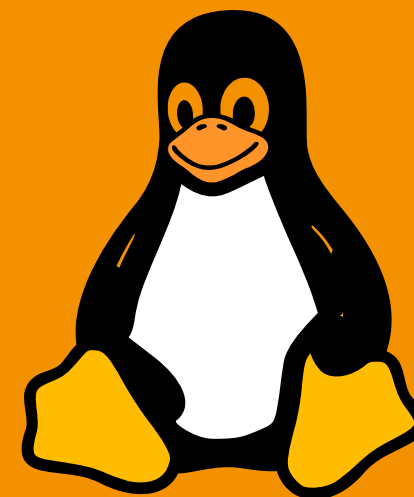




Prof. Dr. Florian Künzner

Technical University of Applied Sciences Rosenheim, Computer Science

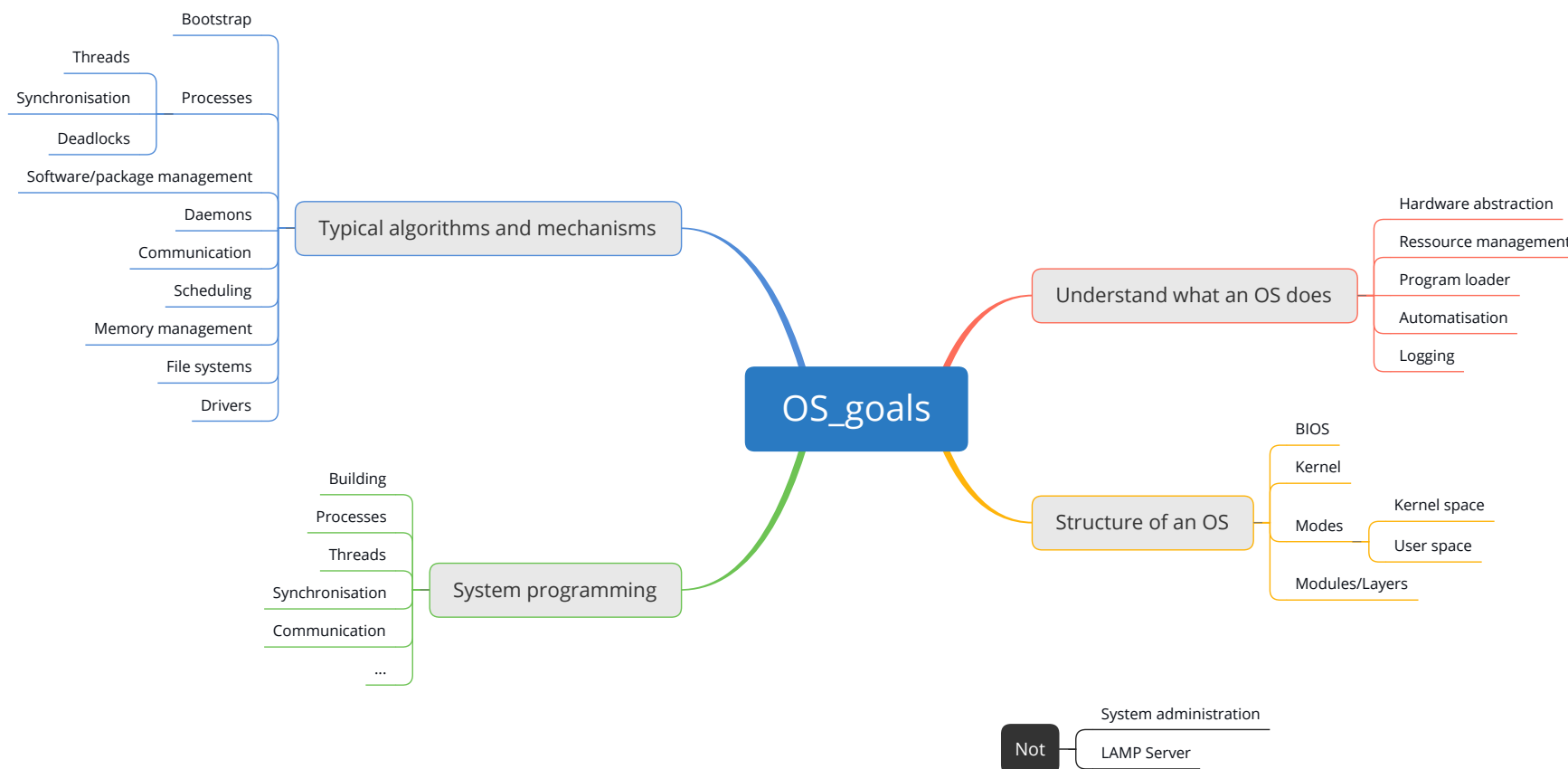
OS 4 – System boot; OS architecture



source: icons.png.com

