Computer Architecture & Real-Time Operating System

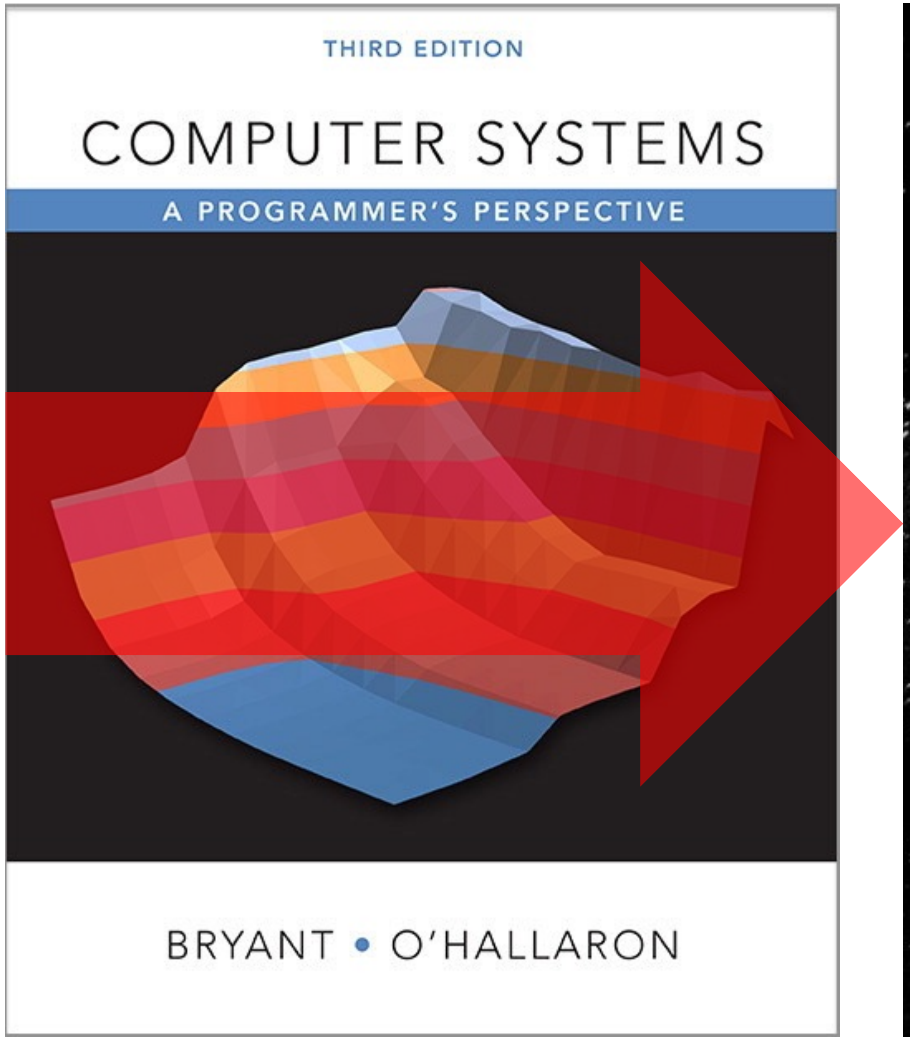
12. Operating System Overview

Prof. Jong-Chan Kim

Dept. Automobile and IT Convergence

KMU 국민대학교
KOOKMIN UNIVERSITY

We are moving on to the next topic (OS)



