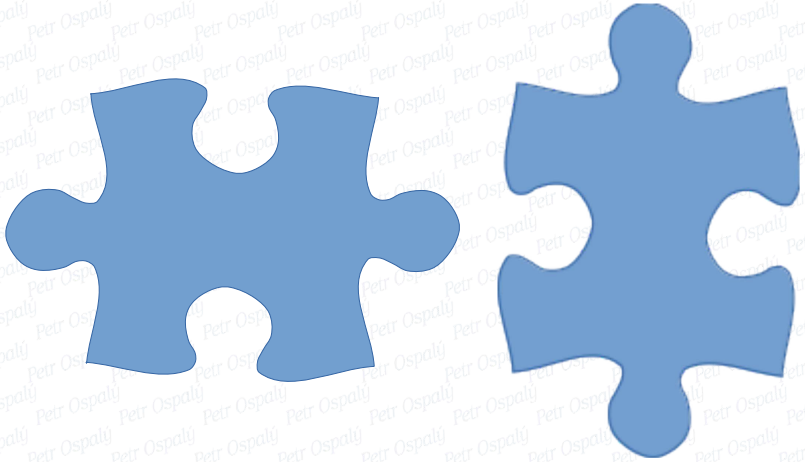


# Intro to Operating Systems

Petr Ospalý

May 2025

# # Operating System

- Umbrella term for a ***software*** which exposes hardware and other features via some high-level interface
- The hardware coverage, its exposure and the sophistication of the interface can vary greatly between different Operating Systems
- The most trivial OS can be just a firmware running on a handheld device (PDAs, GSM phones, graphic calculators)
- As long as the software enables the user to run custom program, routine or application then it is an OS

# # Examples of OSes (1/3)

- On the lowest side of things – a firmware can be dubbed an OS – for example this printer is running port of DOOM
- https://canitrundoom.org/
- https://youtu.be/XLHx3vO7KJM

# # Examples of OSes (2/3)

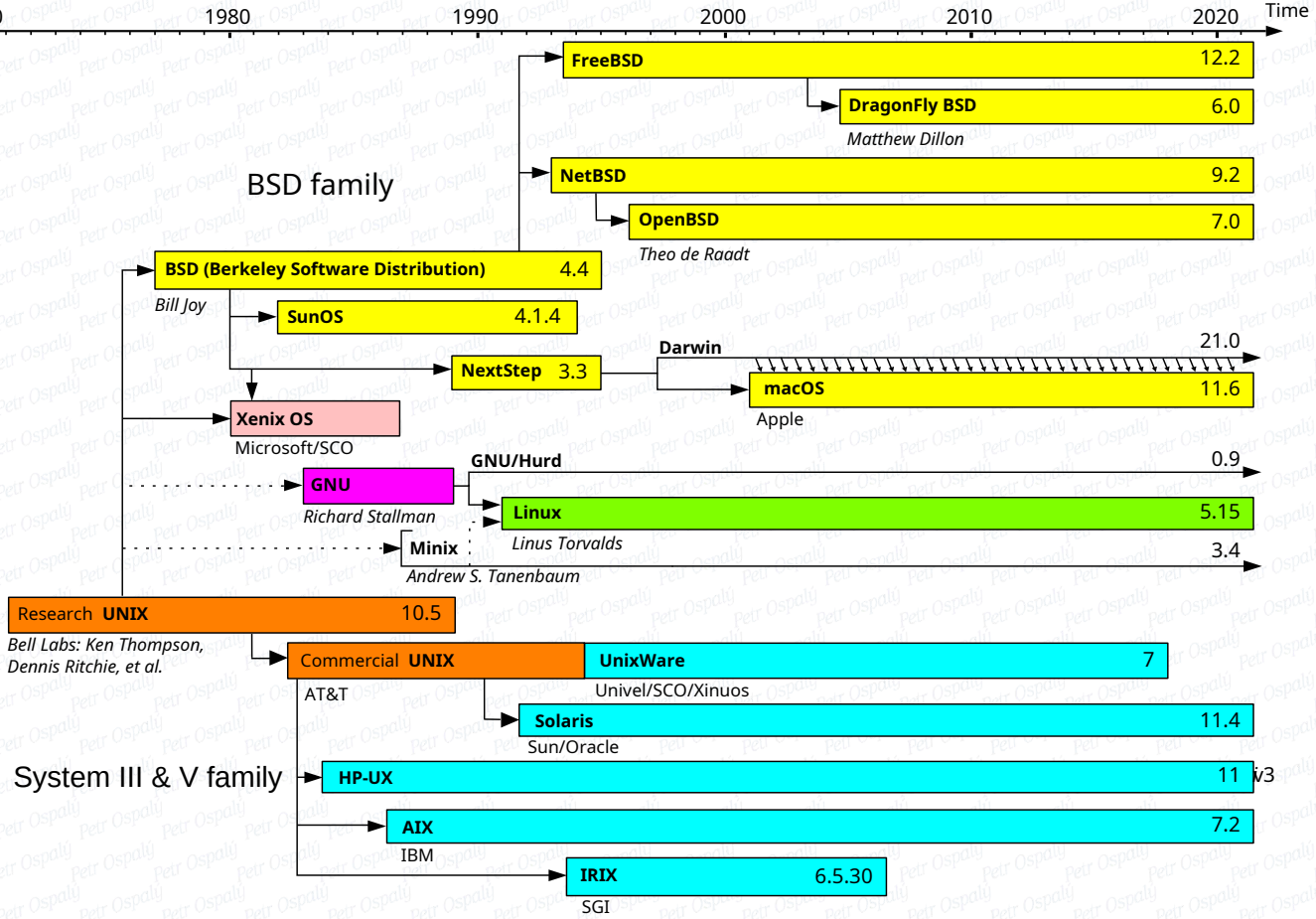
- MS Windows - the most widely used desktop OS
- OSX/Mac – Bundled together with Apple hardware – the second most distributed desktop system
- Android – Mobile OS with Linux kernel

