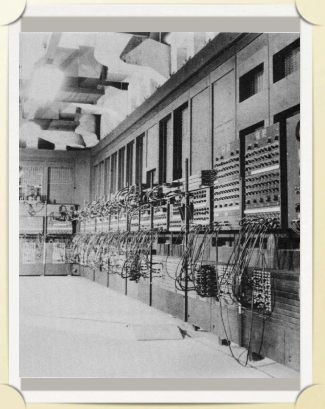
A Short History of Operating Systems

History of Operating Systems: Phases

- Phase 1: Hardware is expensive, humans are cheap
  - User at console: single-user systems
  - Batching systems
  - Multi-programming systems

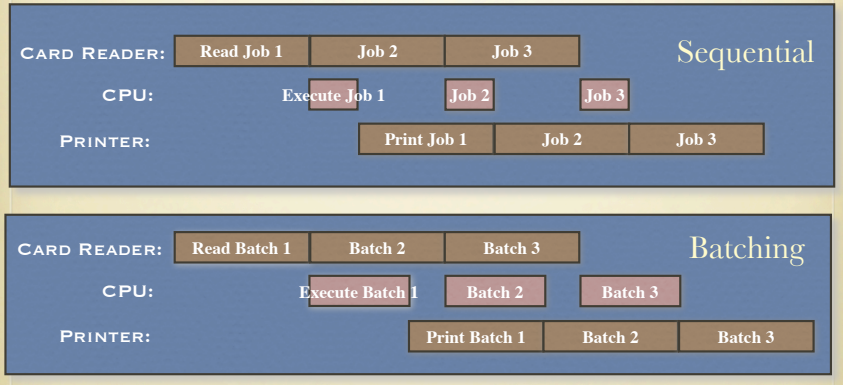
Hand programmed machines (1945-1955)

- Single user systems
- OS = loader + libraries of common subroutines
- Problem: low utilization of expensive components



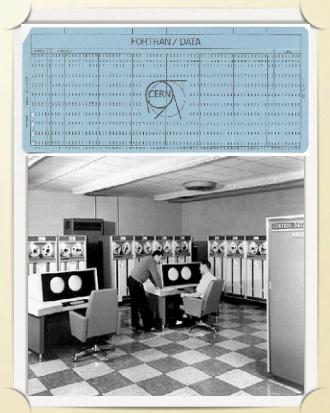
$$\frac{\textit{time device busy}}{\textit{observation interval}} = \%\ \textit{utilization}$$

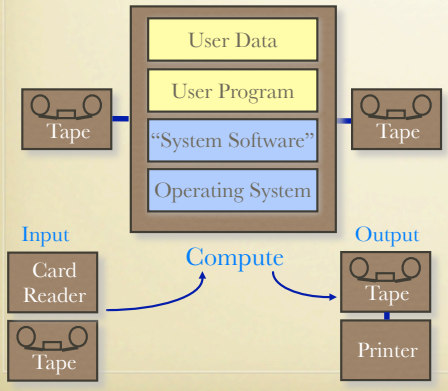
Batch/Off-line processing (1955-1965)



Batch Processing (1955-1965)

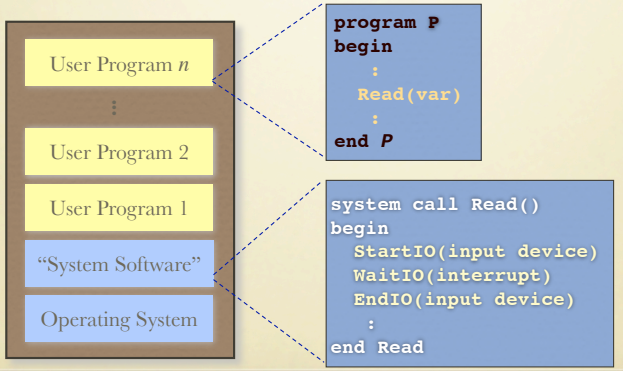
Operating system = loader + sequencer + output processor





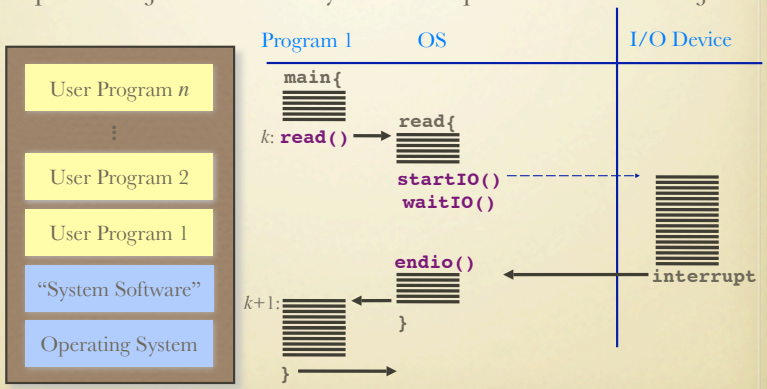
Multiprogramming (1965-1980)

Keep several jobs in memory and multiplex CPU between jobs



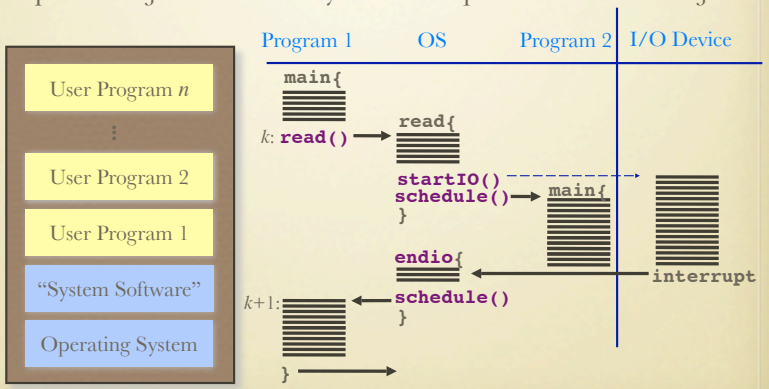
Multiprogramming (1965-1980)