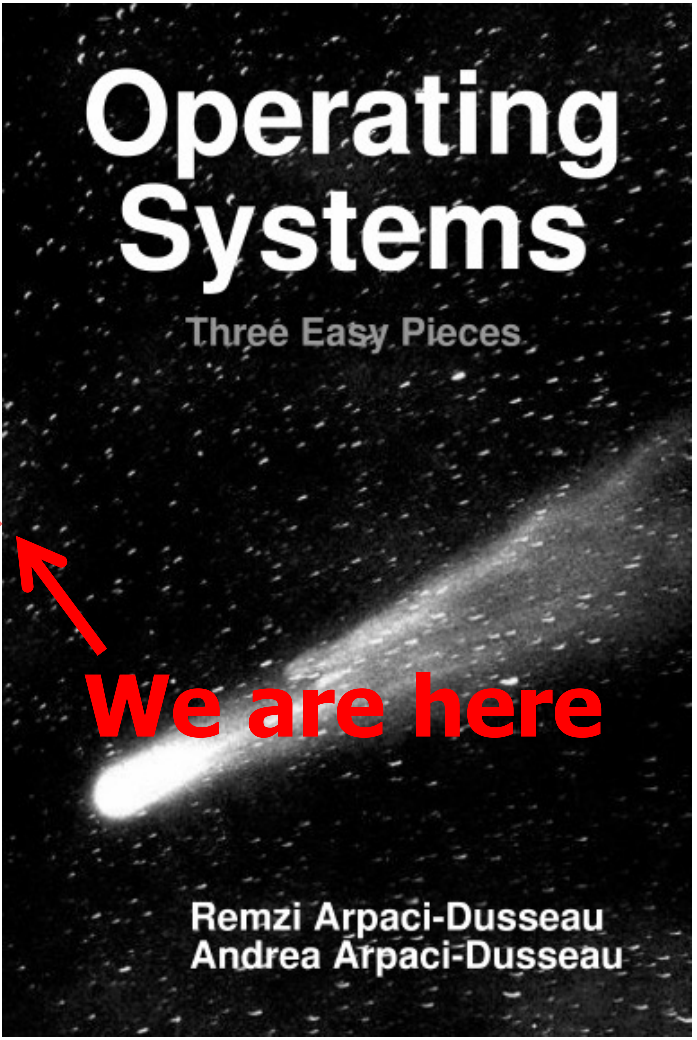
THIRD EDITION

COMPUTER SYSTEMS

A PROGRAMMER'S PERSPECTIVE

BRYANT • O'HALLARON

Computer Architecture Textbook



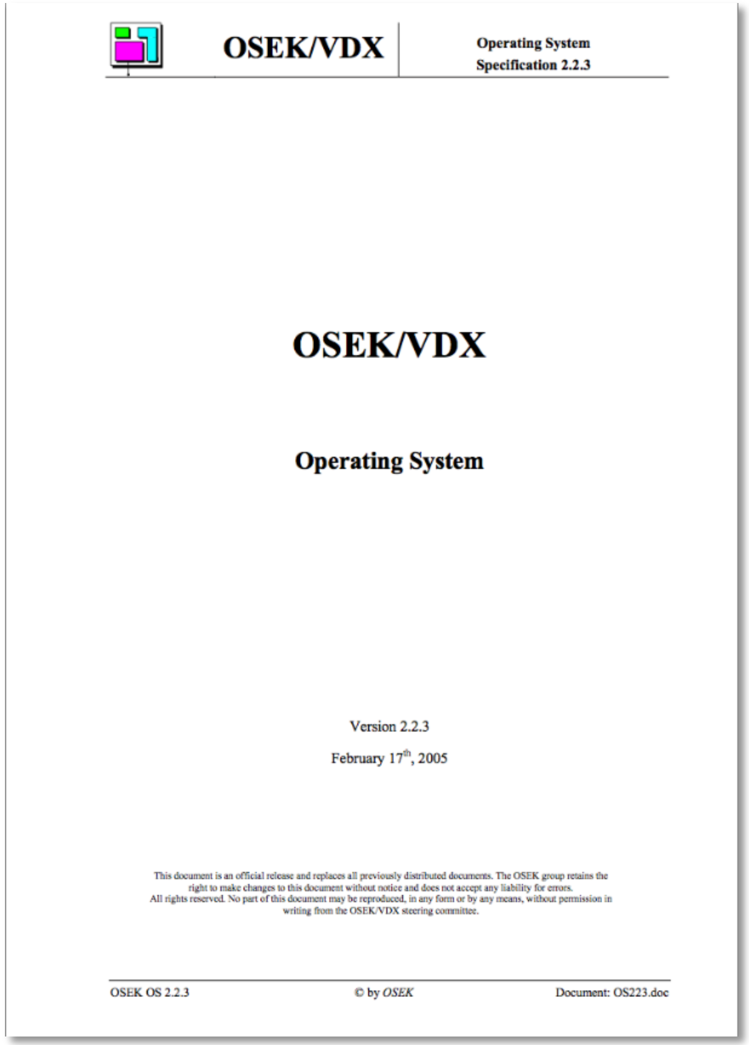
Operating Systems

Three Easy Pieces

Remzi Arpaci-Dusseau
Andrea Arpaci-Dusseau

We are here

OS Textbook



OSEK/VDX | Operating System Specification 2.2.3

OSEK/VDX

Operating System

Version 2.2.3

February 17th, 2005

This document is an official release and replaces all previously distributed documents. The OSEK group retains the right to make changes to this document without notice and does not accept any liability for errors.
All rights reserved. No part of this document may be reproduced, in any form or by any means, without permission in writing from the OSEK/VDX steering committee.

