Computer Systems

Exercise Session 1

About me

- Jonas Gude
- jgude@student.ethz.ch
- Master's Student at ETHZ, focus on Distributed Systems and Information Security
- I'm here to help you to understand the course and to pass the exam :)

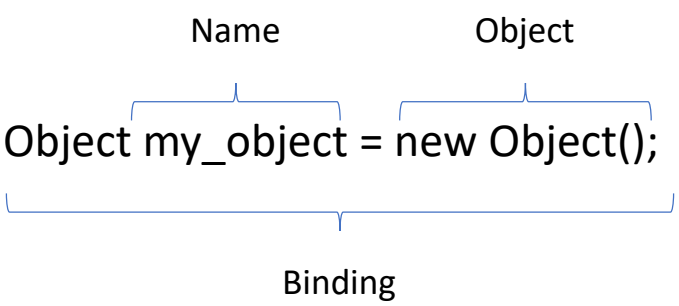
Organization

- Course is loosely divided into two parts:
  - Operating Systems (Prof. Roscoe)
  - Distributed Computing (Prof. Wattenhofer)

- Submit Exercises to: jgude@student.ethz.ch
  - If you have questions email me or ask me during/after the exercise sessions

- Bonus Task (design an exam Question)
  - Submit to: manuelei@ethz.ch
  - Deadline 1st attempt: somewhere in November
  - Deadline 2nd attempt: Probably end of semester