# # Examples of OSes (3/3)

- Gaming Consoles
- Playstation OS
- Linux on PS2

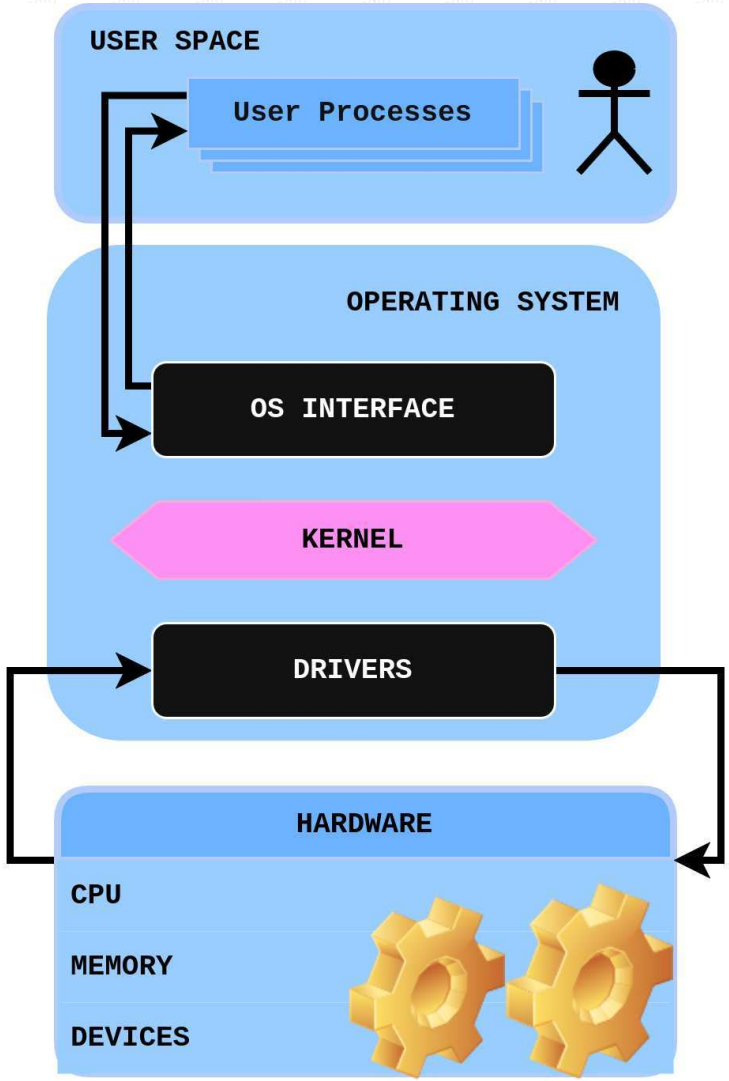

# # NIX family of OSes

- UNIX*
- *BSD
- Linux
- https://en.wikipedia.org/wiki/Unix_wars



# # Generic OS

- „Sits“ on top of hardware
- Consists of device drivers, kernel (the core) and user interface
- User space programs talk to the interface and are shielded from hardware



