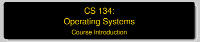
CS 134: Operating Systems

Course Introduction

CS34

2012-12-06



CS 134:
Operating Systems
Course Introduction

Overview

Class Introduction
- Administrivia
- Course Purpose

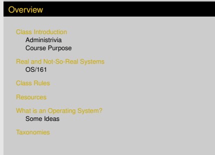
Real and Not-So-Real Systems
- OS/161

Class Rules

Resources

What is an Operating System?
- Some Ideas

Taxonomies



Overview

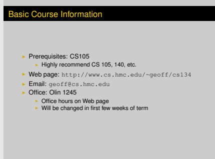
Class Introduction
Administrivia
Course Purpose
Real and Not-So-Real Systems
OS/161
Class Rules
Resources
What is an Operating System?
Some Ideas
Taxonomies

Basic Course Information

- Prerequisites: CS105
  - Highly recommend CS 105, 140, etc.
- Web page: `http://www.cs.hmc.edu/~geoff/cs134`
- Email: `geoff@cs.hmc.edu`
- Office: Olin 1245
  - Office hours on Web page
  - Will be changed in first few weeks of term

2012-12-06

CS34
└Class Introduction
 └Administrivia
  └Basic Course Information



Basic Course Information

- Prerequisites: CS105
  - Highly recommend CS 105, 140, etc.
- Web page: `http://www.cs.hmc.edu/~geoff/cs134`
- Email: `geoff@cs.hmc.edu`
- Office: Olin 1245
  - Office hours on Web page
  - Will be changed in first few weeks of term

Class Exercise

You’ve signed up for this course, but are you clear what it is about?

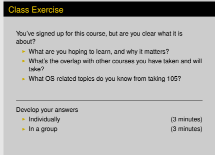
- What are you hoping to learn, and why it matters?
- What’s the overlap with other courses you have taken and will take?
- What OS-related topics do you know from taking 105?

Develop your answers

Individually	(3 minutes)
In a group	(3 minutes)

2012-12-06

```
CS34
└Class Introduction
  └Course Purpose
    └Class Exercise
```



Class Exercise

You’ve signed up for this course, but are you clear what it is about?

- What are you hoping to learn, and why it matters?
- What’s the overlap with other courses you have taken and will take?
- What OS-related topics do you know from taking 105?

Develop your answers

- Individually (3 minutes)
- In a group (3 minutes)

What Are You Looking Forward To?

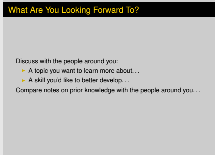
Discuss with the people around you:

- A topic you want to learn more about. . .
- A skill you'd like to better develop. . .

Compare notes on prior knowledge with the people around you. . .

2012-12-06

CS34
└Class Introduction
  └Course Purpose
    └What Are You Looking Forward To?



What Are You Looking Forward To?

Discuss with the people around you:

- A topic you want to learn more about. . .
- A skill you'd like to better develop. . .

Compare notes on prior knowledge with the people around you. . .

Class Exercise

What are you going to do to get the most out of this class?

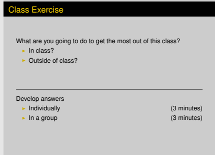
- In class?
- Outside of class?

Develop answers

▸ Individually	(3 minutes)
▸ In a group	(3 minutes)

2012-12-06

CS34
- Class Introduction
  - Course Purpose
    - Class Exercise



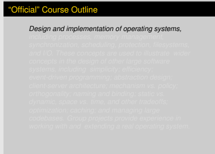
Class Exercise

What are you going to do to get the most out of this class?
- In class?
- Outside of class?

Develop answers
- Individually (3 minutes)
- In a group (3 minutes)

“Official” Course Outline

Design and implementation of operating systems, including processes, memory management, synchronization, scheduling, protection, filesystems, and I/O. These concepts are used to illustrate wider concepts in the design of other large software systems, including simplicity; efficiency; event-driven programming; abstraction design; client-server architecture; mechanism vs. policy; orthogonality; naming and binding; static vs. dynamic, space vs. time, and other tradeoffs; optimization; caching; and managing large codebases. Group projects provide experience in working with and extending a real operating system.