OSEK OS 2.2.3 © by *OSEK* Document: OS223.doc

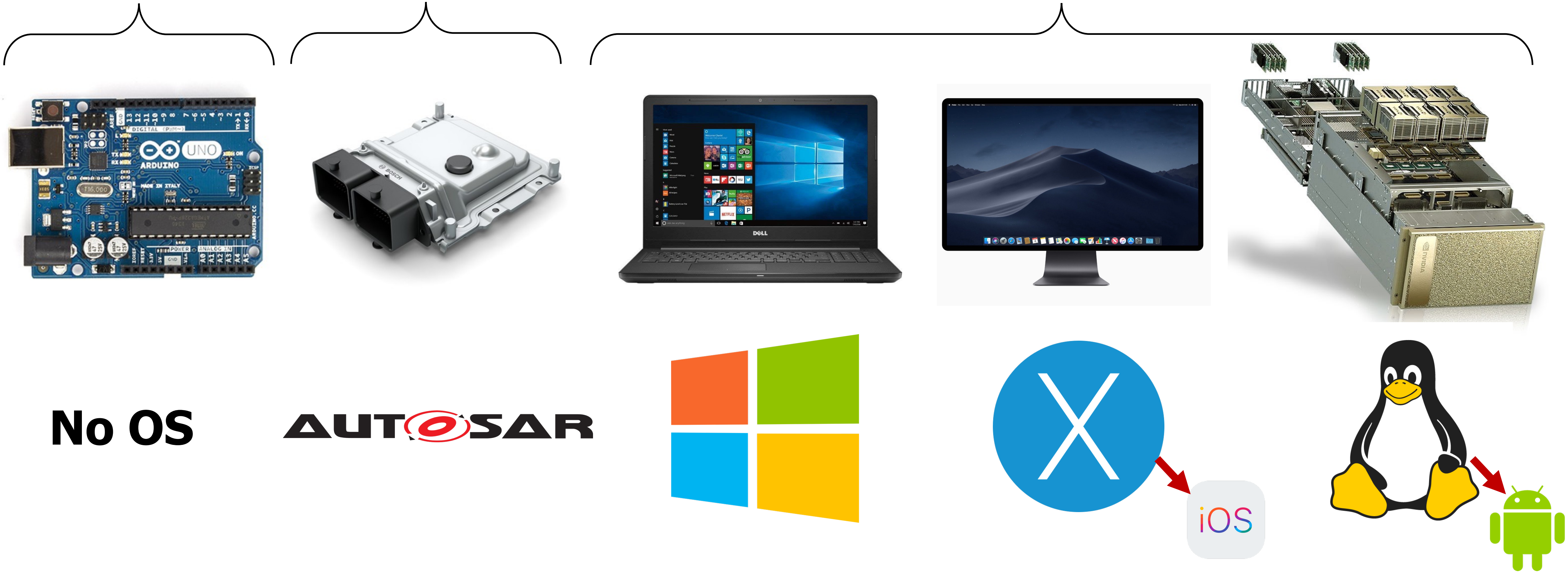
Why OS matters?

- Computers have fundamentally the same hardware architecture
- Different operating systems make the real difference

Firmware

Real-Time OS

General-Purpose OS

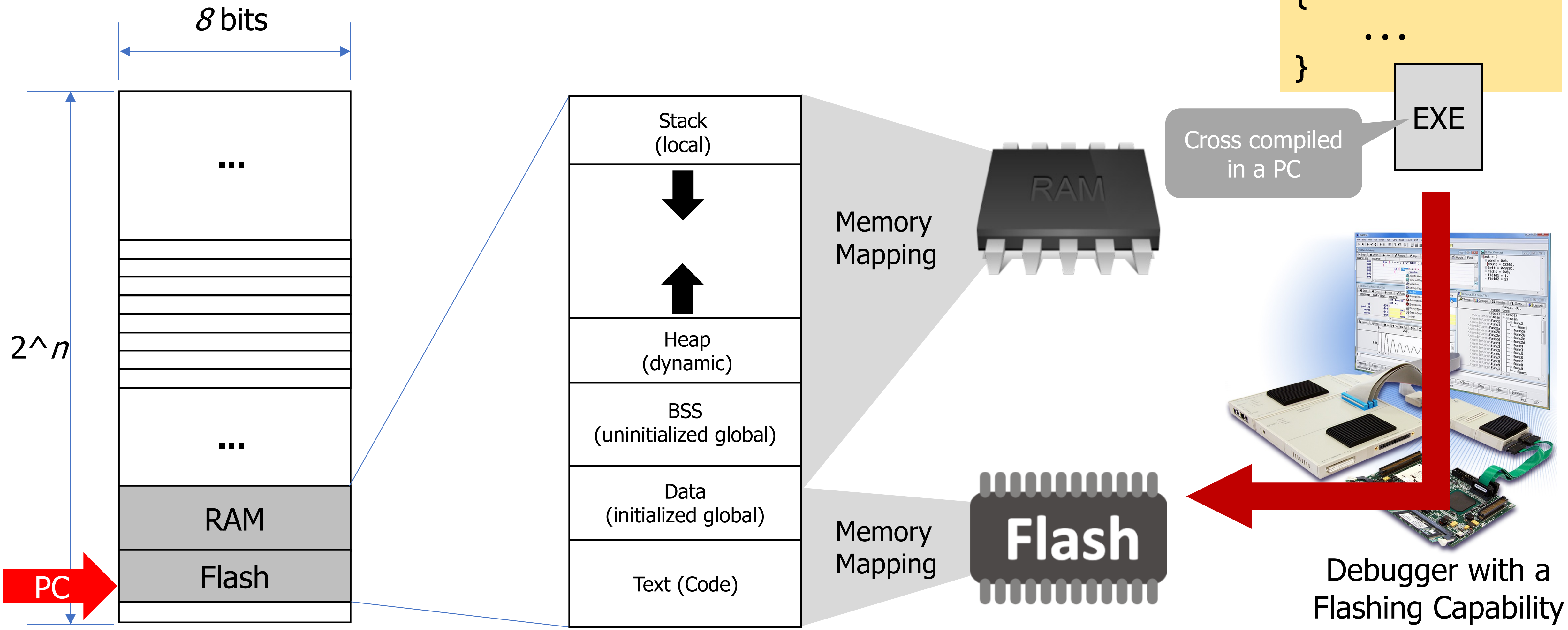


No OS

Firmware-based Systems

- A system with a single application having a single “main” function
- Programs directly executed on flash memory (no loading)

```
int main(void)
{
     ...
}
```



8 bits

2^*n*

...

