

ETH zürich spci.inf.ethz.ch
@spci_eth

ADRIAN PERRIG & TORSTEN HOEFLER

Networks and Operating Systems (252-0062-00)

Chapter 1: Introduction to Operating Systems

If Operating Systems were Airways (~year 2000)

UNIX Airways Everyone brings one piece of the plane along when they come to the airport. They all go out on the runway and put the plane together piece by piece, arguing non-stop about what kind of plane they are supposed to be building.

Air DOS Everybody pushes the airplane until it glides, then they jump on and let the plane coast until it hits the ground again. Then they push again, jump on again, and so on ...

Mac Airlines All the stewards, captains, baggage handlers, and ticket agents look and act exactly the same. Every time you ask questions about details, you are gently but firmly told that you don't need to know, don't want to know, and everything will be done for you without your ever having to know, so just shut up.

Windows Air The terminal is pretty and colorful, with friendly stewards, easy baggage check and boarding, and a smooth take-off. After about 10 minutes in the air, the plane explodes with no warning whatsoever.

Windows NT Air Just like Windows Air, but costs more, uses much bigger planes, and takes out all the other aircraft within a 40-mile radius when it explodes.




Linux Air Disgruntled employees of all the other OS airlines decide to start their own airline. They build the planes, ticket counters, and pave the runways themselves. They charge a small fee to cover the cost of printing the ticket, but you can also download and print the ticket yourself. When you board the plane, you are given a seat, four bolts, a wrench and a copy of the Seat-HOWTO.html. Once settled, the fully adjustable seat is very comfortable, the plane leaves and arrives on time without a single problem, the in-flight meal is wonderful. You try to tell customers of the other airlines about the great trip, but all they can say is, "You had to do what with the seat?" (*Author unknown*)

ETH zürich spci.inf.ethz.ch
@spci_eth

Administrivia

- **Two parts:**
 - Networks – Adrian Perrig
 - Operating Systems – Torsten Hoeffler
- **Lecture:**
 - Thu 8-10am, CAB G61
 - Fri 10am-noon, CAB G11
- **Practice sessions**
 - Thu 3-6pm, ML F 40, ML H 41.1
 - Fri 1-4pm, CHN G 22, CHN D 42, CHN D 48, CAB G 57 (may merge)
- **Go to one of these sessions!**
 - And participate!
 - Well, and **participate in the lecture** as well 😊




2

ETH zürich   spcl.inf.ethz.ch
 @spcl_eth

More Administrivia

- **Course webpage (the authoritative information source)**
 - <http://spcl.inf.ethz.ch/Teaching/2015-osnet/>
 - All slides will be there before the lecture (so you can take notes)
- **Exercises are:**
 - *Theoretical*: Analysis of performance properties
 - *Practical*: Trying out stuff + Programming exercises
- **We assume you know both C and Java.**
 - Exercises start today!
- **There is a mailing list for questions to the TAs**
 - You are not subscribed but can sign up at (if you want)
 - <https://spcl.inf.ethz.ch/cgi-bin/mailman/listinfo/2015-osnet-ta>
- **Please register during the break**
 - put your name into lists at front desk of lecture hall



3

ETH zürich   spcl.inf.ethz.ch
 @spcl_eth

Exam

- **(No mid-term.)**
- **Final exam: tbd (in Exam Session)**
- **Material:**
 - Covered in the lectures, and/or
 - Learned during the lab exercises
- **We will not follow the books closely.**
 - All pieces will be in books though
- **Optional extra readings may appear on the web**



4

ETH zürich   spcl.inf.ethz.ch
@spcl_eth

Course Outline

<ul style="list-style-type: none"> 19.02.: OS Introduction 20.02.: Processes 26.02.: Scheduling 27.02.: Synchronization 05.03.: Memory Management 06.03.: Demand Paging 12.03.: NO CLASS 13.03.: File System Abstractions 19.03.: File System Implementations 20.03.: I/O Subsystem I 26.03.: I/O Subsystem II 27.03.: Virtual Machine Monitors 02.04.: Reliable Storage, Specials 	<ul style="list-style-type: none"> 16.04.: Network Intro / OSI Model 17.04.: Physical Layer 23.04.: Data Link Layer I 24.04.: Data Link Layer II 30.04.: Network Layer I 07.05.: Network Layer I 08.05.: Network Layer II 15.05.: Transport Layer 21.05.: Congestion Control 22.05.: Congestion Control 29.05.: Application Layer
---	--

5

ETH zürich   spcl.inf.ethz.ch
@spcl_eth

Birds-eye perspective

- **Networks**
 - bridge space
- **Databases**
 - bridge time
- **Networks, Operating Systems, Databases**
 - they all manage resources
 - OS, DB: all resources (storage, computation, communication)
 - Networks: focus on communication

6

ETH zürich spci.inf.ethz.ch
@spci_eth

~200 sensors

abc NEWS HOME VIDEO U.S. WORLD POLITICS ENTERTAINMENT TECH

Scientists Hack Into Cars' Computers -- Control Brakes, Engine

Aug. 21, 2010

By MARK CLAYTON

Why d

Like 177 share 16 Tweet 100 +1 1

0 Comments

CHRISTIAN SCIENCE MONITOR

It sounds like a Hollywood movie: cybercriminals in a van use a laptop to hack wirelessly into the computer-controlled systems of the car on the road ahead. In seconds the target car's engine, brakes, and door locks are under their nefarious control.

nts




And it has been broken ☹️ (not this vendor)

ETH zürich spci.inf.ethz.ch
@spci_eth

Today: We start on Operating Systems!