The lecture is based on the work and the documents of Prof. Dr. Ludwig Frank

Goal



Goal

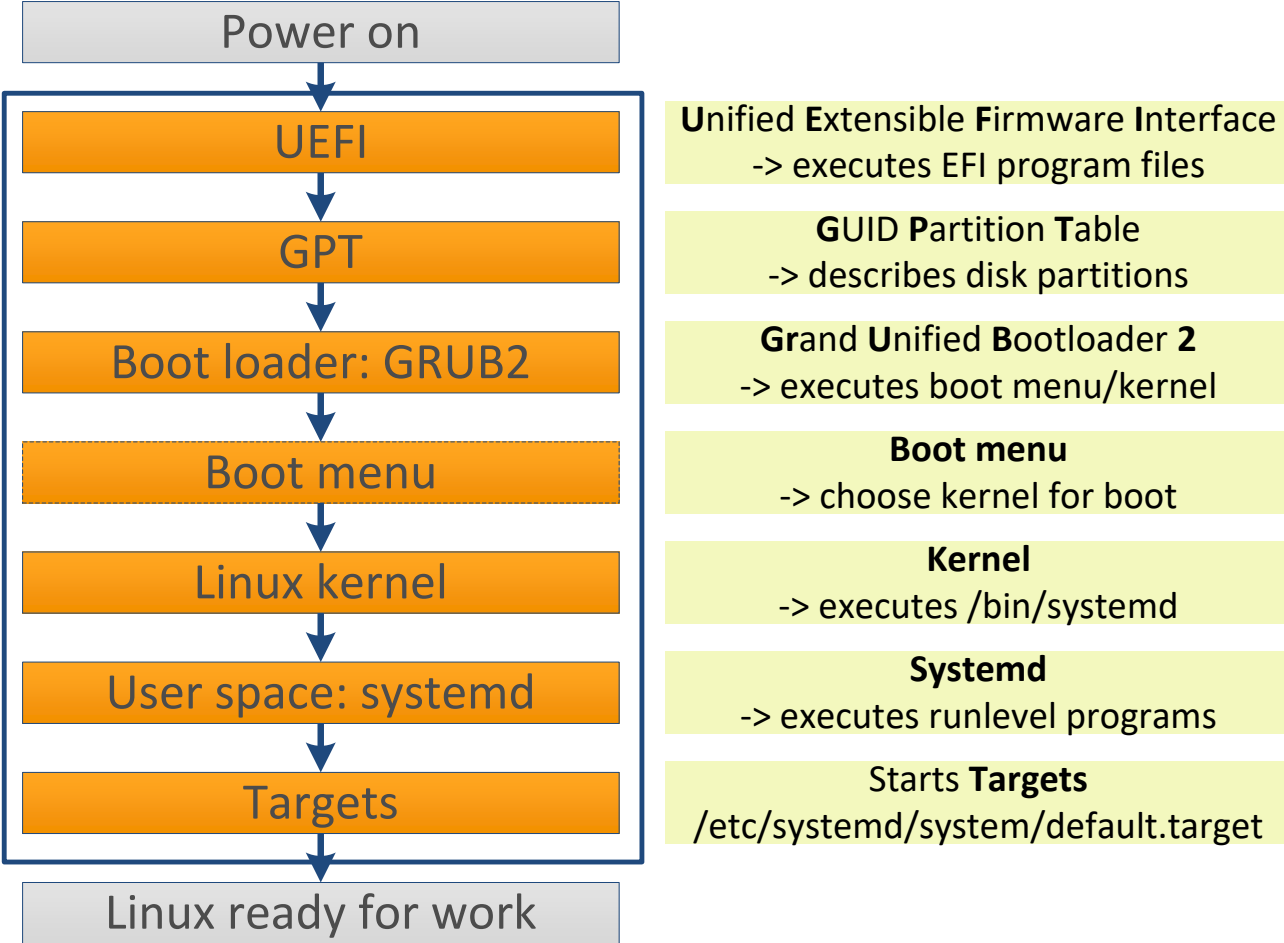
OS::System boot; OS architecture

- Boot procedure
- BIOS/UEFI
- MBR/GPT
- Boot loader: Grub2
- init/systemd
- OS architecture