...

RAM

Flash

PC

Stack
(local)

Heap
(dynamic)

BSS
(uninitialized global)

Data
(initialized global)

Text (Code)

Memory
Mapping

Memory
Mapping

EXE

Cross compiled
in a PC

Debugger with a
Flashing Capability

OS-based Systems

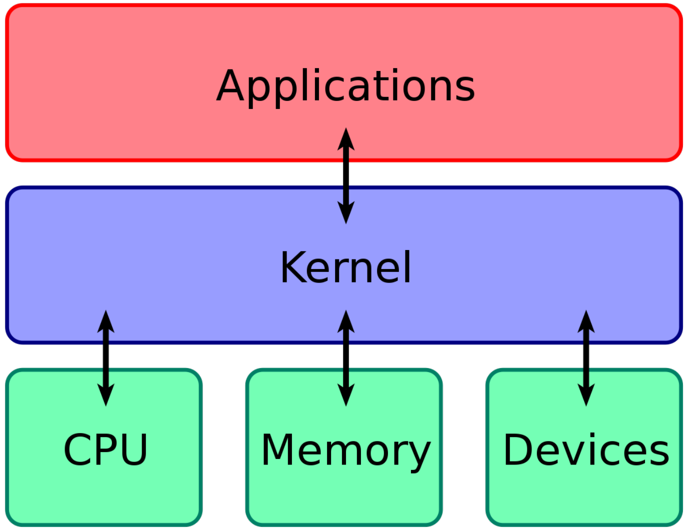
- End User's View
  - Graphical or Command-line User Interface
  - Filesystem and Directory Structure

- Program's View
  - HW Resource Manager
  - Illusion Maker (Dedicated HW for each application)

- Programmer's View
  - Set of Library Functions and System Calls

Inside library files

Directly provided by the OS kernel

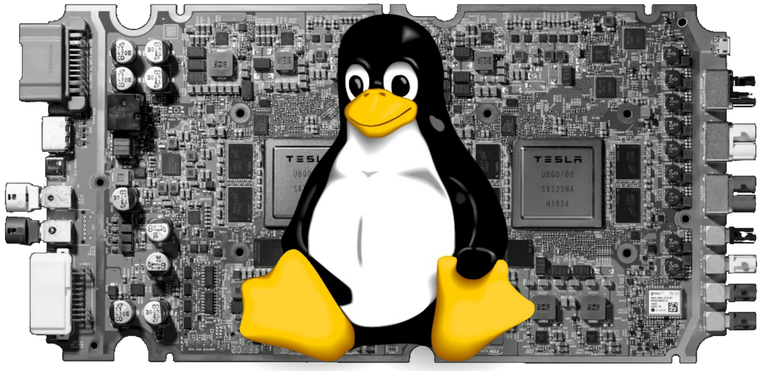


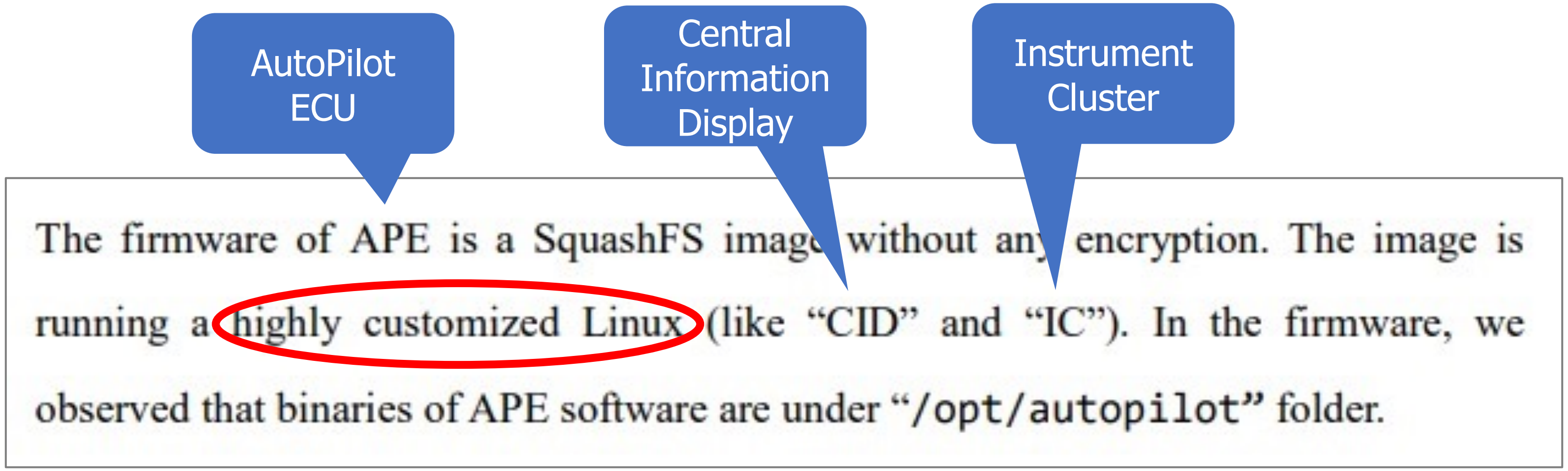
The core component of an OS

Our Focus: Linux Operating System

- The most widely used operating system
- Tesla Autopilot is based on Linux
- Standard OS for developers

Linux Is Not UniX





AutoPilot ECU

Central Information Display

Instrument Cluster

The firmware of APE is a SquashFS image without any encryption. The image is running a highly customized Linux (like "CID" and "IC"). In the firmware, we observed that binaries of APE software are under "`/opt/autopilot`" folder.

