

Operating Systems: Basic Concepts and History

What is an Operating System?

- ◆ A program and an interface
 - An abstract virtual machine
 - A set of abstractions that simplify application design
 - ◆ Files instead of "bytes on a disk"
- ◆ For any OS area (CPU scheduling, file systems, memory management), begin by asking two questions
 - What's the hardware interface? (The Physical Reality)
 - What is the application interface? (The Nicer Interface)
- ◆ Key questions:
 - Why is the application interface defined the way it is?
 - Should we push more functionality into applications, the OS, or the hardware?
 - What are the tradeoffs between programmability, complexity, and flexibility?

Operating System Functions

- ◆ Service provider
 - Provide standard facilities
 - ◆ File system
 - ◆ Standard libraries
 - ◆ Window system
 - ◆ ...
- ◆ Coordinator: three aspects
 - **Security**: prevent jobs from interfering with each other
 - **Communication**: enable jobs to interact with each other
 - **Resource management**: facilitate sharing of resources across jobs
- ◆ Examples
 - Single-function devices (embedded controllers, Nintendo, ...)
 - ◆ OS provides a collection of standard services
 - Multi-function/application devices (workstations and servers)
 - ◆ OS manages application interactions

Why do we need operating systems?

- ◆ **Convenience**
 - Provide a high-level abstraction of physical resources
 - Enable the construction of more complex software systems
 - Enable portable code
- ◆ **Efficiency**
 - Share limited or expensive physical resources
 - Provide protection

Evolution of Operating Systems

- ◆ Why do operating systems change?
 - Key functions: hardware abstraction and coordination
 - Principle: Design tradeoffs change as technology changes
 - Underlying technology has changed immensely over the past two decades !!
- ◆ Comparing computing systems from 1981 and 2000

	1981	2000	Factor
MIPS	1	1000	1000
\$/SPECInt	\$100K	\$2	50000
DRAM size	128KB	256MB	2000
Disk size	10MB	10GB	1000
Net BW	9600 bps	100 Mb/s	10000
Address bits	16	64	4
Users/machine	100	<1	100

5

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: lots of systems per user
- ◆ Phase 4: Richer services
 - Real-time operating systems

6

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: lots of systems per user
- ◆ Phase 4: Richer services
 - Real-time operating systems

7

A Brief History of Operating Systems Hand programmed machines ('45-'55)

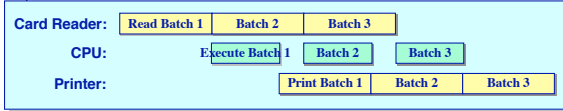
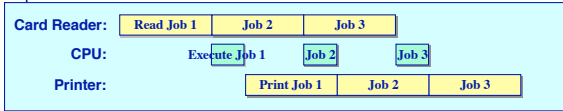
- ◆ Single user systems
- ◆ OS = loader + libraries of common subroutines
- ◆ Problem: low utilization of expensive components

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

8

Batch/Off-line processing ('55-'65)

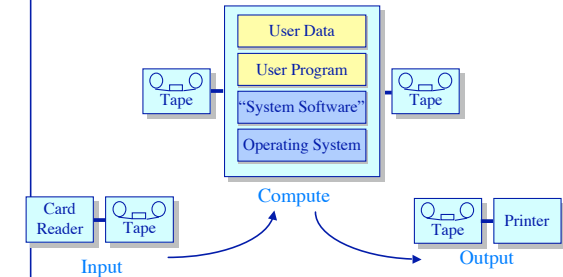
- ◆ Batching v. sequential execution of jobs



9

Batch processing ('55-'65)

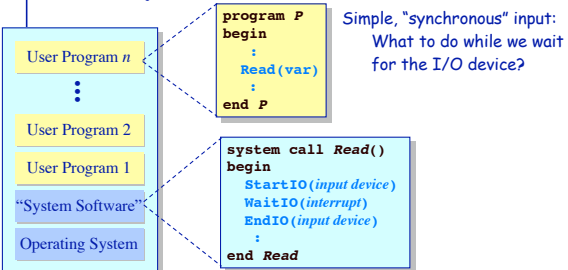
- ◆ Operating system = loader + sequencer + output processor



10

Multiprogramming ('65-'80)