Keep several jobs in memory and multiplex CPU between jobs



Multiprogramming (1965-1980)

Keep several jobs in memory and multiplex CPU between jobs

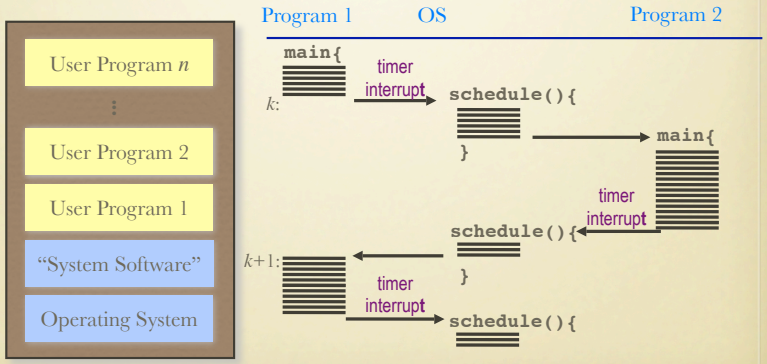


History of Operating Systems: Phases

- Phase 1: Hardware is expensive, humans are cheap
  - User at console: single-user systems
  - Batching systems
  - Multi-programming systems
- Phase 2: Hardware is cheap, humans are expensive
  - Time sharing: Users use cheap terminals and share servers

Timesharing (1970-)

A timer interrupt is used to multiplex CPU between jobs



History of Operating Systems: Phases

- Phase 1: Hardware is expensive, humans are cheap
  - User at console: single-user systems
  - Batching systems
  - Multi-programming systems
- Phase 2: Hardware is cheap, humans are expensive
  - Time sharing: Users use cheap terminals and share servers
- Phase 3: Hardware is very cheap, humans are very expensive
  - Personal computing: One system per user
  - Distributed computing: many systems per user
  - Ubiquitous computing: LOTS of systems per users

Operating Systems for PCs

Personal computing systems

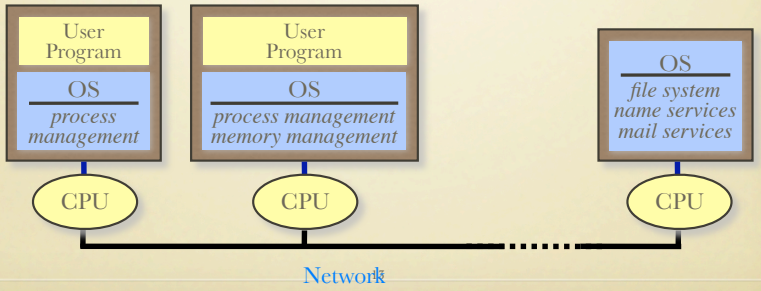
- Single user
- Utilization is no longer a concern
- Emphasis is on user interface and API
- Many services & features not present

Evolution

- Initially: OS as a simple service provider (simple libraries)
- Now: Multi-application systems with support for coordination

Distributed Operating Systems

- Abstraction: present a multi-processor system as a single processor one.
- New challenges in consistency, reliability, resource management, performance, etc.
- Examples: SANs, Oracle Parallel Server



Ubiquitous Computing

- PDAs, cellular phones, sensors
- Challenges
  - Small memory size
  - Slow processor
  - Battery concerns
  - Scale
  - Security
  - Naming

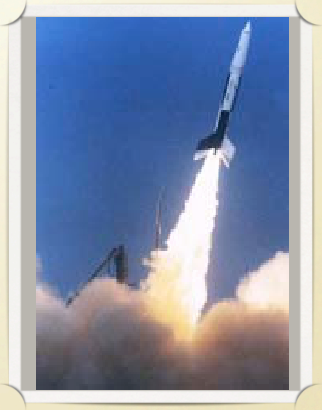
History of Operating Systems: Phases

- Phase 1: Hardware is expensive, humans are cheap
  - User at console: single-user systems
  - Batching systems
  - Multi-programming systems
- Phase 2: Hardware is cheap, humans are expensive
  - Time sharing: Users use cheap terminals and share servers
- Phase 3: Hardware is very cheap, humans are very expensive
  - Personal computing: One system per user
  - Distributed computing: many systems per user
  - Ubiquitous computing: LOTS of systems per user
- Richer Services
  - Real-time operating systems

Real time Operating System

- Goal: To cope with rigid time constraints
- Hard real time:
  - OS guarantees that application will meet deadline
  - Examples: health monitors, factory control, traffic collision avoidance systems (TCAS)
- Soft real time
  - OS provides prioritization, on a best effort basis
  - No critical failure if time constraint is violated
  - Example: most electronic appliances

"Real time" means *predictable* NOT *fast*



Over the years

- Not that batch systems were ridiculous
  - They were exactly right for the tradeoffs at the time
- The tradeoffs change

	1981	2006	Factor
MIPS	1	6570/CPU	1,000
$/MIPS	$100000	$0.11	900,000
DRAM	128KB	2GB	8,000
Disk	10MB	250GB	25,000
Net Bandwidth	9600 b/s	100 Mb/s	10,000
# Users	>> 10	<= 1	0.1
#CPU	1	4	4

- Need to understand the fundamentals
  - So you can design better systems for tomorrow's tradeoffs