Source: Experimental Security Research of Tesla Autopilot (Tencent Keen Security Lab) 2019.3

Birth of UNIX (1969)

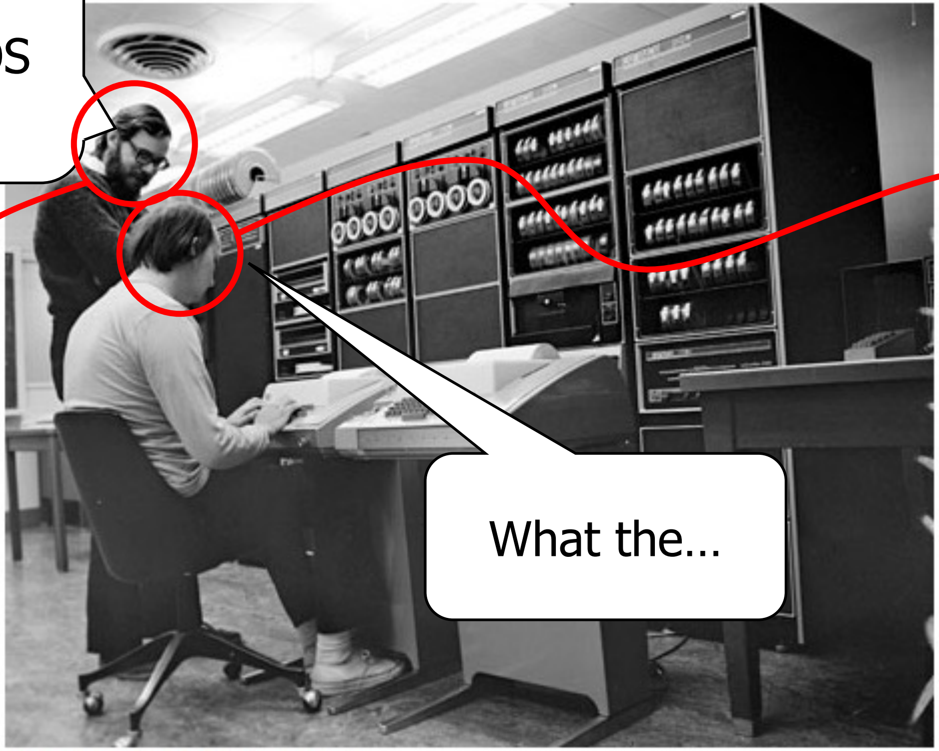


Photo: Alcatel-Lucent

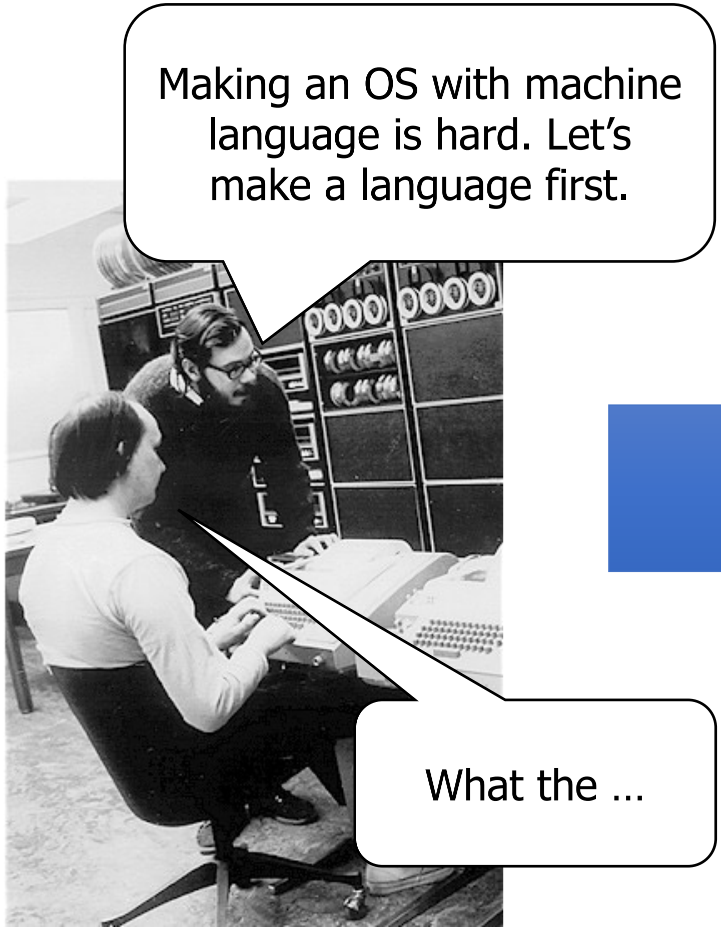
Dennis Ritchie
1941 - 2011

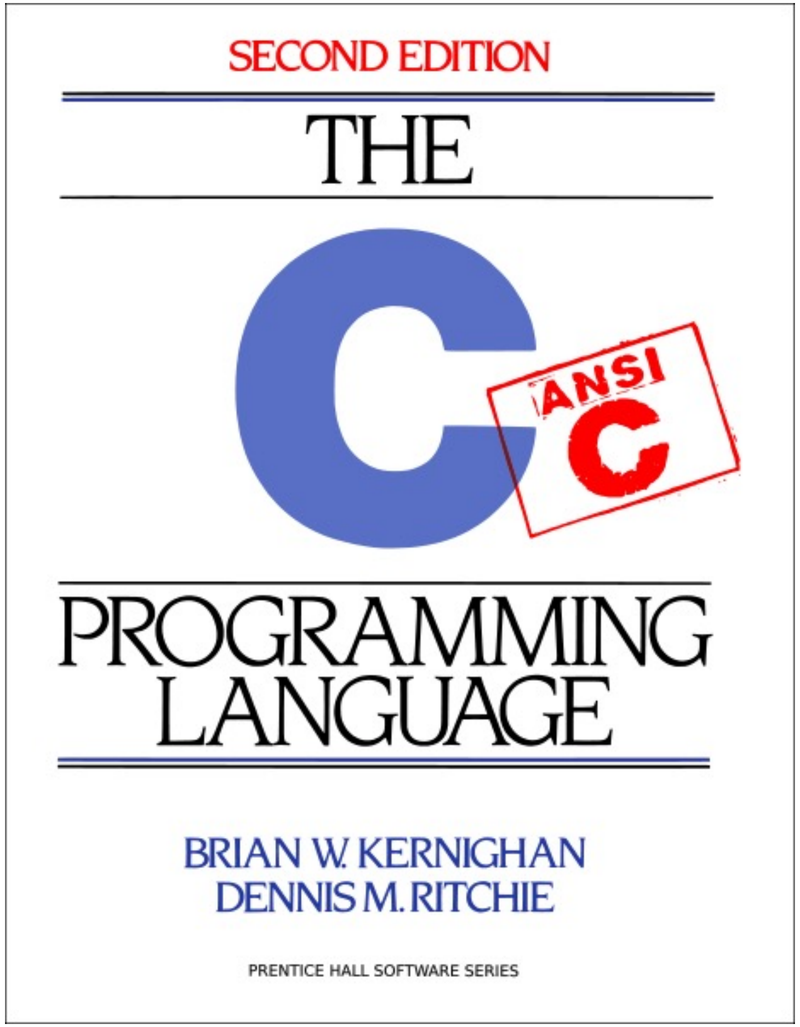
Dennis Ritchie & Ken Thompson
at
AT&T Bell Lab.

Photo: Gabriela Hasbun

Proud father: Ken Thompson poses to show off his brainchild, 40 years after its birth.

Birth of C Programming Language (1972)


Making an OS with machine language is hard. Let's make a language first.
What the …