Boot

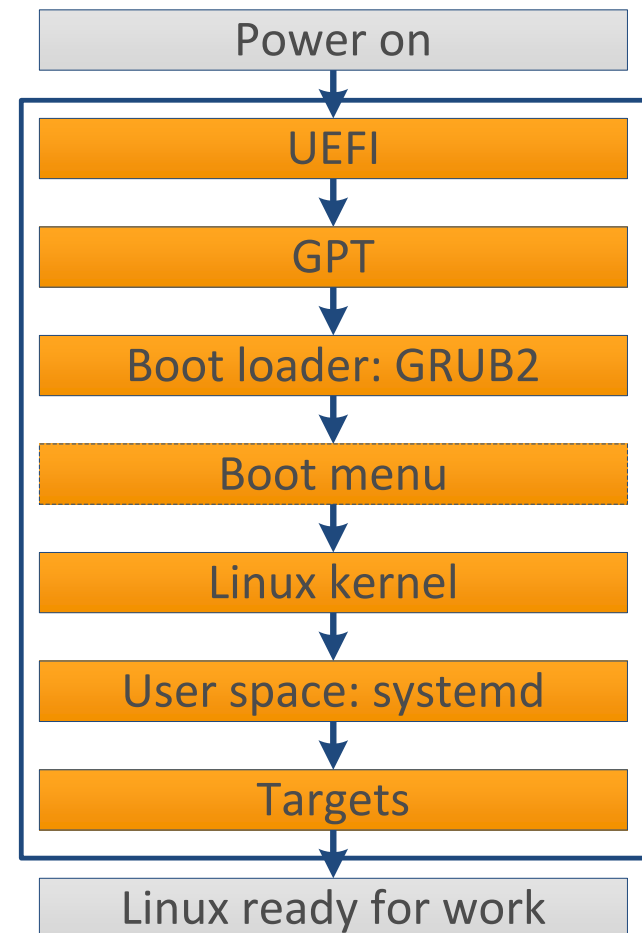
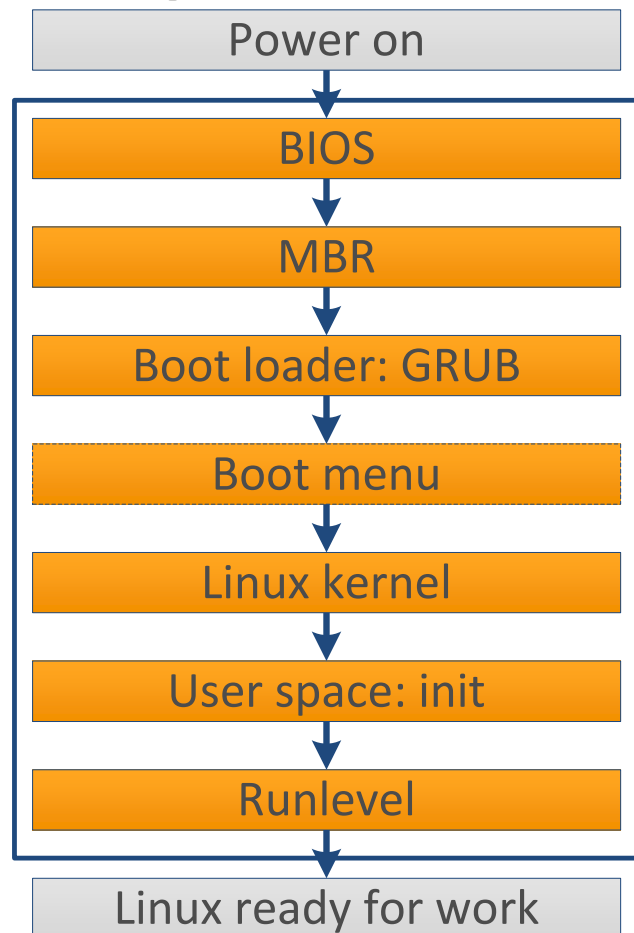
How does a PC boot into the operating system?

Boot procedure: UEFI (typically)





Boot procedure: BIOS vs UEFI



UEFI - Unified ext. firmware interface

- Since 2010 in consumer products
- Graphical user interface (with mouse)
- Fast boot: cache + hibernation (Windows only)
- Secure boot:
 - Protection against malware
 - Prevents against execution of unsigned code
- Network boot
- Modular interface for applications and devices (EFI drivers)
- Supported modes:
 - UEFI mode: Requires an EFI partition on boot device
 - BIOS mode: Old way of booting

More details: <https://www.marksei.com/bios-uefi-explained>

MBR - Master Boot Record

MBR (512 bytes)

Master boot code
(446 bytes)

Partition table (64 bytes)