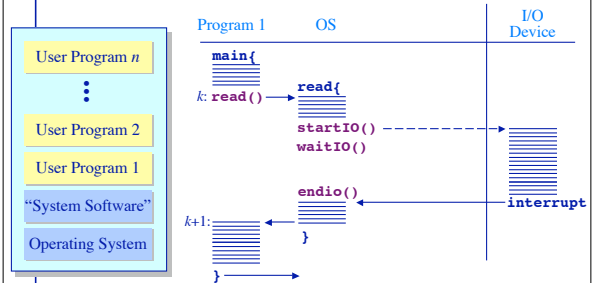
- ◆ Keep several jobs in memory and multiplex CPU between jobs



11

Multiprogramming ('65-'80)

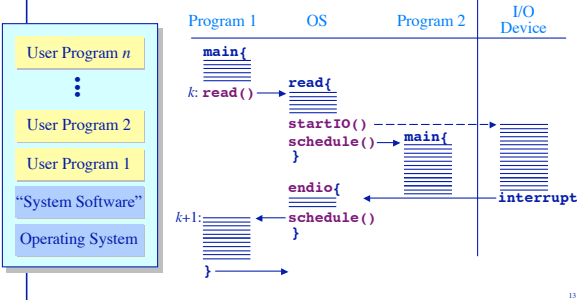
- ◆ Keep several jobs in memory and multiplex CPU between jobs



12

Multiprogramming ('65-'80)

- ◆ Keep several jobs in memory and multiplex CPU between jobs



13

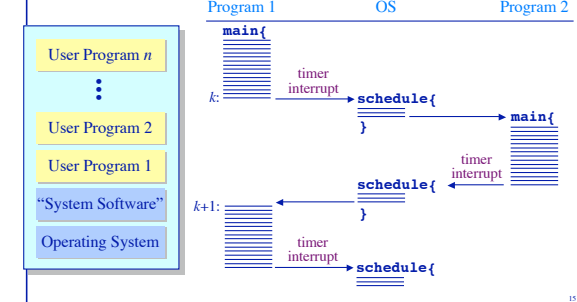
History of Operating Systems: Phases

- ◆ 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: lots of systems per user
- ◆ Phase 4: Richer services
 - Real-time operating systems

14

Timesharing ('70-)

- ◆ A timer interrupt is used to multiplex CPU among jobs



15

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: lots of systems per user
- ◆ Phase 4: Richer services
 - Real-time operating systems

16

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

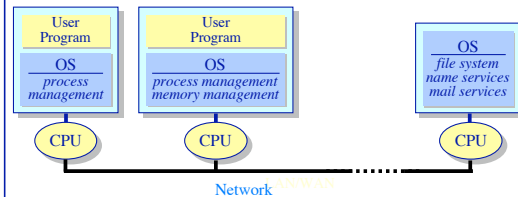


- ◆ Evolution
 - Initially: OS as a simple service provider (simple libraries)
 - Now: Multi-application systems with support for coordination

17

Distributed Operating Systems

- ◆ Typically support distributed services
 - Sharing of data and coordination across multiple systems
- ◆ Possibly employ multiple processors
 - Loosely coupled v. tightly coupled systems
- ◆ High availability & reliability requirements



18

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: lots of systems per user
- ◆ Phase 4: Richer services
 - Real-time operating systems

19

Richer Operating Systems

Real-time operating systems

- ◆ A system with a dual notion of correctness
 - Logical correctness — "it does the right thing"
 - Temporal correctness — "it does it on time"
- ◆ A system wherein *predictability* is more important than *performance*

Example: Digital video payout

```

/* Main processing loop */
loop
    data = read( network)
    video_frame = decompress(data)
    write( frame_buffer, video_frame)
end loop
    
```



Timing constraint: Execute loop once every 33 ms.

20

Real-time Operating Systems: Issues

Digital video processing loop:

```
/* Main processing loop */
loop
  data      = read( network)
  video_frame = decompress(data)
  display( video_frame)
end loop
```



Timing constraint: Execute loop once every 33 ms.

- ◆ Dedicated system — real-time performance *iff*
 $time(loop) \leq 33 \text{ ms}$
 - Real-time computing is a *programming* problem
(“Just buy a faster processor”)
- ◆ Multi-programmed system — real-time performance *iff* ... ???
 - Real-time computing is an *operating systems* problem

21

Course Overview

- ◆ OS Structure, Processes and Process Management
- ◆ Threads and concurrent programming
 - Thread coordination, mutual exclusion, monitors
 - Deadlocks
- ◆ CPU scheduling
- ◆ Memory management
- ◆ Secondary storage management & file systems
- ◆ Distributed systems & networking

22