CS 134:
Operating Systems

Definitions, Abstractions, Taxonomies, Early History

CS 134:
Operating Systems
Definitions, Abstractions, Taxonomies, Early History

Overview

What Is an OS?

History

Hardware

Overview

What Is an OS?

History

Hardware

What *is* an Operating System Anyway?

Class Exercise: Devise three separate definitions. Discuss.

What *is* an Operating System Anyway?
Class Exercise: Devise three separate definitions. Discuss.

Several slides follow that aren't on handout.

It’s A Programmer’s Toolkit

Provide useful functionality to programs:

- Prevent duplicated work
- Promote reuse

It’s A Programmer’s Toolkit
Provide useful functionality to programs:
- Prevent duplicated work
- Promote reuse

It’s a Control Program

Provide the rules for the how the machine will operate:

- Control the operation of the I/O devices
- Ensure smooth running of the machine

It’s a Control Program

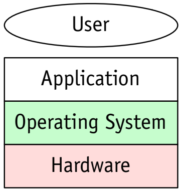
Provide the rules for the how the machine will operate:

- Control the operation of the I/O devices
- Ensure smooth running of the machine

It’s an Abstraction Layer

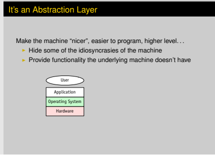
Make the machine “nicer”, easier to program, higher level. . .

- Hide some of the idiosyncrasies of the machine
- Provide functionality the underlying machine doesn’t have



2012-12-06

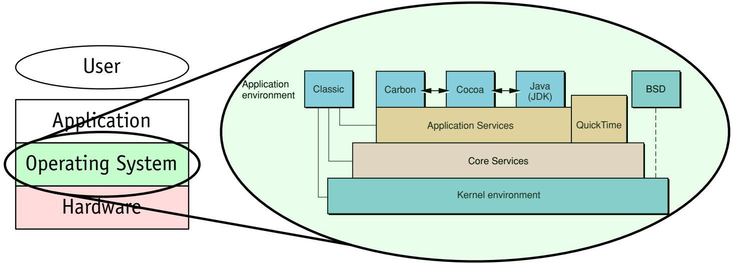
CS34
└What Is an OS?
└It’s an Abstraction Layer



It's an Abstraction Layer

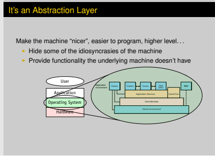
Make the machine "nicer", easier to program, higher level…

- Hide some of the idiosyncrasies of the machine
- Provide functionality the underlying machine doesn't have



2012-12-06

CS34
└What Is an OS?
└It's an Abstraction Layer



It's a Virtual Machine

OS provides an *environment*
This environment can be seen as a "new machine"...

Hardware	—Physical machine
+ Core OS	—Virtual machine
+ OS Libraries	—Virtual machine
+ OS Utilities	—Virtual machine
+ Application	—Virtual machine

It's a Virtual Machine
OS provides an *environment*
This environment can be seen as a "new machine"...
Hardware —Physical machine
+ Core OS —Virtual machine
+ OS Libraries —Virtual machine
+ OS Utilities —Virtual machine
+ Application —Virtual machine

It’s a Protection Layer

Make the machine more robust—less scope for a bug to have devastating consequences

- OS does everything programs can’t be trusted to do
- OS makes programs play nice with others

It’s a Protection Layer

Make the machine more robust—less scope for a bug to have devastating consequences

- OS does everything programs can’t be trusted to do
- OS makes programs play nice with others

It's a Policy Enforcer

OS provides the mechanisms to enforce various policies

It's a Policy Enforcer

OS provides the mechanisms to enforce various policies

It’s a Policy Enforcer

OS provides the mechanisms to enforce various policies

Class Exercise: Examples?

It’s a Policy Enforcer
OS provides the mechanisms to enforce various policies
Class Exercise: Examples?

It’s a Resource Manager

The operating system manages physical resources:

- Processor
- Memory
- Storage devices
- Network devices

  etc. . .

It’s a Resource Manager
The operating system manages physical resources:
Processor
Memory
Storage devices
Network devices
etc. . .

It’s a Resource Manager (cont’d.)

The operating system manages virtual resources:

- Processes
- Files
- Users
- Network connections
- Windows

etc. . . .

It’s a Resource Manager (cont’d.)
The operating system manages virtual resources:
Processes
Files
Users
Network connections
Windows
etc. . . .