1st
entry

2nd
entry

3rd
entry

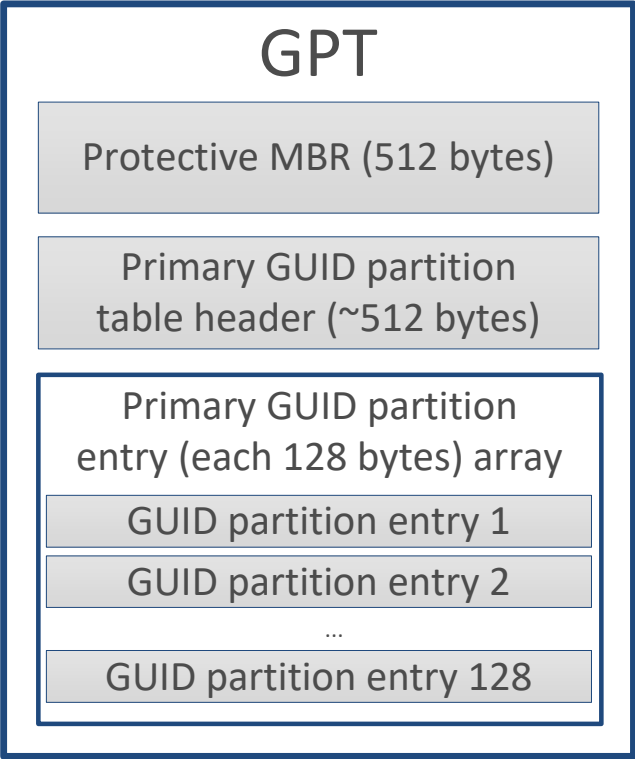
4th
entry

Magic Number: 0x55, 0xAA

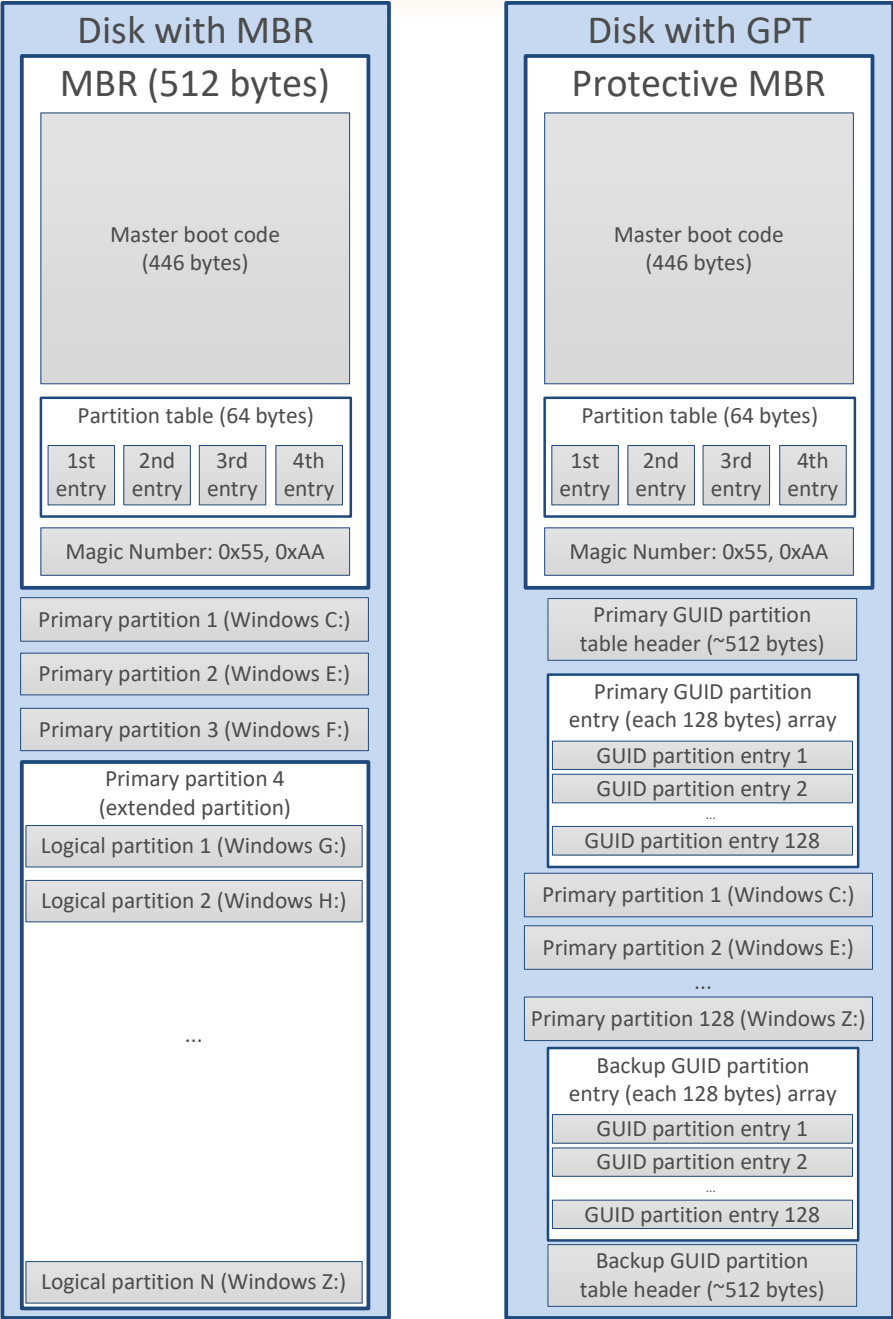
- Supports up to **2 TiB disks**
- Supports up to **2 TiB partitions**
- No safety (**no checksum**)
- Supports **4 primary partitions**
- Supports **one extended partition** (not bootable)



GPT - GUID Partition Table

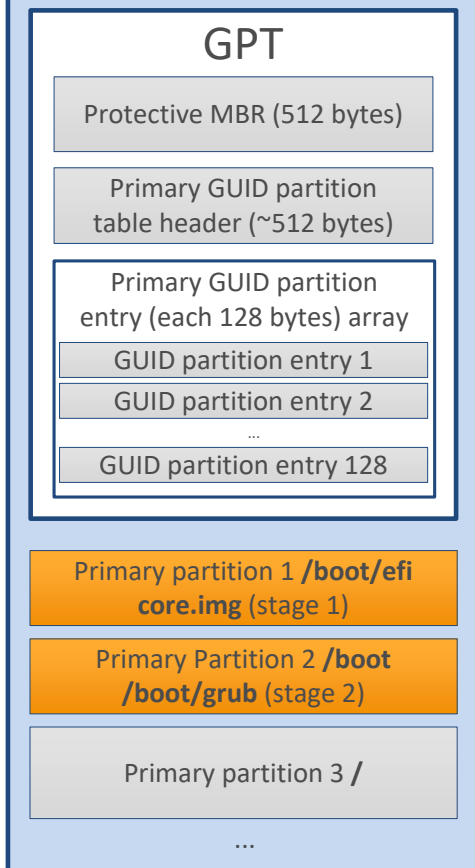


- Supports up to **18 EiB disks**
- Supports up to **18 EiB partitions**
- Safety (**checksum, backup**)
- Version number
- Supports **128 primary partitions**
- Does not have a boot code
- UEFI **boots** from an **EFI partition**