# # Linux OS (1/5)

- Initial implementation created by Finnish student Linus Torvalds who wanted its own **UNIX-like** system
- First public version released in 1991
- Currently the biggest open-source project and the backbone of the internet and cloud infrastructure

# # Linux OS (2/5)

- Linux project actually covers only the **kernel** portion of the Operating System
- To have complete feature-full OS one has to add **glibc** library and other user space programs and utilities – like a **bootloader** without which Linux will not even load (boot up)

# # Linux OS (3/5)

- So this “problem” is solved by bundling Linux kernel and all the necessary software together as so called “**Linux distribution**”
- There are countless such “**distros**” and new are created every year
- Most of them are just based on a few major ones

# # Linux OS (4/5)

- Linux is a new implementation of the UNIX OS – made from scratch
- UNIX was elegant and ubiquitous OS of the 1970+
- Linux does not share any of its code (while BSD variants do) but it provides the same faculties which made the UNIX so popular

# # Linux OS (5/5)

- The classic unix toolbox (grep, sed, awk, shell, pipes, redirections etc.) works (on the surface) the same in Linux as in the old systems
- Linux of course is a modern and powerful OS which over the years added much more new functionality and features and hw support

# # Distribution

- Linux is just a kernel and more components are needed to have fully operational OS
- Linux distribution is answer to that
- Major distros are only few and there is just plenty of offshoots and derivatives
- Maintainers came up with packages as a solution to conveniently distribute their Linux OS

# # Software packages (1/4)



- Linux community pioneered software distribution via packages and centralized repositories
- Package is a an archive with metadata tracking its own dependencies – recursively
- Initial packages were just simple tar archives and dependencies might had to be resolved manually
- Modern package managers will deal with all the work to install and track dependencies – both when installing and uninstalling






# # Software packages (2/4)

- Other platforms (like Windows) for longest time had nothing similar
- One had to visit software website and download installer manually
- Mobile app stores are similar in concept to software repositories
- Every distribution has their own repository

# # Software packages (3/4)



- Software handling packages is called: **package manager**
- Both the format and the tool differ between distro families, e.g.:
  - Debian (and Ubuntu family): **apt*** tools and ***.deb**
  - RedHat (CentOS, Fedora...): **yum/dnf** and ***.rpm**
  - Arch Linux: **pacman** and ***.pkg.tar.zst**
- They also support source packages which need to be compiled first (Gentoo is solely source based)
- Packages are plain tar archives with extra metadata files inside to which only the relevant package manager will understand






# # Software packages (4/4)



- Examples of usage (updating the system):
  - apt-get update && apt-get upgrade # debian
  - dnf update --refresh # redhat
  - pacman -Syu # Arch



- Tools differ sometimes significantly and/or have not the same feature-set

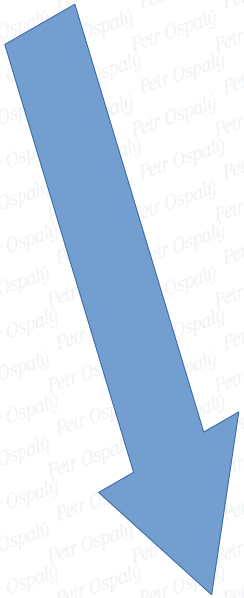


- And many times the package name differs too – e.g. **ssh** in debian but **openssh** in redhat...


# # Basic distro

- Bootloader (e.g. syslinux, grub)
- **Linux** kernel + **glibc** (GNU C standard library)
- Some fundamental tools including shell (command interpreter) to do basic user-space tasks
- Init program and service manager (sysvinit, runit, systemd…)
- Xserver to be able to run desktop
- Login manager or at least login command
- Desktop environment or Window manager

# Bootstrap process



## Power ON

Motherboard firmware (BIOS/UEFI) will do hardware initiliazation.

UEFI is basically OS on itself - it can even run internet browser.

## Bootloader stage 1

BIOS/UEFI will decide where to look for a bootloader - it can be on a HDD, SSD, flash drive, CD-ROM etc. - bootloader is a small program which can find and load the OS kernel image into memory.

