
  - The system appears to be interactive since individuals can access the system via terminals and get a fast response, but the system actually accepts and runs batch programs in the background when the interactive load is low
- Many large computer systems are hybrid in nature; for example: special purpose Supercomputers

Distributed Operating Systems

- Distributed systems facilitate use of geographically distributed resources
  - Computers connected by wires
  - no shared memory or clock (i.e. each CPU only has access to the memory in its computer, and each CPU “ticks” at its own rate)
- Require support for communication between parts of a job or different jobs: ‘**Inter-process communication (IPC)**’

- Users benefit from sharing of distributed resources, hardware and software (Resource utilisation and access)
- Also permits some parallelism, but speeding up execution of processes is not the primary objective

Parallel Operating Systems

- Objective: Support parallel applications wishing to get speed up of computationally complex tasks; For eg: Military, Weather Prediction, Oil Exploration, Traffic Simulation, Film Special Effects & Animation, Car Industry
- Needs basic primitives (simple rules) for dividing one task into multiple parallel activities
- OS must support
  - efficient communication between those multiple parallel activities
  - synchronisation of activities to coordinate sharing of information
- It's common now to use networks (*clusters*) of high-performance ordinary PCs/workstations as a parallel computer => '**COTS**' = Commodity Off The Shelf

Real-time Operating Systems (RTOS)

- **<u>Goal</u>**: To cope with rigid time constraints
- Hard real-time
  - OS guarantees that applications ***will*** meet their deadlines
  - <u>Eg</u>: CD player, TCAS, health monitors, factory control
- Soft real-time
  - OS provides prioritisation, on a best-effort basis
  - No deadline guarantees, but bounded delays
  - <u>Eg</u>: most electronic appliances
- Real-time means “predictable”
  - Some RTOS are very fast, others are NOT necessarily fast

Personal Computing

- Computers are cheap, so we could give everyone a dedicated computer!
- How did this come about?
  - Initially, the OS was just a library. This was due to technology available at the time ("Hardware Constraints")
  - Improving Technology Allowed Computer Scientists to implemented new (clever) techniques:
    - Multiprogramming
    - Memory Protection
    - Other advances

Ubiquitous Computing

- The decreased cost of processing makes it possible to embed computers everywhere. Each “embedded” application needs its own control software:
  - PDAs, cell phones, intelligent appliances, etc.
- In the near future, you will have 100s of these devices
  - If not already
- Poses lots of problems for current systems
  - Structure, naming, scaling, security, etc.

Using Lessons from History for Planning Future OSes

- An Important Point is not that batch systems were ridiculous. They were exactly the right tradeoffs for the status and price of the technology at the time

- The tradeoffs changed:

	1981	2001	Factor
MIPS	1	1000	1000
$/MIPS	$100000	$5000	20000
DRAM	128KB	256MB	2000
Disk	10MB	80GB	8000
Net Bandwidth	9600 b/s	100 Mb/s	10000
# Users	>> 10	<= 1	0.1

- If we understand these fundamentals => we can design better systems for tomorrow's tradeoffs

Technology Advances: Hardware => Miniaturisation, and Software Tools & Ideas => OSes