- **Introduction: Why?**
- **Roles of the OS**
 - Referee
 - Illusionist
 - Glue
- **Structure of an OS**




8

ETH zürich    spcl.inf.ethz.ch
[@spcl_eth](https://twitter.com/spcl_eth)

Goals

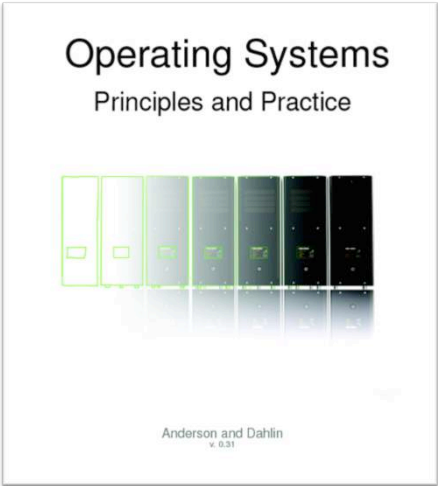
- **Demystify operating systems themselves**
 - What is an OS? What does it do?
 - What is its structure?
 - How do the OS and applications relate to each other?
 - What services does the OS provide?
- **Quintessential “systems” problem**
 - Non-idealizable / non-reducible
 - Scaling, emergent properties
 - Concurrency and asynchrony

9

ETH zürich    spcl.inf.ethz.ch
[@spcl_eth](https://twitter.com/spcl_eth)

The Book

- **On the web:**
<http://osp.p.washington.edu/>



Operating Systems
Principles and Practice

Anderson and Dahlin
v. 0.31

10

ETH zürich spci.inf.ethz.ch
@spci_eth

Also worth a look

- **Jerome H. Saltzer and M. Frans Kaashoek:**
“Principles of Computer System Design”
- **Focus on principles, with illustrative examples**




11

ETH zürich spci.inf.ethz.ch
@spci_eth

Also worth a look




- **Andrew S. Tanenbaum:**
“Modern Operating Systems”
- **Must be at least 3rd Edition!**
- **Very broad – lots of references to recent (2006) research.**

12

ETH zürich    spcl.inf.ethz.ch
[@spcl_eth](https://twitter.com/spcl_eth)

Introduction to Operating Systems

13

ETH zürich    spcl.inf.ethz.ch
[@spcl_eth](https://twitter.com/spcl_eth)

Why learn about Operating Systems?

- **One of the most complex topics in Computer Science!**
 - Very few simplifying assumptions
 - Dealing with the real world
 - Intersection of many areas
- **Mainstream OSES are large:**
 - Windows 7 ~ 40-50 million lines of code
Average modern high-end car: 100 million [1]
 - Linux rapidly catching up in complexity (~15 million LOC)
- **Most other software systems are a subset**
 - Games, browsers, databases, servers, cloud, etc.

[1]: <http://www.informationisbeautiful.net/visualizations/million-lines-of-code/>

14

ETH zürich spci.inf.ethz.ch
@spci_eth

There are lots of operating systems concepts...

- Systems calls
- Concurrency and asynchrony
- Processes and threads
- Security, authorization, protection
- Memory, virtual memory, and paging
- Files and file systems, data management
- I/O: Devices, Interrupts, DMA
- Network interfaces and protocol stacks

15

ETH zürich spci.inf.ethz.ch
@spci_eth

There are lots of operating systems...





















16

ETH zürich   spl.inf.ethz.ch
[@spl_eth](https://twitter.com/spl_eth)

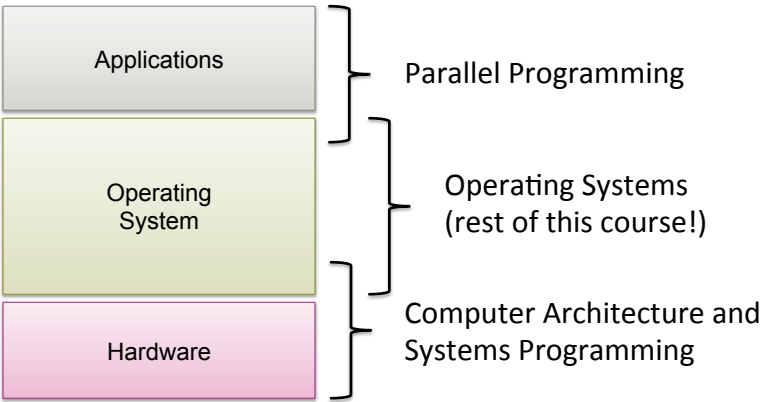
Goals: what makes a good OS?