It’s a Product

Many operating systems are sold by commercial companies

- Market vs. technical considerations
- The operating system is what comes in the box marked “operating system”

It’s a Product
Many operating systems are sold by commercial companies
- Market vs. technical considerations
- The operating system is what comes in the box marked “operating system”

Fundamental Abstractions

What are the (user-level) abstractions we'd expect to find in a modern OS?

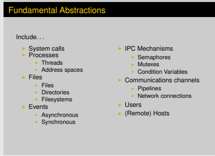
Fundamental Abstractions

What are the (user-level) abstractions we'd expect to find in a modern OS?

Fundamental Abstractions

Include...

- System calls
- Processes
  - Threads
  - Address spaces
- Files
  - Files
  - Directories
  - Filesystems
- Events
  - Asynchronous
  - Synchronous
- IPC Mechanisms
  - Semaphores
  - Mutexes
  - Condition Variables
- Communications channels
  - Pipelines
  - Network connections
- Users
- (Remote) Hosts



It’s a Resource Manager

What are the “resources” that an operating system manages?

It’s a Resource Manager
What are the “resources” that an operating system manages?

It's a Resource Manager

The operating system manages physical resources:

- Processor
- Memory
- Storage devices
- Network devices

  etc.. . .

It's a Resource Manager
The operating system manages physical resources:
Processor
Memory
Storage devices
Network devices
etc.. . .

It’s a Resource Manager (cont’d.)

The operating system manages virtual resources:

- Processes
- Files
- Users
- Network connections
- Windows

etc. . . .

2012-12-06

CS34
└What Is an OS?
└It’s a Resource Manager (cont’d.)

It’s a Resource Manager (cont’d.)
The operating system manages virtual resources:
- Processes
- Files
- Users
- Network connections
- Windows
etc. . . .

Taxonomy of Computer Systems

Different computer systems ask different things from their OS

Taxonomy of Computer Systems

Different computer systems ask different things from their OS

Taxonomy of Computer Systems

Different computer systems ask different things from their OS

Class Exercise: Give some dimensions across which computer systems vary

Taxonomy of Computer Systems

Different computer systems ask different things from their OS

Class Exercise: Give some dimensions across which computer systems vary

Partial Taxonomy of Computer Systems

Different computer systems ask different things from their OS:

Special-purpose	↔	*General-purpose*
Single-user	↔	*Multi-user*
Non–Resource-sharing	↔	*Resource sharing*
Single processor	↔	*Multiprocessor*
Stand alone	↔	*Networked*
Centralized	↔	*Distributed*
Batch	↔	*Interactive*
Deadline-free	↔	*Real-time*
Insecure	↔	*Secure*
Symmetric	↔	*Asymmetric*
Simple	↔	*Complex*
Small	↔	*Large*
Inexpensive	↔	*Expensive*
	etc.	

Partial Taxonomy of Computer Systems
Different computer systems ask different things from their OS:
Special-purpose ↔ General-purpose
Single-user ↔ Multi-user
Non–Resource-sharing ↔ Resource sharing
Single processor ↔ Multiprocessor
Stand alone ↔ Networked
Centralized ↔ Distributed
Batch ↔ Interactive
Deadline-free ↔ Real-time
Insecure ↔ Secure
Symmetric ↔ Asymmetric
Simple ↔ Complex
Small ↔ Large
Inexpensive ↔ Expensive
etc.

Early Computers

1950s—large complex machines

- Operated directly from a console
- Used interactively by a single user
- Ran one program at a time (uniprogramming)
- Read data from paper tape, punched cards, or toggle switches

OS? Maybe a library containing code to work the I/O devices was useful.

Early Computers

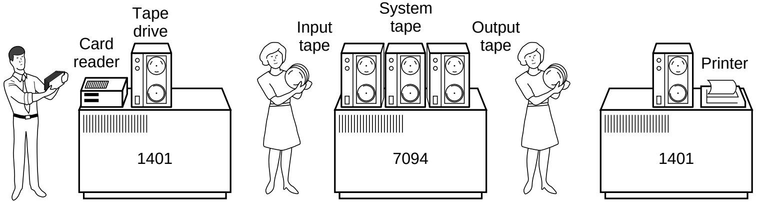
1950s—large complex machines
- Operated directly from a console
- Used interactively by a single user
- Ran one program at a time (uniprogramming)
- Read data from paper tape, punched cards, or toggle switches

OS? Maybe a library containing code to work the I/O devices was useful.