GRUB2 - GRand Unified Bootloader 2

Disk with GPT + GRUB2



- Boots in stages:
 - **stage1** (core.img): Loads directly stage 2 (/boot/grub), usually from /boot partition. Contains file system drivers.
 - **stage2** (/boot/grub): Loads the default configuration file, the **kernel**, and any other modules needed.
- Themes, **graphical menus**, scripting support
- Uses **UUIDs** to identify disks
- Supports additionally: PowerPC
- Supports LLVM and RAID
- Boots live CD images from hard drive
- Automated search for other OS (like Windows)

User space: systemd

- **systemd** is the **first process** started by the kernel
- The mother of all subprocesses
- Has always **PID 1**
- Looks in `/etc/systemd/system/default.target` for default target
- **Executes** the default **target** (unit) scripts
- **Starts** the user space **processes** on boot:
 - Daemons (crond, httpd, sshd, inetd, syslogd, ...)
 - Terminals
 - Graphical desktop
 - ...
- Speed-up the boot: starts processes in **parallel** (where possible)

User space: systemd (commands)

- Get default target: `systemctl get-default`
- Change runlevel: `systemctl isolate poweroff.target`
- Control a daemon:
 `/etc/systemd/system/<daemon>.service`
 - `service daemon start`
 - `service daemon stop`
 - `service daemon reload`
 - `service daemon restart`
 - `service daemon status`
 - ...
- Enable a daemon: `systemctl enable daemon.service`
- Disable a daemon: `systemctl disable daemon.service`

User space: systemd (service example)

```
daemon.service:
```

```
1  [Unit]
2  Description=Demo daemon that does something useful
3
4  [Service]
5  Type=forking
6  ExecStart=/usr/sbin/daemon
7  ExecStopPost=/bin/rm /var/run/daemon.pid
8
9  [Install]
10 WantedBy=multi-user.target
```

More details: <https://www.digitalocean.com/community/tutorials/understanding-systemd-units-and-unit-files>

Init runlevel vs system targets

init runlevel	systemd target	systemd target aliases	Description
	halt.target		Halts the system without powering it down.
0	poweroff.target	runlevel0.target	Halts the system and turns the power off.
S	emergency.target		Single user mode. No services are running; filesystems are not mounted.
1	rescue.target	runlevel1.target	A base system including mounting the filesystems with only the most basic services running and a rescue shell on the main console.
2		runlevel2.target	Multiuser, without NFS but all other non-GUI services running.
3	multi-user.target	runlevel3.target	All services running but command line interface (CLI) only.
4		runlevel4.target	Unused.
5	graphical.target	runlevel5.target	Multiuser with a GUI.
6	reboot.target	runlevel6.target	Reboot.
	default.target		This target is always aliased with a symbolic link to either multi-user.target or graphical.target. systemd always uses the default.target to start the system.

More details: <https://opensource.com/article/17/2/linux-boot-and-startup>

Linux high level overview (1)

GUI

guake

code

xreader

Desktop (Mate)