- **Reliability: does it keep working?**
 - And availability
- **Security: can it be compromised?**
 - And isolation: is it fair?
- **Portability: how easily can it be retargeted?**
- **Performance: how fast/cheap/hungry is it?**
- **Adoption: will people use it?**
- ...

17

ETH zürich   spl.inf.ethz.ch
[@spl_eth](https://twitter.com/spl_eth)

Operating Systems



Applications

Operating System




Hardware

Parallel Programming

Operating Systems
(rest of this course!)




Computer Architecture and
Systems Programming

18

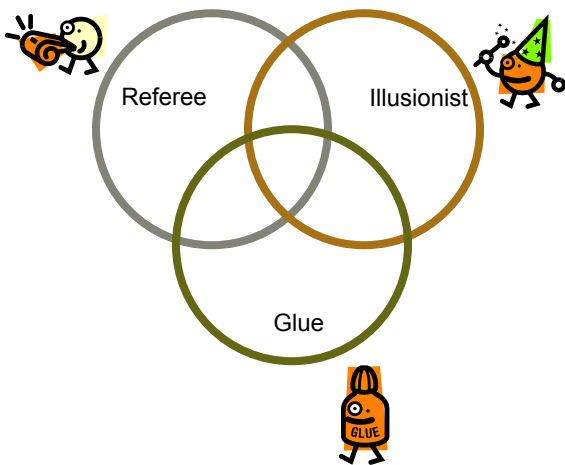
ETH zürich    spl.inf.ethz.ch
[@spl_eth](https://twitter.com/spl_eth)

Operating System Roles

19

ETH zürich    spl.inf.ethz.ch
[@spl_eth](https://twitter.com/spl_eth)

OS roles

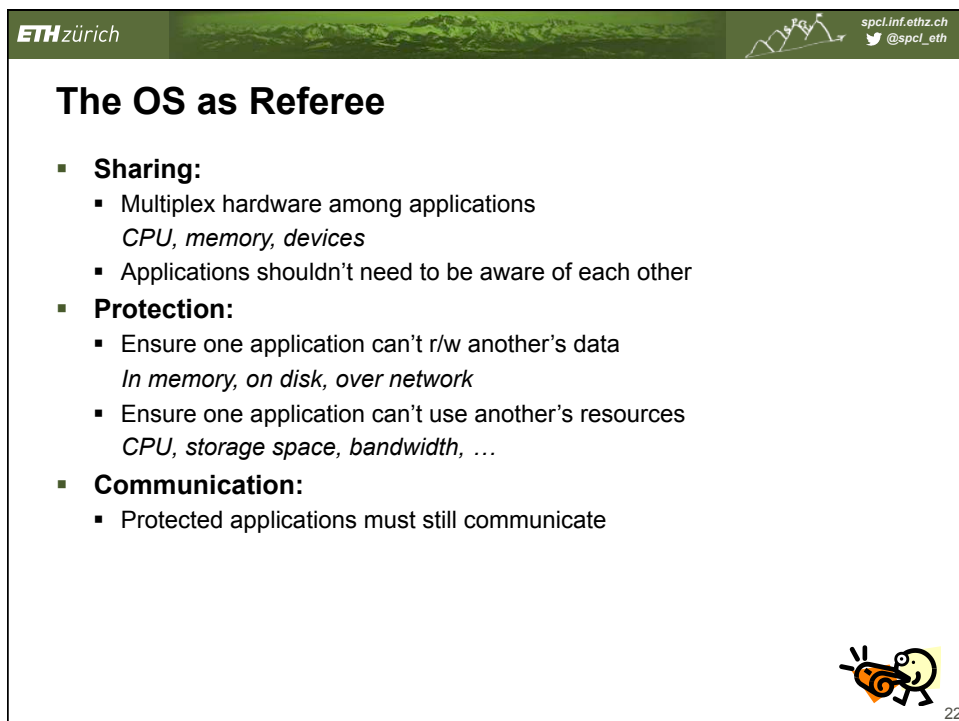
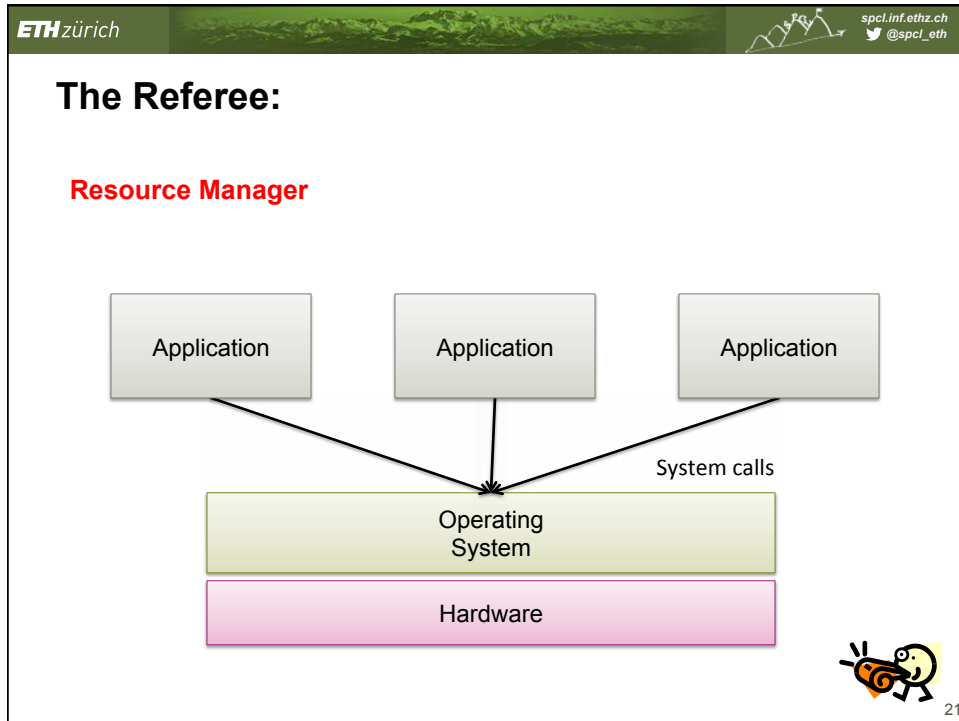