SECOND EDITION

THE

C

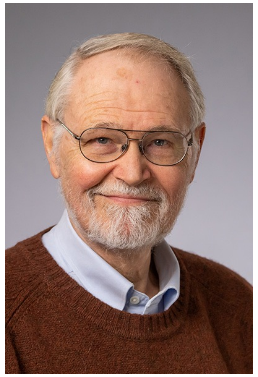
ANSI C

PROGRAMMING LANGUAGE

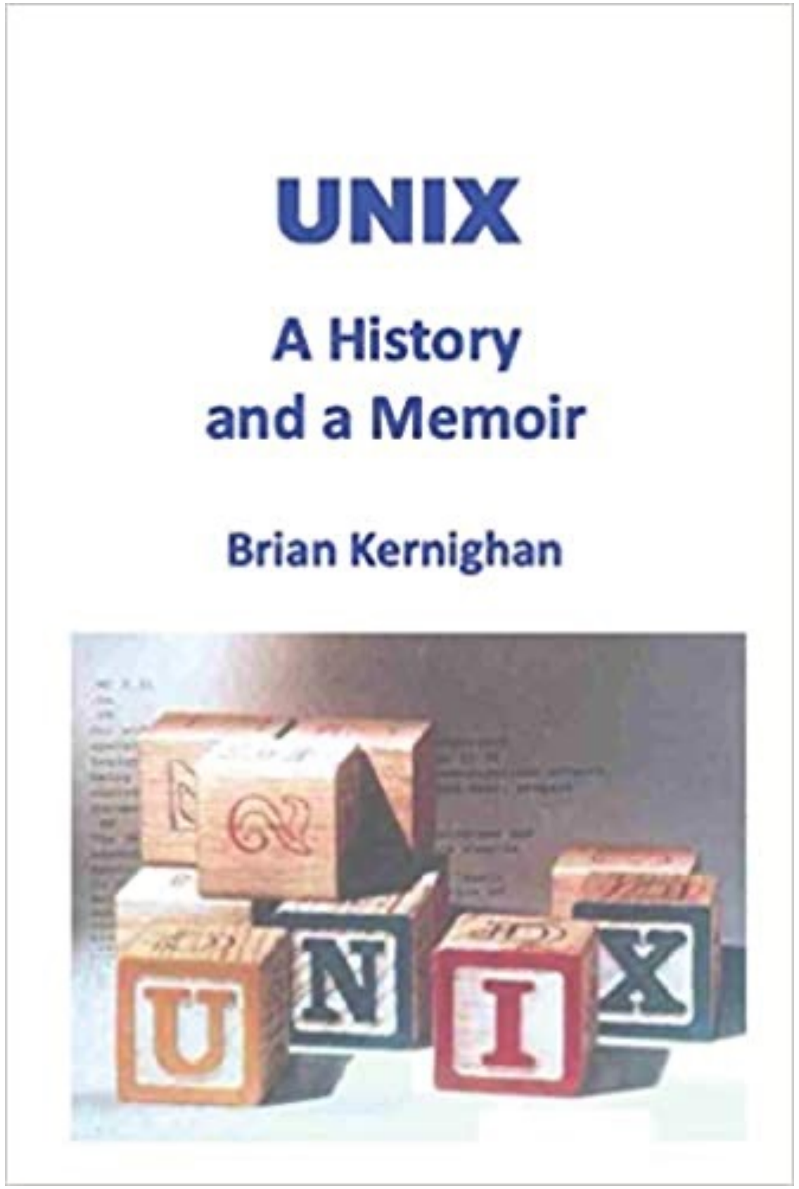
BRIAN W. KERNIGHAN
DENNIS M. RITCHIE

PRENTICE HALL SOFTWARE SERIES

A History



Brian Kernighan

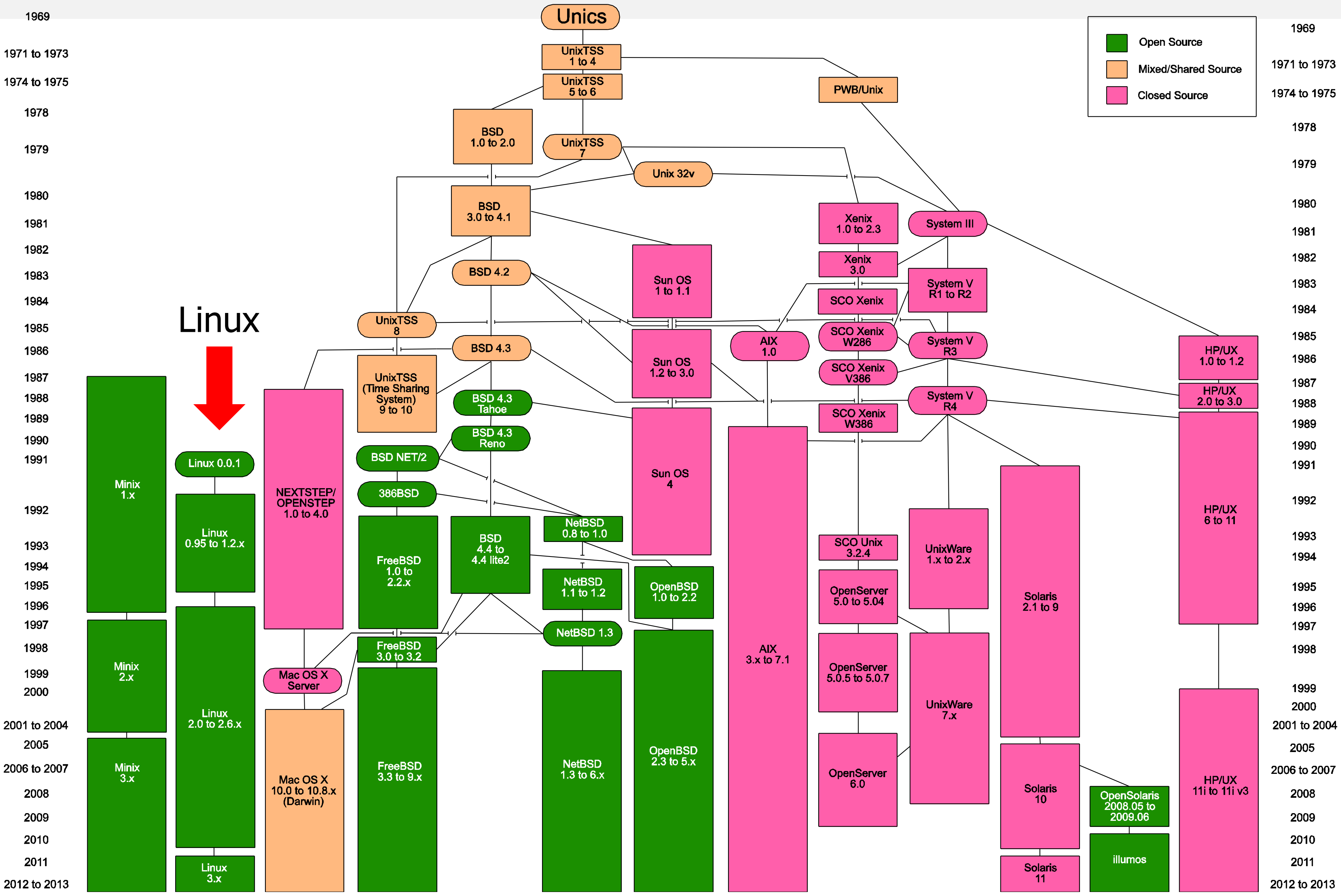


UNIX

A History
and a Memoir

Brian Kernighan

UNIX Family Tree



🟩 Open Source

🟧 Mixed/Shared Source

🟥 Closed Source

Linux

1969
1971 to 1973
1974 to 1975
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001 to 2004
2005
2006 to 2007
2008
2009
2010
2011
2012 to 2013