Window Manager

X Window System / X11

Console

Shell

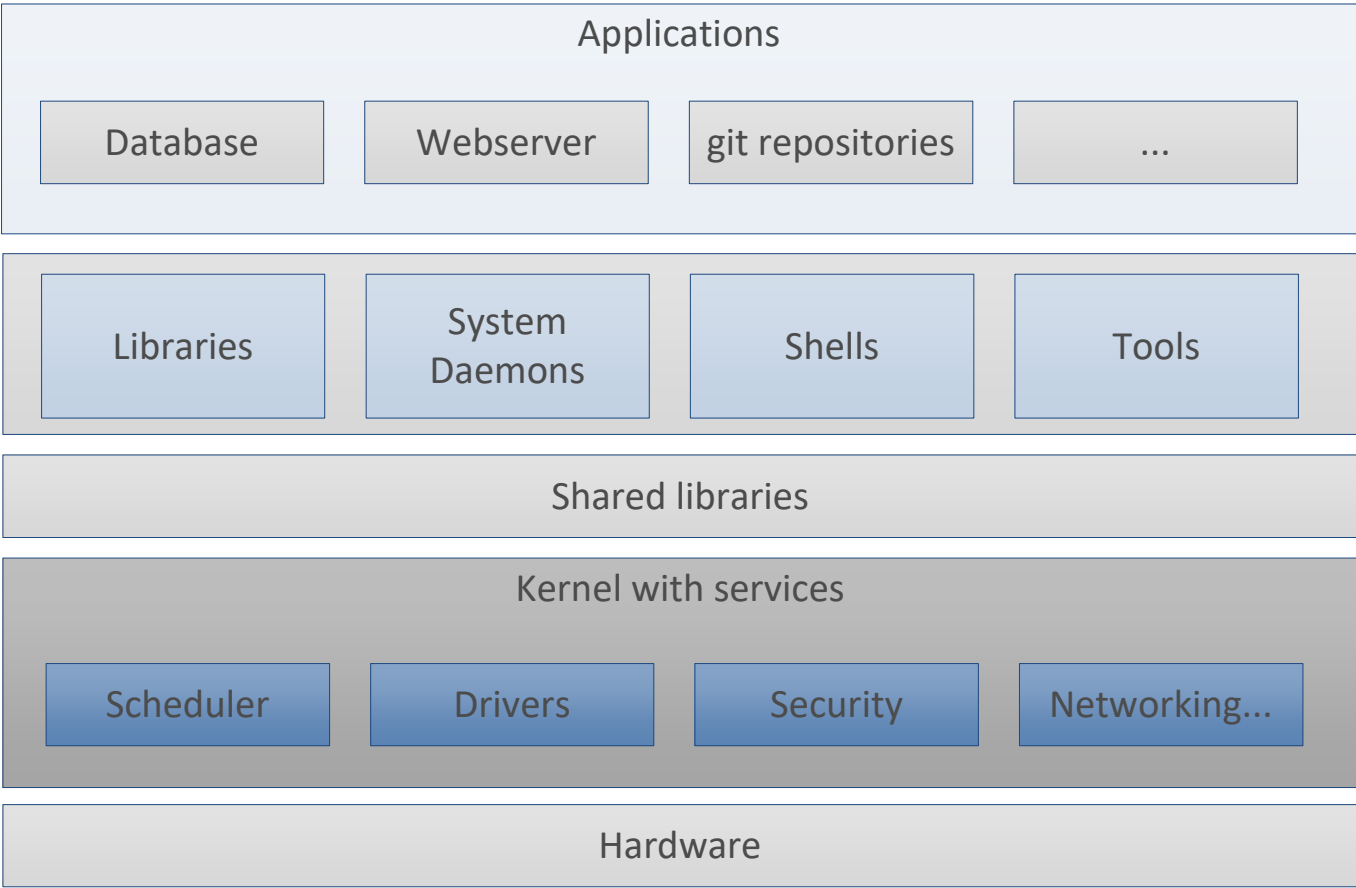
Kernel with services

Hardware

OS tasks

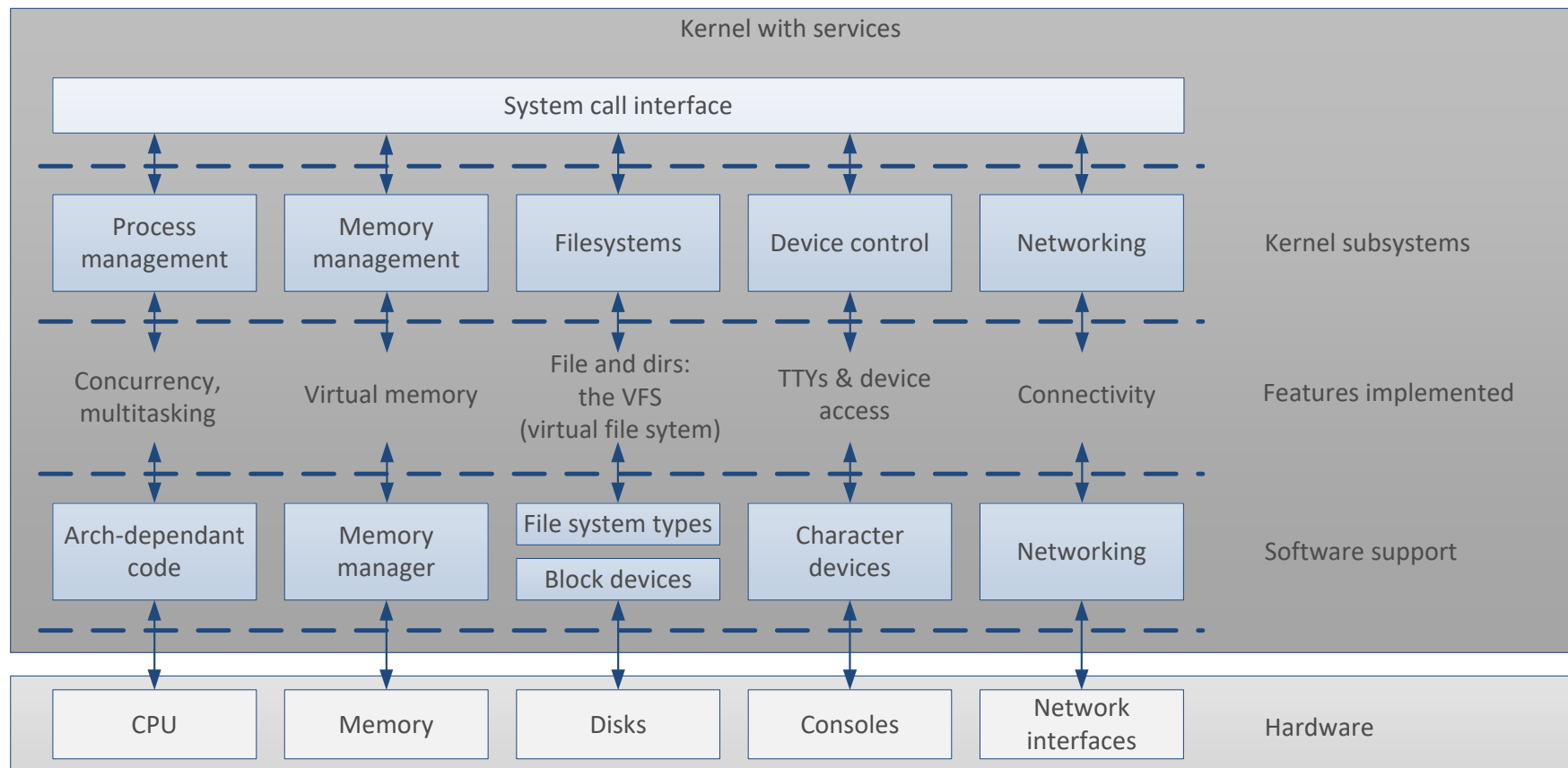
- Execute (graphical) user applications
- Provide desktop environment
- Draw windows (graphical elements)
- Provide shells
- Manage resources
- Support, abstract and virtualise hardware