Referee

Illusionist

Glue

20



ETH zürich   spcl.inf.ethz.ch
[@spcl_eth](https://twitter.com/spcl_eth)

Resource management goals



- **Fairness:**
 - No starvation, every application makes progress
- **Efficiency:**
 - Best use of complete machine resources
 - Minimize e.g. power consumption
- **Predictability:**
 - Guarantee real-time performance

All in mutual contradiction






23



ETH zürich   spcl.inf.ethz.ch
[@spcl_eth](https://twitter.com/spcl_eth)

Example: Threads

- **Threads are virtual CPUs**
 - Physical resource: CPUs
 - Virtual resource: Threads
- Mechanism: pre-emption, timeslicing, context switching, scheduling
- **More on this later in the course...**




24

ETH zürich   spcl.inf.ethz.ch
[@spcl_eth](https://twitter.com/spcl_eth)



The Illusionist

Virtualization:

- **OS creates illusion of a “real” resource**
 - Processor, storage, network, links, ...
- **Virtual resource looks a *bit* like a physical resource**
- **However, is frequently quite different...**
 - Simpler, larger, better, ...




25



ETH zürich   spcl.inf.ethz.ch
[@spcl_eth](https://twitter.com/spcl_eth)

How?

- 1. Multiplexing**
 - Divide resources up among clients
- 2. Emulation**
 - Create the illusion of a resource using software
- 3. Aggregation**
 - Join multiple resources together to create a new one




26



ETH zürich   spcl.inf.ethz.ch
[@spcl_eth](https://twitter.com/spcl_eth)

Why?

- 1. Sharing**
 - Enable multiple clients of a single resource
- 2. Sandboxing**
 - Prevent a client from accessing other resources
- 3. Decoupling**
 - Avoid tying a client to a *particular* instance of a resource
- 4. Abstraction**
 - Make a resource easier to use




27



ETH zürich   spcl.inf.ethz.ch
[@spcl_eth](https://twitter.com/spcl_eth)

Example: Virtual Memory

- **Easier memory to manage**
 - Physical resource: RAM
 - Virtual resource: Virtual Memory
 - Method: Multiplexing
 - Mechanism: virtual address translation




28



ETH zürich   sycl.inf.ethz.ch
[@sycl_eth](https://twitter.com/sycl_eth)

Example: Paged virtual memory

- **More memory than you really have**
 - Physical resource: RAM and **disk**
 - Virtual resource: paged virtual memory
 - Method: multiplexing and **emulation**
 - Mechanism: virtual memory + paging to/from disk
- **Much more on this later in the course...**




29

ETH zürich   sycl.inf.ethz.ch
[@sycl_eth](https://twitter.com/sycl_eth)

Example: Virtual machines

- **Quite popular topic commercially right now:**
 - Xen, VMware, HyperV, kvm, etc.
- **Many uses:**
 - Run one OS on another
 - Consolidate servers
 - Migrate running machines around datacenter
 - Run hundreds of “honeypot” machines
 - Deterministic replay of whole machines
 - Etc.




30

ETH zürich spci.inf.ethz.ch
@spci_eth

Example: Files (or database!)