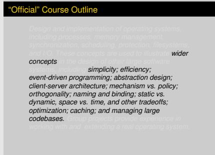
“Official” Course Outline

Design and implementation of operating systems,

This slide has animations.

“Official” Course Outline

Design and implementation of operating systems, including processes, memory management, synchronization, scheduling, protection, filesystems, and I/O. These concepts are used to illustrate wider concepts in the design of other large software systems, including simplicity; efficiency; event-driven programming; abstraction design; client-server architecture; mechanism vs. policy; orthogonality; naming and binding; static vs. dynamic, space vs. time, and other tradeoffs; optimization; caching; and managing large codebases. Group projects provide experience in working with and extending a real operating system.



“Official” Course Outline

Design and implementation of operating systems, including processes, memory management, synchronization, scheduling, protection, filesystems, and I/O. These concepts are used to illustrate wider concepts in the design of other large software systems, including simplicity; efficiency; event-driven programming; abstraction design; client-server architecture; mechanism vs. policy; orthogonality; naming and binding; static vs. dynamic, space vs. time, and other tradeoffs; optimization; caching; and managing large codebases. Group projects provide experience in working with and extending a real operating system.

This slide has animations.

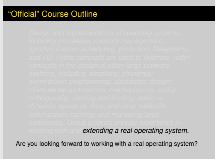
“Official” Course Outline

Design and implementation of operating systems, including processes, memory management, synchronization, scheduling, protection, filesystems, and I/O. These concepts are used to illustrate wider concepts in the design of other large software systems, including simplicity; efficiency; event-driven programming; abstraction design; client-server architecture; mechanism vs. policy; orthogonality; naming and binding; static vs. dynamic, space vs. time, and other tradeoffs; optimization; caching; and managing large codebases. Group projects provide experience in working with and extending a real operating system.

Are you looking forward to working with a real operating system?

2012-12-06

CS34
└Class Introduction
 └Course Purpose
  └“Official” Course Outline



“Official” Course Outline

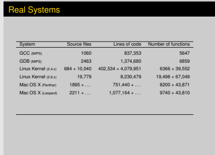
extending a real operating system.

Are you looking forward to working with a real operating system?

This slide has animations.

Real Systems

System	Source files	Lines of code	Number of functions
GCC (MIPS)	1060	837,353	5647
GDB (MIPS)	2463	1,374,680	6859
Linux Kernel (2.4.x)	684 + 10,040	402,534 + 4,079,951	6366 + 39,552
Linux Kernel (2.6.x)	19,779	8,230,479	19,498 + 67,049
Mac OS X (Panther)	1895 + . . .	751,440 + . . .	8200 + 43,871
Mac OS X (Leopard)	2211 + . . .	1,077,164 + . . .	9740 + 43,810



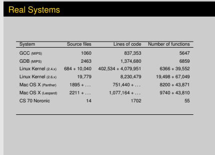
Real Systems

System	Source files	Lines of code	Number of functions
GCC (MIPS)	1060	837,353	5647
GDB (MIPS)	2463	1,374,680	6859
Linux Kernel (2.4.x)	684 + 10,040	402,534 + 4,079,951	6366 + 39,552
Linux Kernel (2.6.x)	19,779	8,230,479	19,498 + 67,049
Mac OS X (Panther)	1895 + . . .	751,440 + . . .	8200 + 43,871
Mac OS X (Leopard)	2211 + . . .	1,077,164 + . . .	9740 + 43,810

This slide has animations.