Linux high level overview (2)

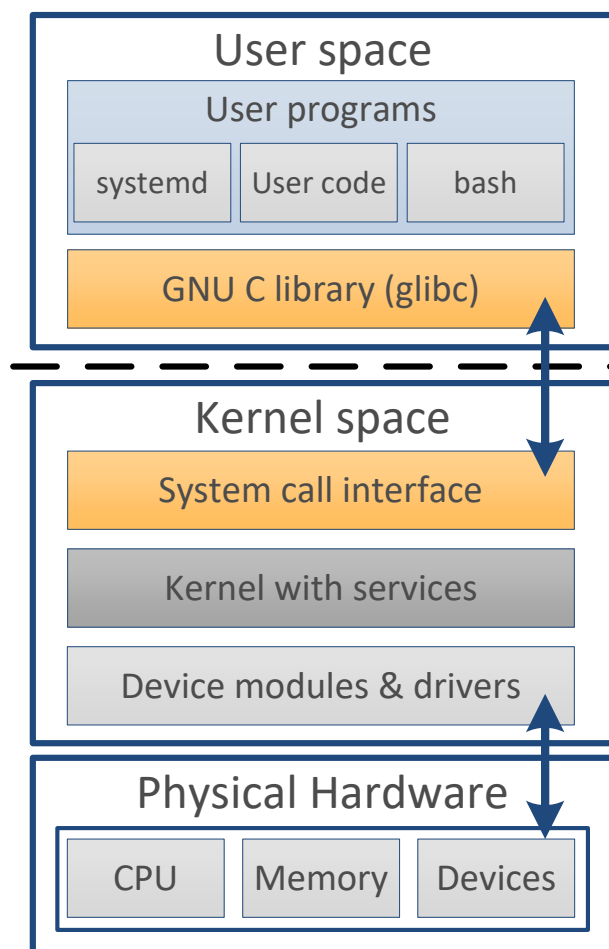




OS overview: Linux kernel



Protection: user vs kernel space



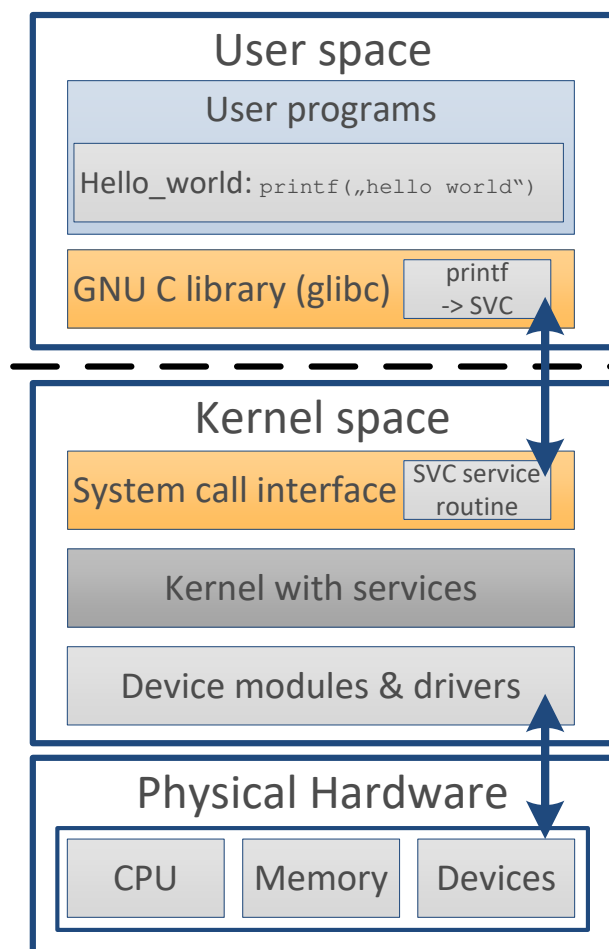
User space

- All the code outside the kernel
- Also called “userland”
- **Restricted** (encapsulated) **access** to the **hardware**
- Can **only** use a **subset of CPU instructions**
- Can **only** access the **assigned memory** addresses
- **Crash** in a user process: **only stops the process.**