## Bootloader stage 2

Bootloader has enough support to read different filesystems or raid devices to be able find a kernel image file. Once particular kernel file is selected it will be loaded to the memory and kernel will take over.

## Kernel

At this point the OS is actually running - some additional bootstrapping stages might be taking place (like replacing initial ramdisk) but kernel is now prepared to start the first user-space process.

## Init Process

Kernel will start first user-space process with PID number 1 and all other processes will derive (be children) of this process - usually it is some true init-like process but it can be any kind of program.

## User Login

Under normal operation - the init process will start multiple process/services including a login prompt which will enable a user to login and use the OS.

# # Terminology (1/3)

- **Boot**: Process of “bootstrapping” the OS (kernel)
- **Bootloader**: Small program which knows enough to find the OS kernel on the disk and load it to memory where the kernel can execute itself
- **BIOS/UEFI**: Firmware on the motherboard which can find and start bootloader
- **Disk**: It can be classic Hard Disk, Flash Drive or SSD – in general a block device
- **Block device**: Any disk or medium (Floppy Disc) which supports **Seek** operation
- **File**: Representation of a blocks of binary data as a higher structure in the OS
- **Filesystem**: System of organizing file data and metadata on the physical/virtual device (disk)
- **Directory**: Structure in the filesystem which enables collecting multiple files under one node (think of a folder)
- **Inode**: Unique identificator of the file or directory on unix filesystem

# # Terminology (2/3)

- **Program**: script or binary executable in a form of a file
- **Process**: a program file loaded to a memory and executed
- **PID**: Process ID – we can reference our processes with it – it is unique
- **CPU**: Processing unit which can execute the process instructions
- **Multiprocessor**: A system with more than one CPU (core) – it can run multiple processes at once
- **Context switch**: Usually OS is running more processes than it has CPU cores – so it must pause and switch out and in different processes – this is costly operation but enables illusion of multiple processes running simultaneously
- **Thread**: Lighter alternative to a process – it is a separate execution “thread” under the same process
- **Forked process**: More traditional way of parallel execution – it has bigger overhead than threads but it removes issues with dead-locks and double read/writes, races and all other problems plaguing bad parallel code

# # Terminology (3/3)

- **User**: Usually unprivileged account in the OS, there can be many of user accounts and groups
- **Root**: Privileged user account in the unix-like systems (think of a superadmin)
- **File permissions**: Unix have ownership, attributes and permissions associated with every file
- **Login**: Process of accessing the OS as a user - credentials (username/password) is needed
- **Shell**: Traditionally after the login the user is greeted with a shell interpreter which was the default user interface (command line interface)
- **CLI**: Command Line Interface
- **Terminal**: Emulated or physical device communicating via serial connection and traditional user facing device where shell could be accessed (screen and keyboard) – now mostly emulated as terminal application
- **Script**: Readable text file which can be executed in a similar manner as a program – it can be written in any language as long it is supported (has interpreter)

# # Login prompt after the boot

```
server login: friday
Password:
Welcome to Alpine!

The Alpine Wiki contains a large amount of how-to guides and general
information about administrating Alpine systems.
See <https://wiki.alpinelinux.org/>.

server:~$ whoami
friday
server:~$
```

# # Trivial init process tree

Notice that the „init“ process (PID 1) is a shell binary in this example which is not proper init in the general sense

```
USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root           1  0.0  0.0   2624  1664 pts/0    Ss   11:18   0:00 sh
root        1831  0.0  0.0   4164  3328 pts/0    S    11:21   0:00 bash
root        1847  0.0  0.0   3264  2432 ?        S    11:26   0:00  \_ login -- friday
friday      1848  0.0  0.0   4188  3328 ?        Ss   11:26   0:00      \_ -bash
friday      1855  0.0  0.0   4216  3072 ?        R    11:27   0:00          \_ ps aux --forest
```

# # Recap

- **OS must be able to run user programs**
- It should manage the whole lifetime of such program/process and return back to its default state
- OS abstracts the hardware and exposes it in the user space via its interfaces
- Kernel is the core of the OS but not the only part
- There are many Operating Systems both in architecture and feature sets

# # Final words

- **Linux comes in great variety…**
- Positive for some
- Negative for others
- Linux itself is just the kernel
- Distros can differ in filesystem layout, configuration files, admin tools and software management