Real Systems

System	Source files	Lines of code	Number of functions
GCC (MIPS)	1060	837,353	5647
GDB (MIPS)	2463	1,374,680	6859
Linux Kernel (2.4.x)	684 + 10,040	402,534 + 4,079,951	6366 + 39,552
Linux Kernel (2.6.x)	19,779	8,230,479	19,498 + 67,049
Mac OS X (Panther)	1895 + . . .	751,440 + . . .	8200 + 43,871
Mac OS X (Leopard)	2211 + . . .	1,077,164 + . . .	9740 + 43,810
CS 70 Noronic	14	1702	55



Real Systems

System	Source files	Lines of code	Number of functions
GCC (MIPS)	1060	837,353	5647
GDB (MIPS)	2463	1,374,680	6859
Linux Kernel (2.4.x)	684 + 10,040	402,534 + 4,079,951	6366 + 39,552
Linux Kernel (2.6.x)	19,779	8,230,479	19,498 + 67,049
Mac OS X (Panther)	1895 + . . .	751,440 + . . .	8200 + 43,871
Mac OS X (Leopard)	2211 + . . .	1,077,164 + . . .	9740 + 43,810
CS 70 Noronic	14	1702	55

This slide has animations.

What is OS/161?

System/161 simulates a real machine

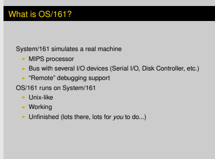
- MIPS processor
- Bus with several I/O devices (Serial I/O, Disk Controller, etc.)
- “Remote” debugging support

OS/161 runs on System/161

- Unix-like
- Working
- Unfinished (lots there, lots for *you* to do...)

2012-12-06

CS34
└Real and Not-So-Real Systems
 └OS/161
  └What is OS/161?



What is OS/161?

System/161 simulates a real machine

- MIPS processor
- Bus with several I/O devices (Serial I/O, Disk Controller, etc.)
- “Remote” debugging support

OS/161 runs on System/161

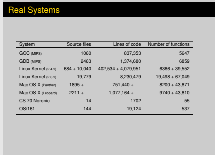
- Unix-like
- Working
- Unfinished (lots there, lots for *you* to do...)

Real Systems

System	Source files	Lines of code	Number of functions
GCC (MIPS)	1060	837,353	5647
GDB (MIPS)	2463	1,374,680	6859
Linux Kernel (2.4.x)	684 + 10,040	402,534 + 4,079,951	6366 + 39,552
Linux Kernel (2.6.x)	19,779	8,230,479	19,498 + 67,049
Mac OS X (Panther)	1895 + . . .	751,440 + . . .	8200 + 43,871
Mac OS X (Leopard)	2211 + . . .	1,077,164 + . . .	9740 + 43,810
CS 70 Noronic	14	1702	55
OS/161	144	19,124	537

2012-12-06

CS34
└Real and Not-So-Real Systems
  └OS/161
    └Real Systems


Real Systems

System	Source files	Lines of code	Number of functions
GCC (MIPS)	1060	837,353	5647
GDB (MIPS)	2463	1,374,680	6859
Linux Kernel (2.4.x)	684 + 10,040	402,534 + 4,079,951	6366 + 39,552
Linux Kernel (2.6.x)	19,779	8,230,479	19,498 + 67,049
Mac OS X (Panther)	1895 + . . .	751,440 + . . .	8200 + 43,871
Mac OS X (Leopard)	2211 + . . .	1,077,164 + . . .	9740 + 43,810
CS 70 Noronic	14	1702	55
OS/161	144	19,124	537

This slide has animations.

OS/161 Quickstart

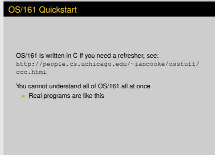
OS/161 is written in C If you need a refresher, see: `http://people.cs.uchicago.edu/~iancooke/osstuff/ccc.html`

You cannot understand all of OS/161 all at once

- Real programs are like this

2012-12-06

CS34
└ Real and Not-So-Real Systems
  └ OS/161
    └ OS/161 Quickstart



OS/161 Quickstart

OS/161 is written in C If you need a refresher, see: http://people.cs.uchicago.edu/~iancooke/osstuff/ccc.html

You cannot understand all of OS/161 all at once

- Real programs are like this

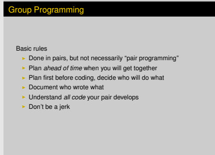
Group Programming

Basic rules

- Done in pairs, but not necessarily “pair programming”
- Plan *ahead of time* when you will get together
- Plan first before coding, decide who will do what
- Document who wrote what
- Understand *all code* your pair develops
- Don’t be a jerk