Simple Batch Systems

Provide better use of resources:

- Users access computer indirectly
- Programs and input *(jobs)* taken from a *batch queue*
- Computer has a human operator to feed it jobs

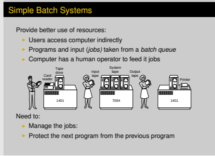


Need to:

- Manage the jobs:
- Protect the next program from the previous program

2012-12-06

CS34
└History
└Simple Batch Systems



SPOOLing Batch Systems

Provide better use of resources—buffer input and output

- Read-ahead input from disk/tape
- Write-behind output to disk/tape

SPOOLing Batch Systems

Provide better use of resources—buffer input and output
- Read-ahead input from disk/tape
- Write-behind output to disk/tape

SPOOLing Batch Systems

Provide better use of resources—buffer input and output

- Read-ahead input from disk/tape
- Write-behind output to disk/tape

Class Exercise: Why does buffering improve performance?
Does buffering *always* improve performance?
(What assumptions are we making?)

SPOOLing Batch Systems
Provide better use of resources—buffer input and output
- Read-ahead input from disk/tape
- Write-behind output to disk/tape
Class Exercise: Why does buffering improve performance?
Does buffering *always* improve performance?
(What assumptions are we making?)

Multiprogrammed Batch Systems

Provide better use of resources—multiplex the processor:

- Run multiple independent programs at once
- Switch to another program when running program waits for I/O

More work for OS. More complex management of

- I/O
- Memory
- Processor

Multiprogrammed Batch Systems

Provide better use of resources—multiplex the processor:
- Run multiple independent programs at once
- Switch to another program when running program waits for I/O

More work for OS. More complex management of
- I/O
- Memory
- Processor

Time-Sharing Systems

Provide better environment for users—multiplex the processor between users:

- Run multiple independent programs at once
- Switch between users rapidly
  - Illusion of having the machine's full attention

Yet more complexity for OS:

Time-Sharing Systems
Provide better environment for users—multiplex the processor between users:
Run multiple independent programs at once
Switch between users rapidly
Illusion of having the machine's full attention
Yet more complexity for OS:

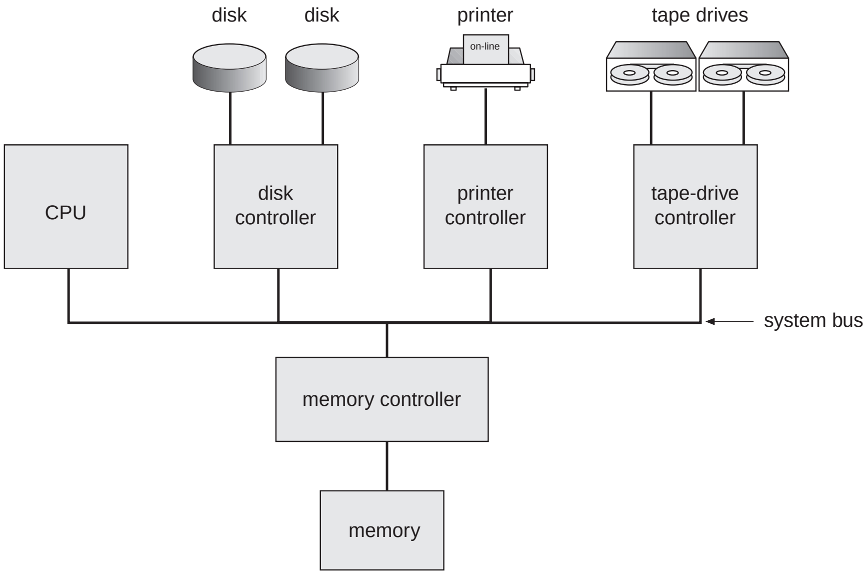
History Repeats Itself

As new, “smaller” hardware appears, it tends to repeat this evolution

- Mini computers
- Personal computers
- PDAs
- Embedded systems
  - Cell phones
  - MP3 Players
  - Cameras, etc.

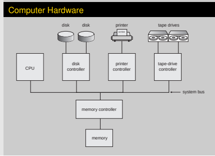
History Repeats Itself
As new, “smaller” hardware appears, it tends to repeat this evolution
Mini computers
Personal computers
PDAs
Embedded systems
Cell phones
MP3 Players
Cameras, etc.

Computer Hardware



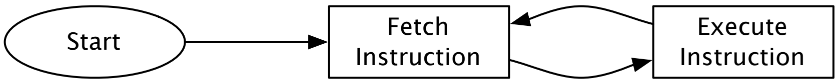
2012-12-06

CS34
Hardware
Computer Hardware



Computer Hardware—CPU & Memory

Need to perform computation!