Naming Basics

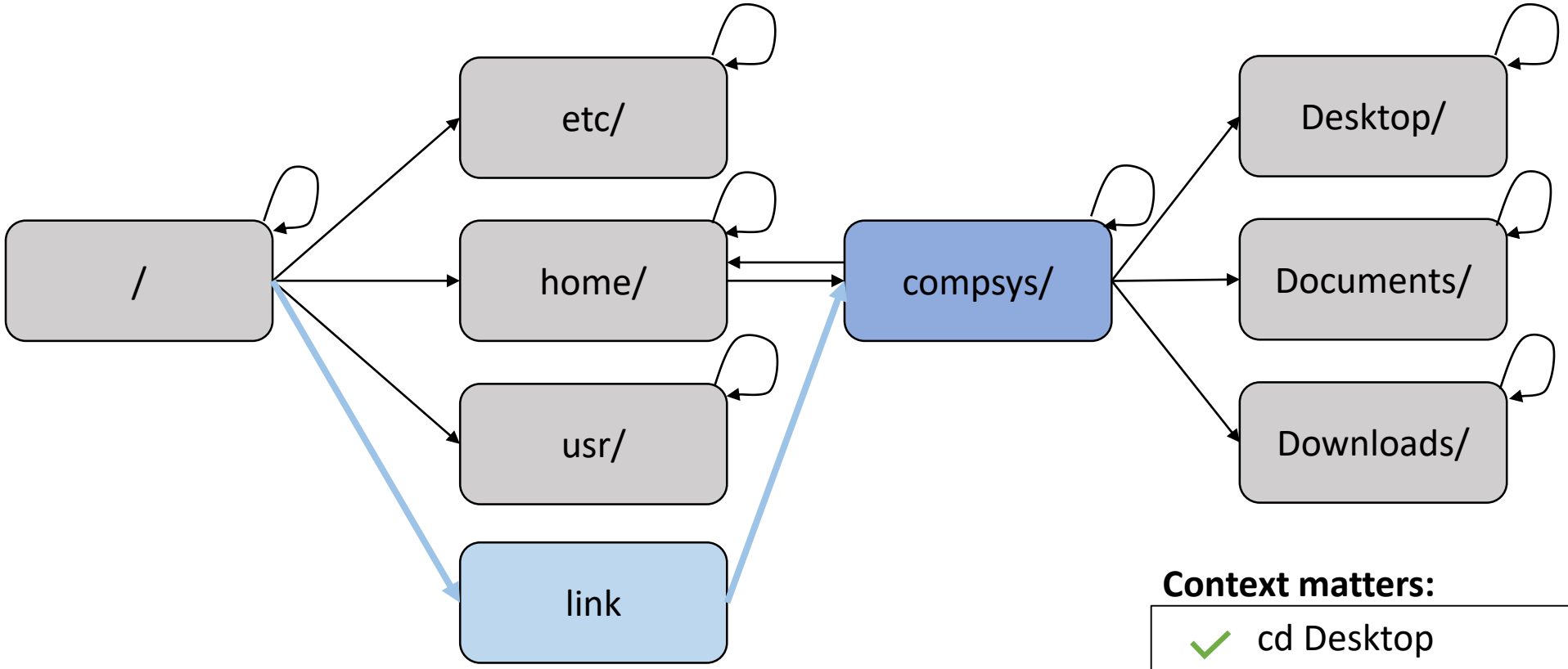


Global scope

Function scope

```
1 #include <stdio.h>
2
3 /* global variable definition */
4 int x,y;
5
6 x = 0;
7 y = 0;
8
9 int main () {
10     /* local variable definition and initialization */
11     int y,z;
12
13     y = 10;
14     z = 10;
15
16     printf ("value of x = %d, y = %d and z = %d\n", x, y, z);
17     return 0;
18 }
```

Naming Networks



Create SymLink: ln -s /home/compsys /link

Context matters:

- ✓ cd Desktop
- ✗ cd home
- ✓ cd /home/compsys/Downloads
- ✓ cd /link/Downloads
- ✓ cd Desktop/../Downloads

Search Paths

compSys@ubuntu:~$ echo $PATH

/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:

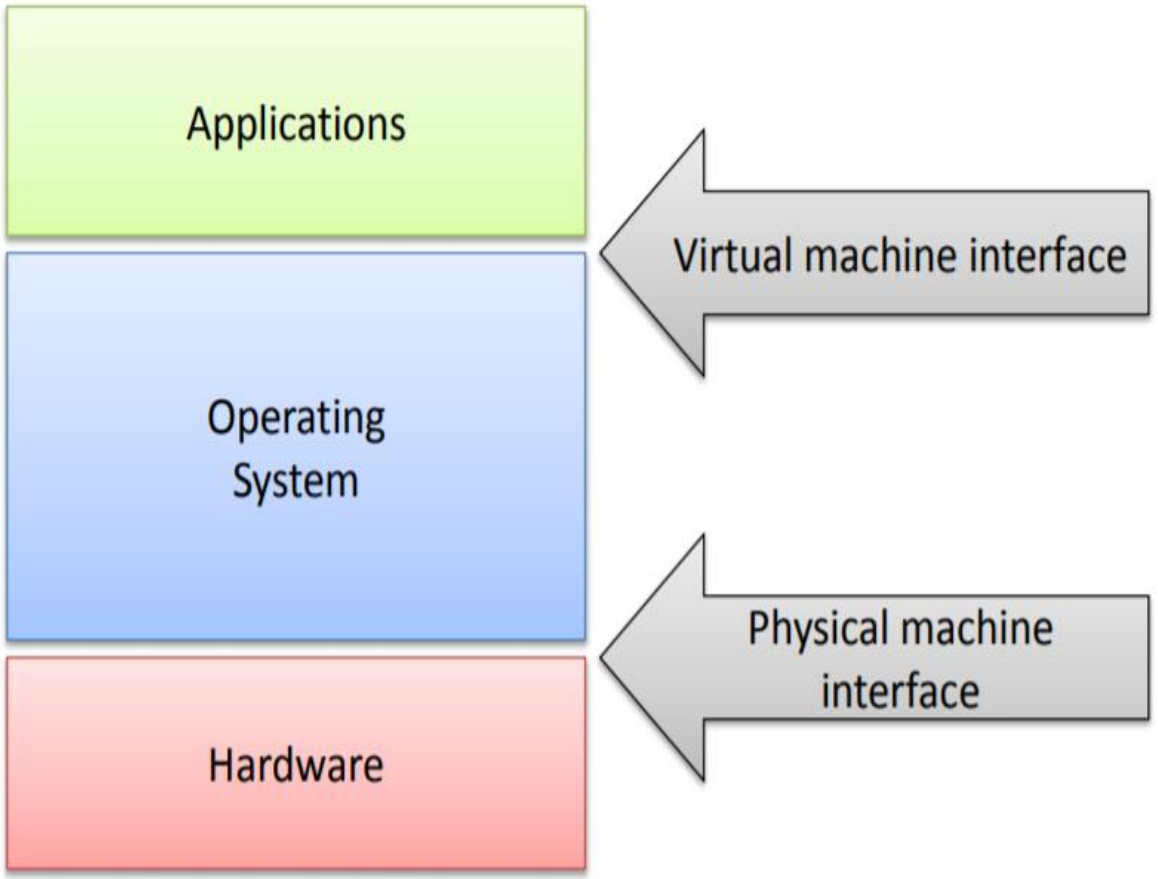
compSys@ubuntu:~$ ln -ls

…

compSys@ubuntu:~$ custom_command

custom_command: command not found

The Role of the OS



Referee:

- Ensure resource sharing
- Ensure protection (mem protection, process isolation)
- Ensure Inter-process communication

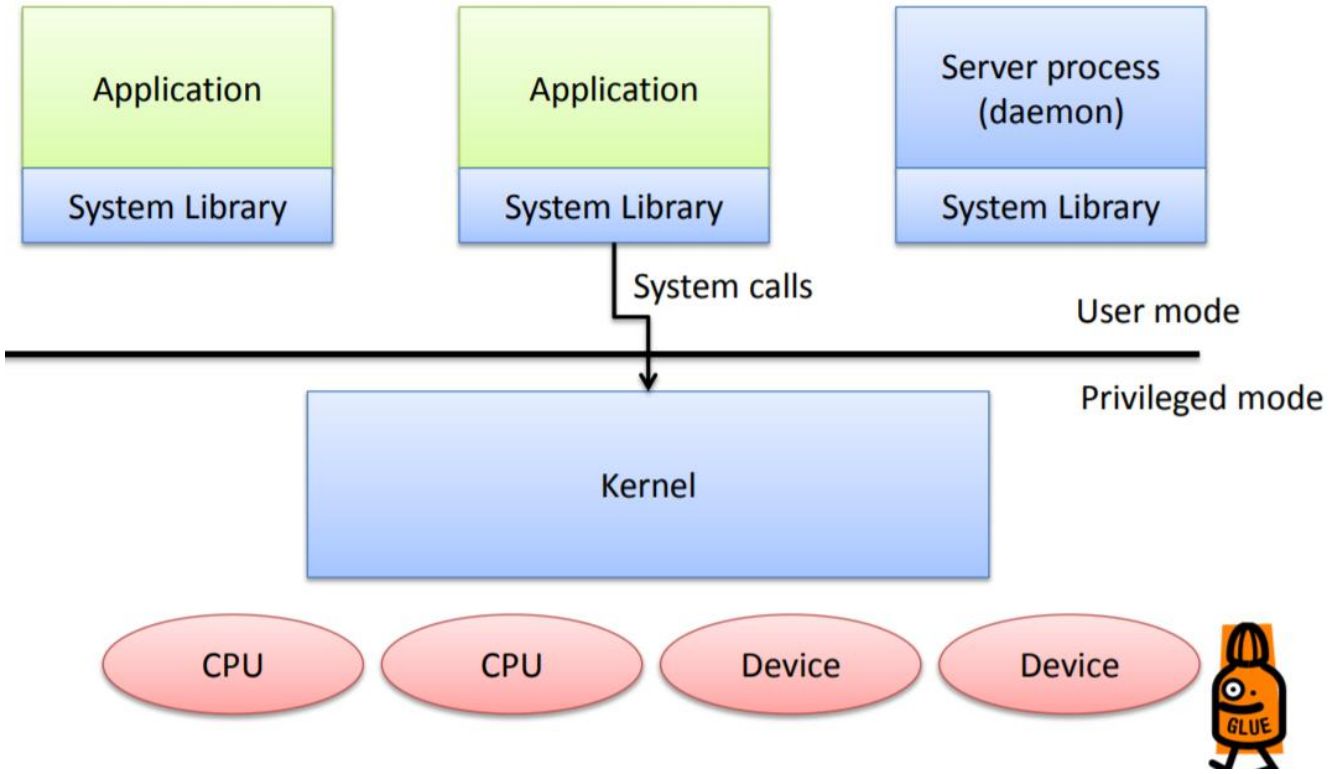
Illusionist:

- provide virtual resources to user-space processes
- Virtual Memory (full address space)
- Shared network interface

Glue:

- Provide high-level abstraction to user-space applications

General OS Structure



Kernel:

- Special process that runs in privileged mode
- Typically event driven server

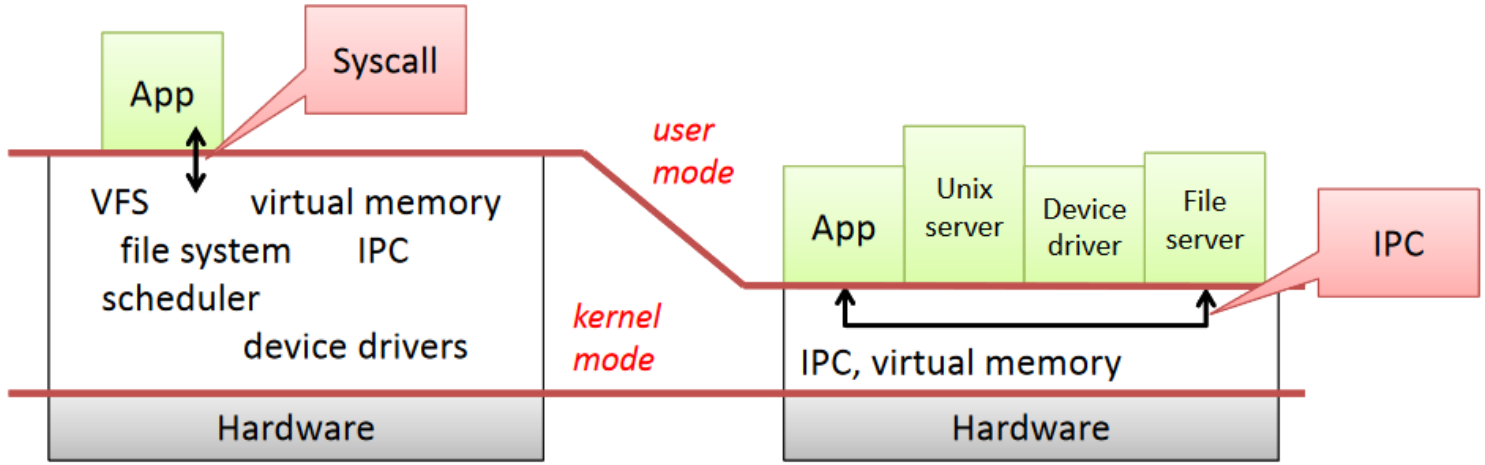
SysLib:

- Provide convenience functions and sysCall wrappers

Deamon:

- Processes which are part of the kernel but live in user-space
- Provides modularity, fault-tolerance

Monolithic vs. Microkernel



- Monolithic OS
  - lots of privileged code
  - services invoked by syscall
- Microkernel OS:
  - little privileged code
  - services invoked by IPC
  - “horizontal” structure

Exokernel: move functionality into system libraries instead of user-space

Multikernel: run different kernels on different cores (e.g. Barrelffish)

Bootstrapping

1. Power on
2. Load BIOS from ROM to memory
3. BIOS loads boot loader into memory
4. boot loader loads OS (kernel) from disk
5. Transfer control to OS