2012-12-06

CS34
└Class Rules
└Group Programming



Group Programming

Basic rules

- Done in pairs, but not necessarily “pair programming”
- Plan *ahead of time* when you will get together
- Plan first before coding, decide who will do what
- Document who wrote what
- Understand *all code* your pair develops
- Don’t be a jerk

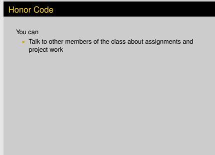
Honor Code

You can

- Talk to other members of the class about assignments and project work

2012-12-06

CS34
└─Class Rules
└─Honor Code



Honor Code

You can

- Talk to other members of the class about assignments and project work

This slide has animations.

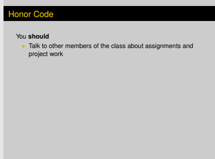
Honor Code

You **should**

- Talk to other members of the class about assignments and project work

2012-12-06

CS34
└Class Rules
  └Honor Code


Honor Code

You **should**

- Talk to other members of the class about assignments and project work

This slide has animations.

Honor Code

You **should**

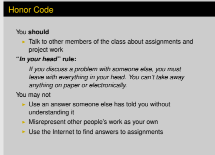
- Talk to other members of the class about assignments and project work

"*In your head*" rule:

> *If you discuss a problem with someone else, you must leave with everything in your head. You can't take away anything on paper or electronically.*

You may not

- Use an answer someone else has told you without understanding it
- Misrepresent other people's work as your own
- Use the Internet to find answers to assignments



Honor Code

You **should**

- Talk to other members of the class about assignments and project work

"*In your head*" rule:

If you discuss a problem with someone else, you must leave with everything in your head. You can't take away anything on paper or electronically.

You may not

- Use an answer someone else has told you without understanding it
- Misrepresent other people's work as your own
- Use the Internet to find answers to assignments

This slide has animations.

Peer Review & Grading

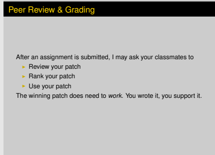
After an assignment is submitted, I may ask your classmates to

- Review your patch
- Rank your patch
- Use your patch

The winning patch does need to *work*. You wrote it, you support it.

2012-12-06

CS34
└Class Rules
└Peer Review & Grading



Peer Review & Grading

After an assignment is submitted, I may ask your classmates to

- Review your patch
- Rank your patch
- Use your patch

The winning patch does need to *work*. You wrote it, you support it.

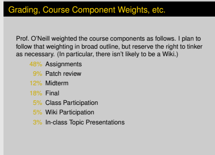
Grading, Course Component Weights, etc.

Prof. O’Neill weighted the course components as follows. I plan to follow that weighting in broad outline, but reserve the right to tinker as necessary. (In particular, there isn’t likely to be a Wiki.)

48%	Assignments
9%	Patch review
12%	Midterm
18%	Final
5%	Class Participation
5%	Wiki Participation
3%	In-class Topic Presentations

2012-12-06

CS34
└Class Rules
└Grading, Course Component Weights, etc.



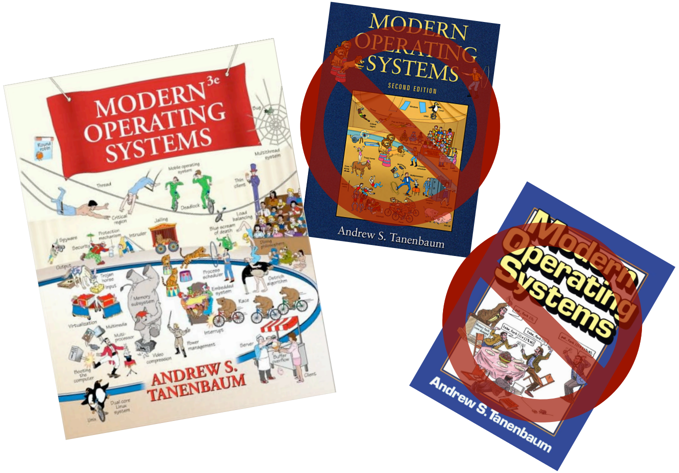
Grading, Course Component Weights, etc.