- **Virtual resource: persistent memory**
- **Physical resource: disk**
- **Method: multiplexing, emulation**
- **Mechanism: block allocation, metadata**

▪ **Again, more later...**

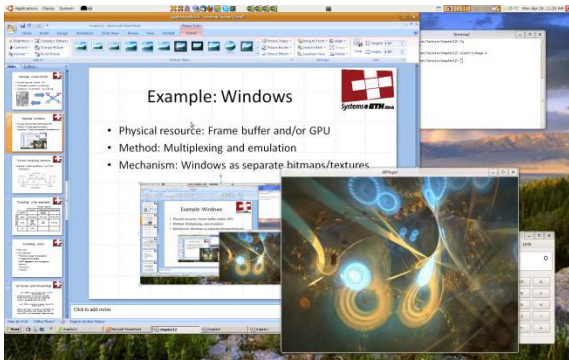



31

ETH zürich spci.inf.ethz.ch
@spci_eth

Example: Windows (not the Microsoft OS)

- **Physical resource: Frame buffer and/or GPU**
- **Method: Multiplexing and emulation**
- **Mechanism: Windows as separate bitmaps/textures**





32

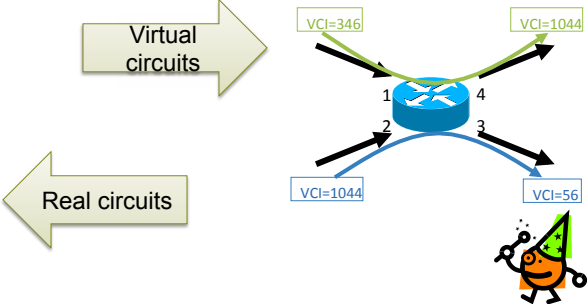
ETH zürich spcl.inf.ethz.ch
@spcl_eth

Example: Virtual circuits

- **Physical resource:** network link
- **Virtualization method:** multiplexing
- **Mechanism:** VC identifiers, VC switching



Virtual circuits



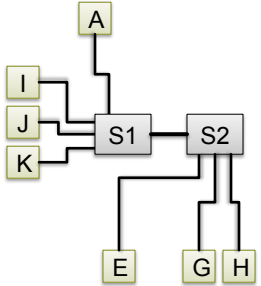
Real circuits

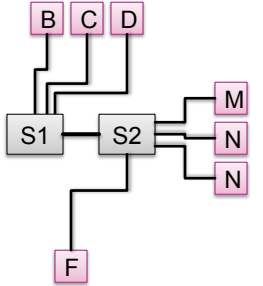
33

ETH zürich spcl.inf.ethz.ch
@spcl_eth

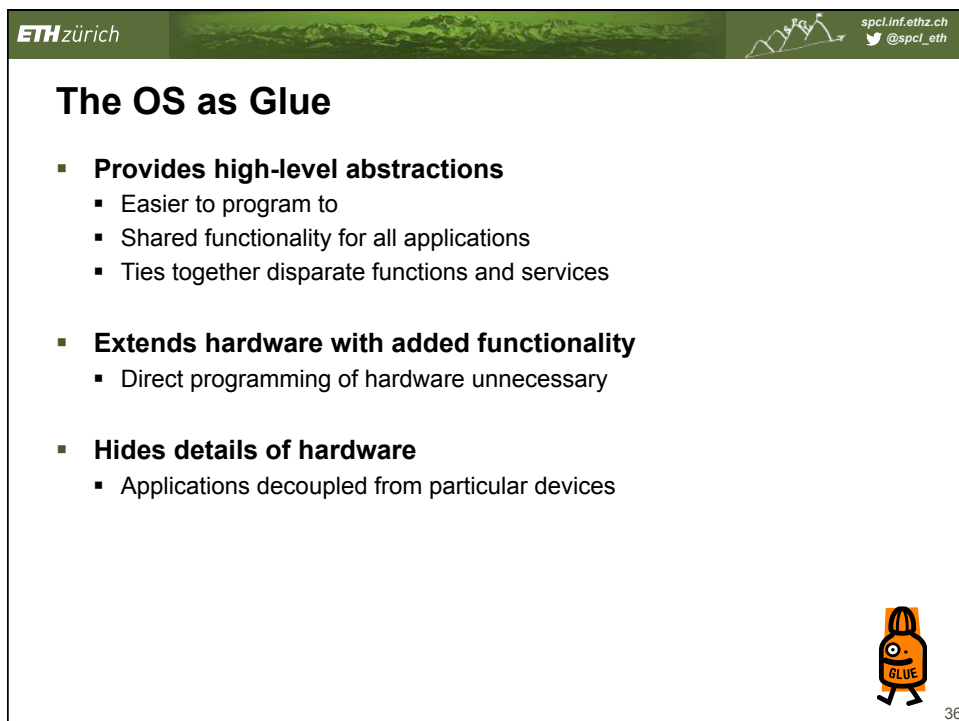
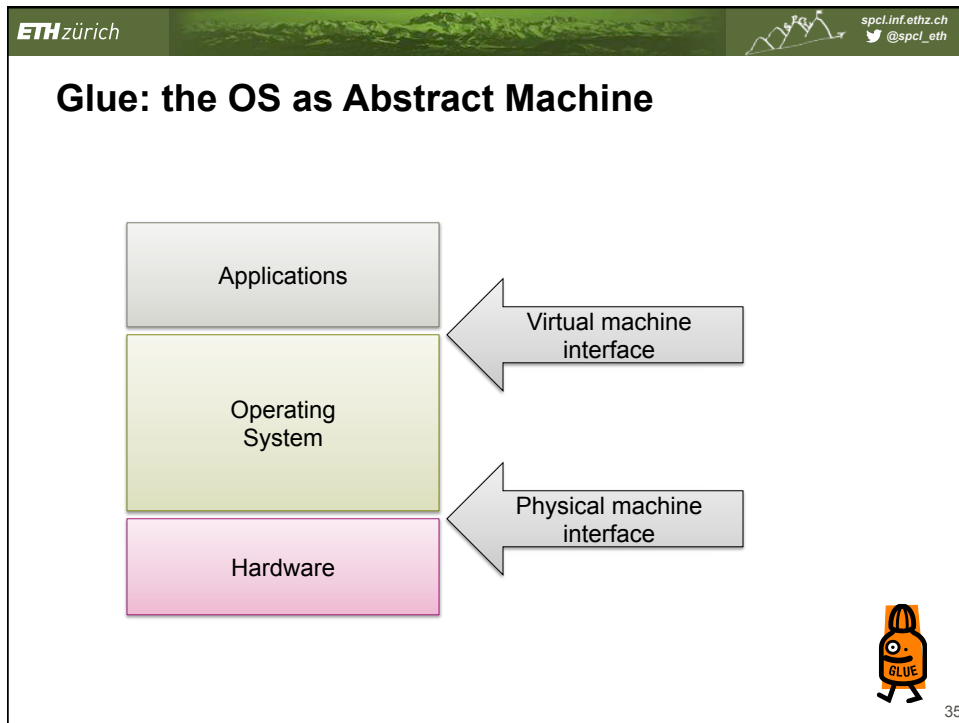
Example: VLANs