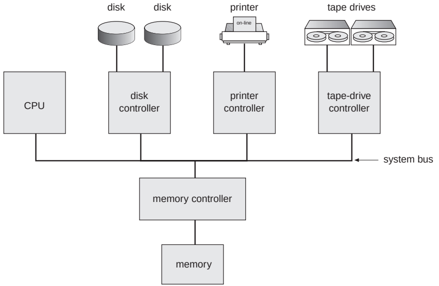
- Memory contains program instructions and program data
- Processor registers maintain processor state. Registers include:
  - General purpose (address & data) registers
  - Instruction pointer (aka program counter)
  - Stack pointer(s)
  - Control and status registers

Computer Hardware—CPU & Memory
Need to perform computation!
Start
Fetch Instruction
Execute Instruction
Memory contains program instructions and program data
Processor registers maintain processor state. Registers include:
General purpose (address & data) registers
Instruction pointer (aka program counter)
Stack pointer(s)
Control and status registers

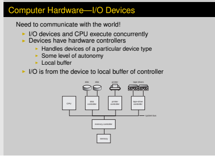
Computer Hardware—I/O Devices

Need to communicate with the world!

- I/O devices and CPU execute concurrently
- Devices have hardware controllers
  - Handles devices of a particular device type
  - Some level of autonomy
  - Local buffer
- I/O is from the device to local buffer of controller



Computer Hardware—I/O Devices



Programmed I/O

After I/O starts, control returns to user program only on I/O completion

- CPU waits until I/O completes.
- At most one I/O request is outstanding at a time
  - No simultaneous I/O processing

Programmed I/O

After I/O starts, control returns to user program only on I/O completion

- CPU waits until I/O completes.
- At most one I/O request is outstanding at a time
  - No simultaneous I/O processing

Polled I/O

Polling == Querying the I/O device
Separate I/O into two parts:

- Initiation
- Polling

Advantages?

Polled I/O
Polling == Querying the I/O device
Separate I/O into two parts:
- Initiation
- Polling
Advantages?

Interrupt-Driven I/O

Separate I/O into two parts:

- Initiation
- Asynchronous notification

Interrupt-Driven I/O
Separate I/O into two parts:
- Initiation
- Asynchronous notification

I/O in User-Level Code

User-level code almost always uses "programmed I/O"
(e.g. `read` and `write` on a file)

Why?

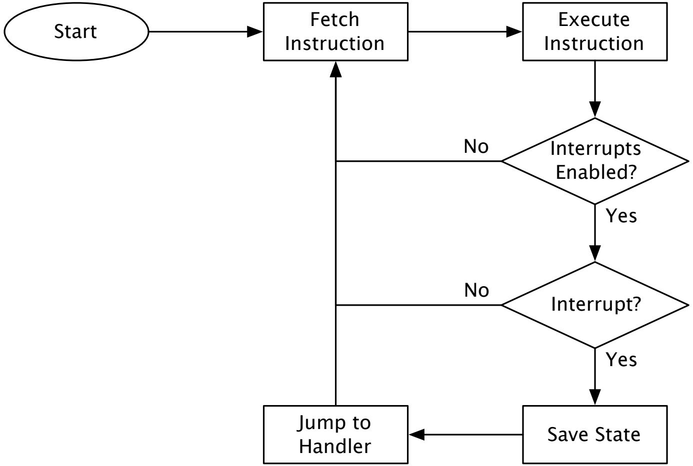
I/O in User-Level Code

User-level code almost always uses "programmed I/O"
(e.g. read and write on a file)

Why?

Computer Hardware—CPU with Interrupts

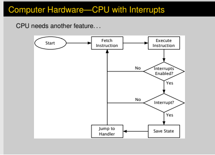
CPU needs another feature. . .



2012-12-06

CS34
└Hardware

└Computer Hardware—CPU with Interrupts



Handling an Interrupt

What needs to happen:

- Save state
  - All registers
  - Switch stacks?
- Find out what interrupt was...
  - Polling
  - Vectored interrupts

Handling an Interrupt
What needs to happen:
Save state
All registers
Switch stacks?
Find out what interrupt was...
Polling
Vectored interrupts

Types of Interrupts

Various types

- Software exception (also called a trap)
- Timer
- I/O
- Hardware failure

A modern operating system is *interrupt driven*

2012-12-06

CS34
└─Hardware
└─Types of Interrupts

Types of Interrupts

Various types
- Software exception (also called a trap)
- Timer
- I/O
- Hardware failure

A modern operating system is *interrupt driven*

Other Hardware Features

We've covered interrupts, but hardware has other cool features, including:

- Caches
- Memory management
- Protection

We'll come back to hardware as we address these topics.

Other Hardware Features

We've covered interrupts, but hardware has other cool features, including:

- Caches
- Memory management
- Protection

We'll come back to hardware as we address these topics.