Prof. O’Neill weighted the course components as follows. I plan to follow that weighting in broad outline, but reserve the right to tinker as necessary. (In particular, there isn’t likely to be a Wiki.)

48% Assignments
9% Patch review
12% Midterm
18% Final
5% Class Participation
5% Wiki Participation
3% In-class Topic Presentations

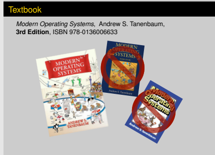
Textbook

Modern Operating Systems, Andrew S. Tanenbaum,
3rd Edition, ISBN 978-0136006633



2012-12-06

CS34
└Resources
└Textbook


Textbook

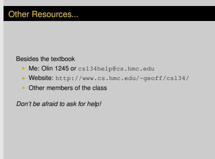
Modern Operating Systems, Andrew S. Tanenbaum,
3rd Edition, ISBN 978-0136006633

Other Resources...

Besides the textbook

- Me: Olin 1245 or `cs134help@cs.hmc.edu`
- Website: `http://www.cs.hmc.edu/~geoff/cs134/`
- Other members of the class

Don't be afraid to ask for help!



Other Resources...

Besides the textbook

- Me: Olin 1245 or `cs134help@cs.hmc.edu`
- Website: `http://www.cs.hmc.edu/~geoff/cs134/`
- Other members of the class

Don't be afraid to ask for help!

Mailing Lists

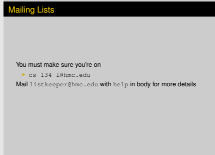
You must make sure you’re on

- `cs-134-l@hmc.edu`

Mail `listkeeper@hmc.edu` with `help` in body for more details

2012-12-06

CS34
└Resources
└Mailing Lists



Mailing Lists

You must make sure you’re on

- `cs-134-l@hmc.edu`

Mail `listkeeper@hmc.edu` with `help` in body for more details

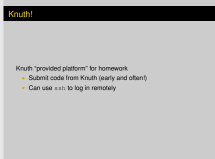
Knuth!

Knuth “provided platform” for homework

- Submit code from Knuth (early and often!)
- Can use `ssh` to log in remotely

2012-12-06

CS34
└─Resources
└─Knuth!



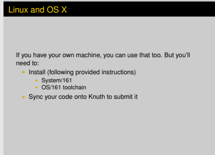
Knuth!

Knuth “provided platform” for homework
- Submit code from Knuth (early and often!)
- Can use `ssh` to log in remotely

Linux and OS X

If you have your own machine, you can use that too. But you'll need to:

- Install (following provided instructions)
  - System/161
  - OS/161 toolchain
- Sync your code onto Knuth to submit it



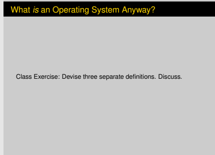
Linux and OS X

If you have your own machine, you can use that too. But you'll need to:

- Install (following provided instructions)
  - System/161
  - OS/161 toolchain
- Sync your code onto Knuth to submit it

What *is* an Operating System Anyway?

Class Exercise: Devise three separate definitions. Discuss.



What *is* an Operating System Anyway?

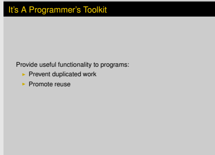
Class Exercise: Devise three separate definitions. Discuss.

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

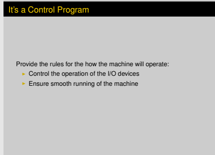
Provide useful functionality to programs:

- 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

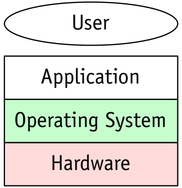
Provide the rules for the how the machine will operate:

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

It’s an Abstraction Layer

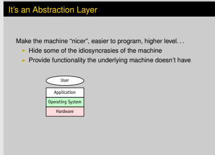
Make the machine “nicer”, easier to program, higher level. . .