- **Methods:** multiplexing
- **Mechanisms:** port assignment, tags






34





ETH zürich   sycl.inf.ethz.ch
[@sycl_eth](https://twitter.com/sycl_eth)

Services provided by an OS

- **Program execution**
 - Load program, execute on 1 or more processors
- **Access to I/O devices**
 - Disk, network, keyboard, screen,...
- **Protection and access control**
 - For files, connections, etc.
- **Error detection and reporting**
 - Trap handling, etc.
- **Accounting and auditing**
 - Statistics, billing, forensics, etc.

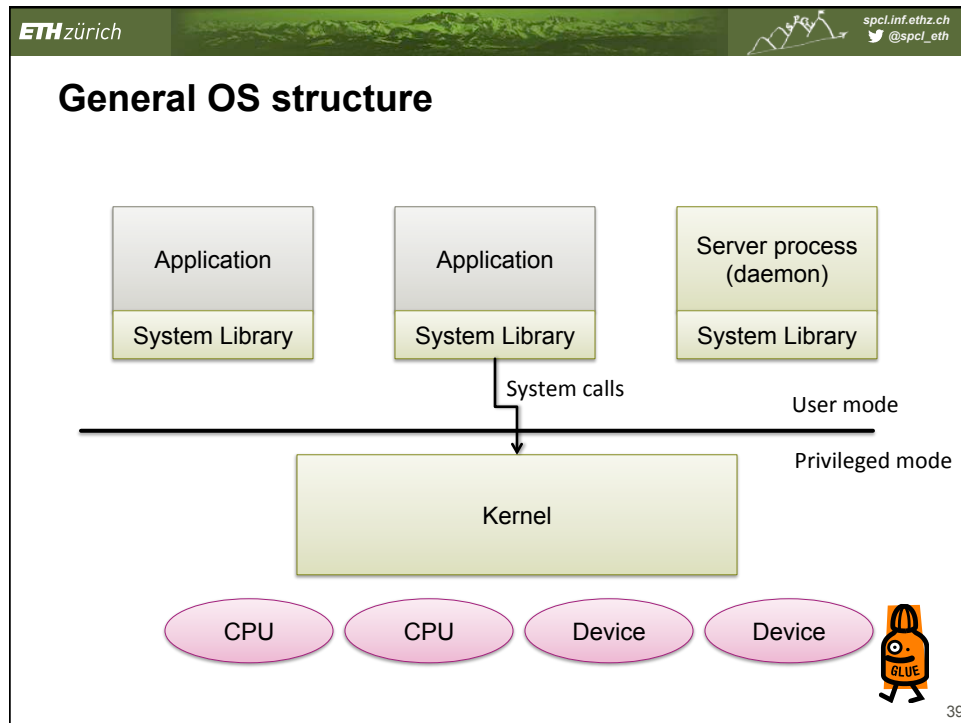


37

ETH zürich   sycl.inf.ethz.ch
[@sycl_eth](https://twitter.com/sycl_eth)