Kernel space

- **Complete** and **unrestricted access** to the **hardware**
- Can execute **any CPU instruction**
- Can access **any memory address**
- **Crashes** in kernel are “catastrophic”: **system stop!**

Accessing the kernel: supervisor call

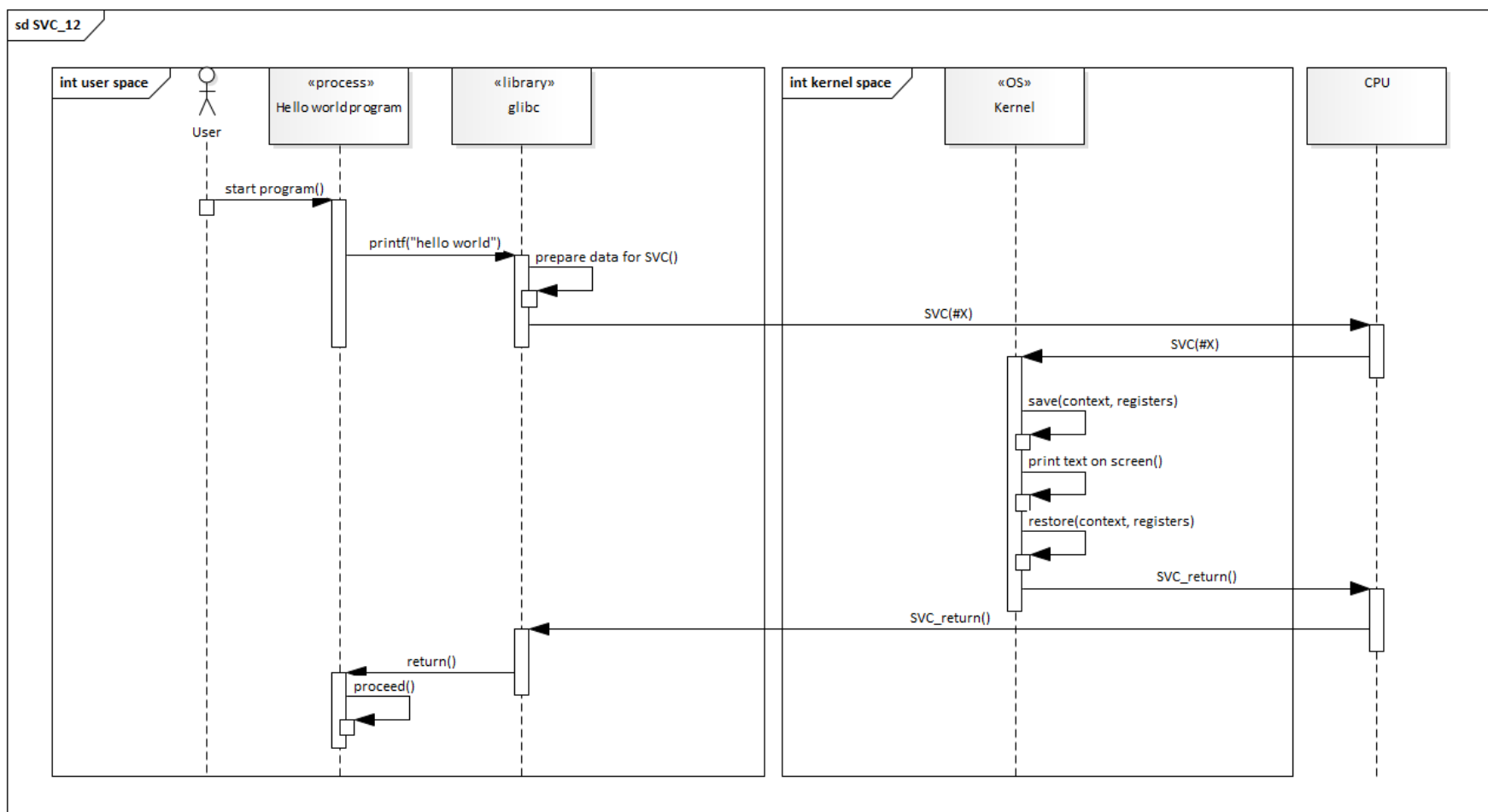


Supervisor call (SVC)

- **CPU instruction** to give control to the OS/kernel
- Requests for an **OS service**:
 - Start process
 - Allocate memory
 - File open/close/read/write/...
 - Print something on terminal (TTY) or screen
 - Send data over network
 - ...
- SVCs are **numbered**
- The calling **process is interrupted** while the kernel executes the SVC.



Supervisor call (SVC) example sequence



Summary and outlook

Summary

- Boot procedure
- BIOS/UEFI
- MBR/GPT
- Boot loader: Grub2
- init/systemd
- OS architecture

Outlook

- Processes vs Threads
- Parallelisation