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



2012-12-06

CS34
└What is an Operating System?
└Some Ideas
└It’s an Abstraction Layer


It’s an Abstraction Layer

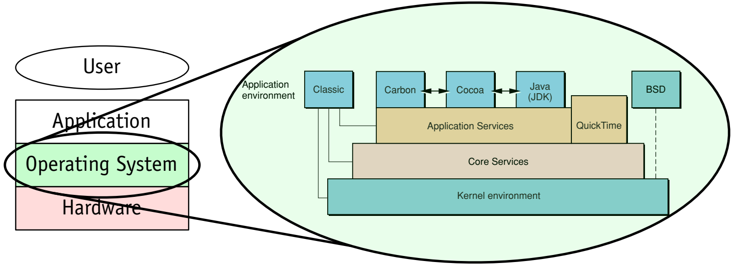
Make the machine “nicer”, easier to program, higher level. . .

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

It's an Abstraction Layer

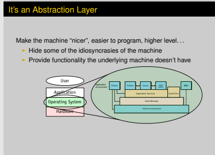
Make the machine "nicer", easier to program, higher level…

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



2012-12-06

CS34
└What is an Operating System?
  └Some Ideas
    └It's an Abstraction Layer


It's an Abstraction Layer

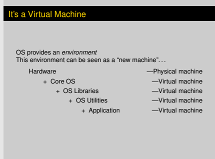
Make the machine "nicer", easier to program, higher level…

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

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

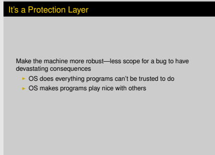
OS provides an *environment*
This environment can be seen as a “new machine”. . .

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



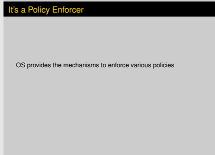
It’s a Protection Layer

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

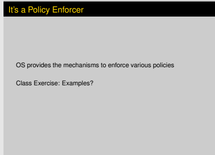
It’s a Policy Enforcer

OS provides the mechanisms to enforce various policies

Class Exercise: Examples?

2012-12-06

CS34
└─What is an Operating System?
  └─Some Ideas
    └─It’s a Policy Enforcer



It’s a Policy Enforcer

OS provides the mechanisms to enforce various policies

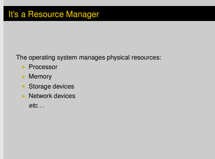
Class Exercise: Examples?

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.)

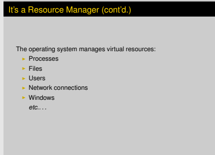
The operating system manages virtual resources:

- Processes
- Files
- Users
- Network connections
- Windows

etc.. . .

2012-12-06

CS34
What is an Operating System?
Some Ideas
It’s a Resource Manager (cont’d.)



It’s a Resource Manager (cont’d.)

The operating system manages virtual resources:

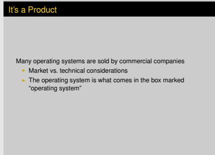
- Processes
- Files
- Users
- Network connections
- Windows

etc.. . .

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

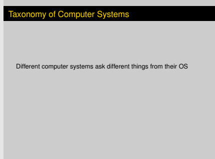
- Market vs. technical considerations
- The operating system is what comes in the box marked “operating system”

Taxonomy of Computer Systems

Different computer systems ask different things from their OS

2012-12-06

CS34
└Taxonomies
└Taxonomy of Computer Systems



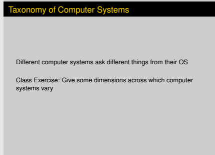
Taxonomy of Computer Systems

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

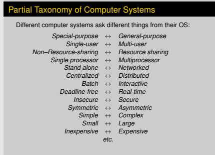
Different computer systems ask different things from their OS:

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.

2012-12-06

CS34
└Taxonomies
  └Partial Taxonomy of Computer Systems



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.