Operating System Structure

38

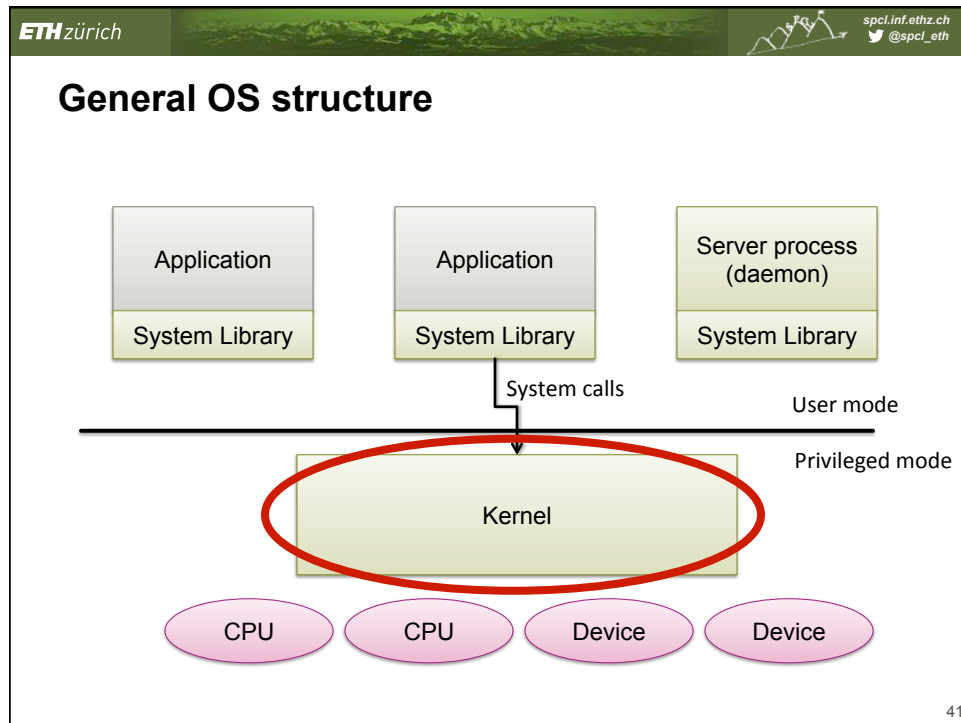


ETH zürich spci.inf.ethz.ch
@spci_eth

Privileged Mode and User Mode

- **As we saw in Computer Architecture, most CPUs have a “privileged mode”:**
 - ia32 protection rings
 - VAX kernel mode
 - Etc.
- **Most Operating Systems use this for protection**
 - In particular, protecting the OS from applications!

40



ETH zürich spci.inf.ethz.ch
@spci_eth

Kernel

- **That part of the OS which runs in privileged mode**
 - Large part of Unix and Windows (except libraries)
 - Small part of L4, Barrelfish, etc. (microkernels)
 - Does not exist in some embedded systems
- **Also known as:**
 - Nucleus, nub, supervisor, ...

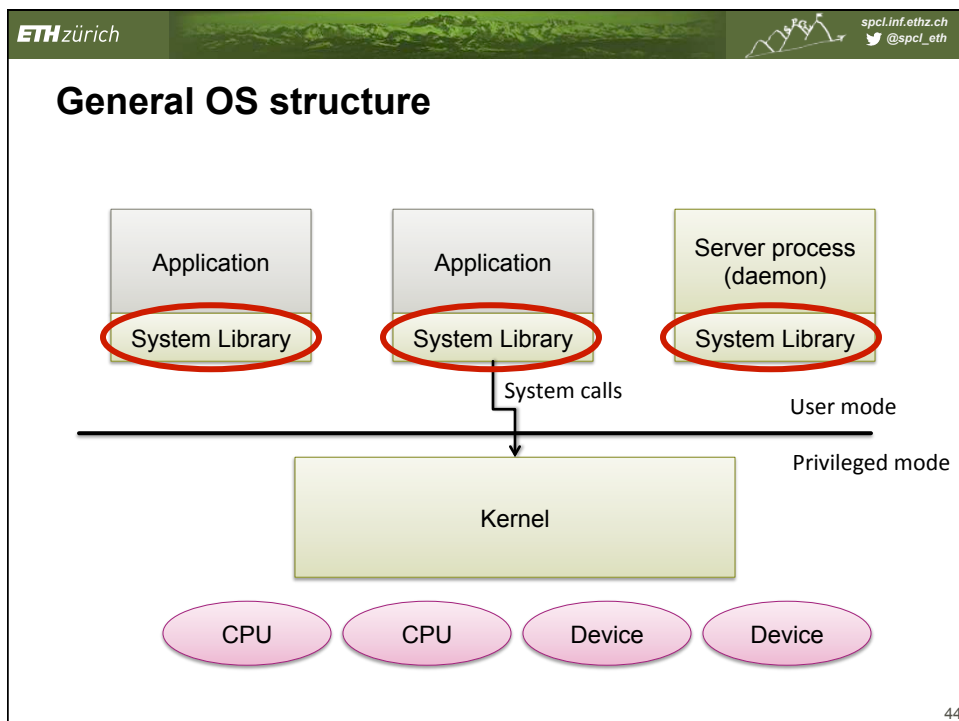
42

ETH zürich spci.inf.ethz.ch
@spci_eth

The kernel is a program!