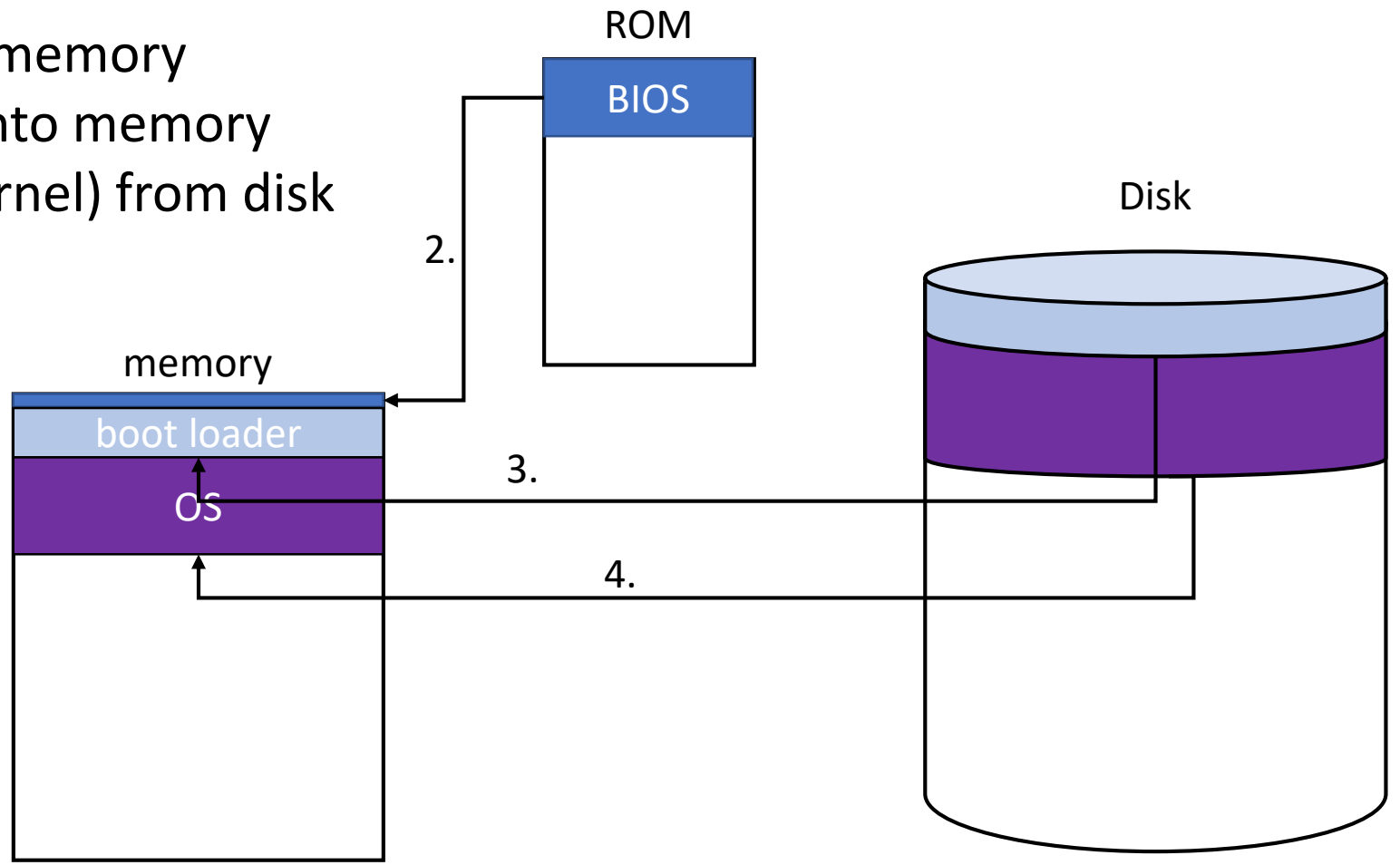


Illustration adapted from Maegan Pitman

Entering and Leaving the Kernel

- on Start-Up
- Exception occurs(caused by program)
- Interrupt occurs (caused by “something else)
- upon a System Call