- Kernel is just a (special) computer program.
- Typically an event-driven server.
- Responds to multiple entry points:
 - System calls
 - Hardware interrupts
 - Program traps
- May also include internal threads.

43

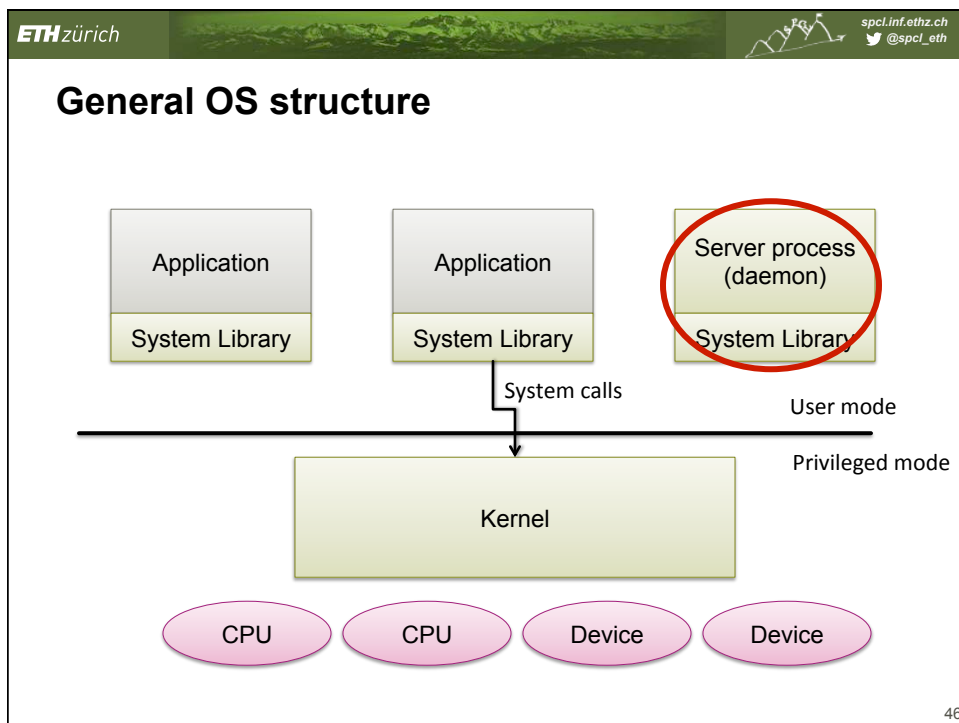





ETH zürich sycl.inf.ethz.ch
@sycl_eth

System Libraries

- **Convenience functions**
 - strcmp(), etc.
 - Common functionality
- **System call wrappers**
 - Create and execute system calls from high-level languages
 - See 'man syscalls' on Linux


45






ETH zürich   sycl.inf.ethz.ch
 @sycl_eth

Daemons

- **Processes which are part of the OS**
 - Microkernels: most of the OS
 - Linux: increasingly large quantity
- **Advantages:**
 - Modularity, fault tolerance
 - Easier to *schedule*...



47

ETH zürich   sycl.inf.ethz.ch
 @sycl_eth

Entering and exiting the kernel

48

ETH zürich spci.inf.ethz.ch
@spci_eth

When is the kernel entered?

- Startup
- Exception: caused by user program
- Interrupt: caused by “something else”
- System calls

49



ETH zürich spci.inf.ethz.ch
@spci_eth

Recall: System Calls

- RPC to the kernel
- Kernel is a series of syscall event handlers
- Mechanism is hardware-dependent

The diagram illustrates the flow of a system call. A horizontal line separates 'User mode' (top) from 'Privileged mode' (bottom). In User mode, a box labeled 'User process runs' has an arrow pointing to 'Execute syscall'. From 'Execute syscall', an arrow points down to 'Execute kernel code' in Privileged mode. From 'Execute kernel code', an arrow points up to 'Process resumes' in User mode.

System calls 50

ETH zürich   sycl.inf.ethz.ch
[@sycl_eth](https://twitter.com/sycl_eth)

System call arguments



Syscalls are *the* way a program requests services from the kernel.

Implementation varies:

- Passed in processor registers
- Stored in memory (address in register)
- Pushed on the stack

- System library (libc) wraps as a C function
- Kernel code wraps handler as C call

51

ETH zürich   sycl.inf.ethz.ch
[@sycl_eth](https://twitter.com/sycl_eth)

When is the kernel exited?

- **Creating a new process**
 - Including startup
- **Resuming a process after a trap**
 - Exception, interrupt or system call
- **User-level upcall**
 - Much like an interrupt, but to user-level
- **Switching to another process**

52