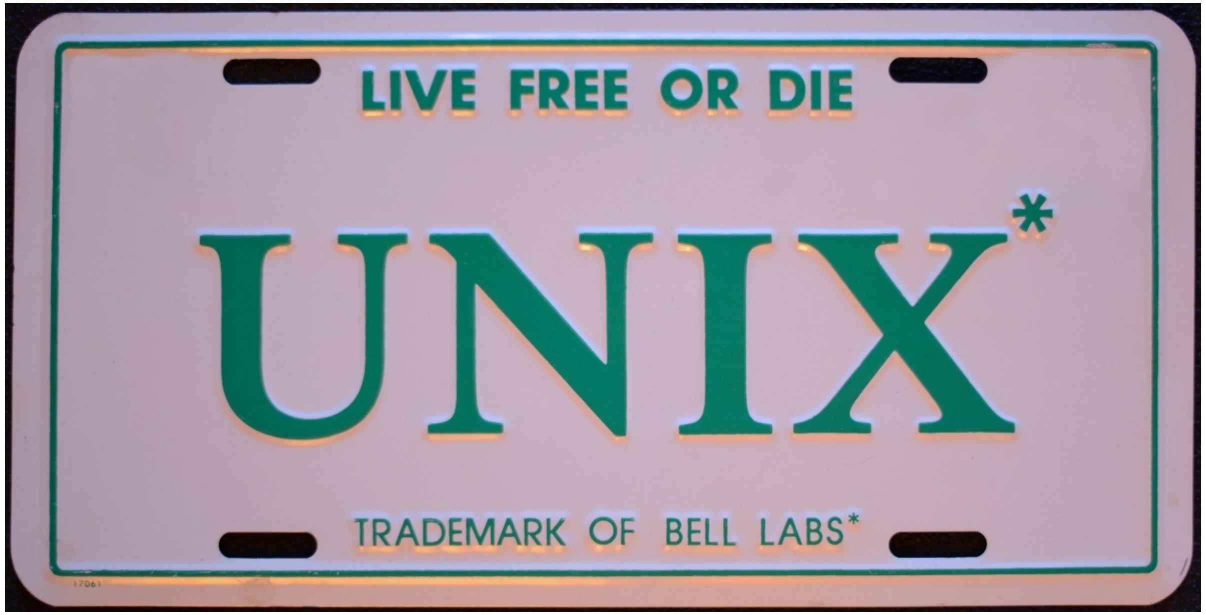
Unics
UnixTSS 1 to 4
UnixTSS 5 to 6
PWB/Unix
BSD 1.0 to 2.0
UnixTSS 7
Unix 32v
BSD 3.0 to 4.1
Xenix 1.0 to 2.3
System III
Sun OS 1 to 1.1
Xenix 3.0
BSD 4.2
System V R1 to R2
SCO Xenix
UnixTSS 8
SCO Xenix W286
System V R3
AIX 1.0
HP/UX 1.0 to 1.2
BSD 4.3
Sun OS 1.2 to 3.0
SCO Xenix V386
UnixTSS (Time Sharing System) 9 to 10
HP/UX 2.0 to 3.0
BSD 4.3 Tahoe
System V R4
SCO Xenix W386
BSD 4.3 Reno
Linux 0.0.1
BSD NET/2
Minix 1.x
NEXTSTEP/ OPENSTEP 1.0 to 4.0
Sun OS 4
386BSD
HP/UX 6 to 11
NetBSD 0.8 to 1.0
Linux 0.95 to 1.2.x
BSD 4.4 to 4.4 lite2
FreeBSD 1.0 to 2.2.x
SCO Unix 3.2.4
UnixWare 1.x to 2.x
Solaris 2.1 to 9
NetBSD 1.1 to 1.2
OpenBSD 1.0 to 2.2
OpenServer 5.0 to 5.04
NetBSD 1.3
FreeBSD 3.0 to 3.2
Minix 2.x
Mac OS X Server
AIX 3.x to 7.1
OpenServer 5.0.5 to 5.0.7
UnixWare 7.x
Linux 2.0 to 2.6.x
Mac OS X 10.0 to 10.8.x (Darwin)
FreeBSD 3.3 to 9.x
NetBSD 1.3 to 6.x
OpenBSD 2.3 to 5.x
Minix 3.x
OpenServer 6.0
Solaris 10
OpenSolaris 2008.05 to 2009.06
HP/UX 11i to 11i v3
illumos
Linux 3.x
Solaris 11

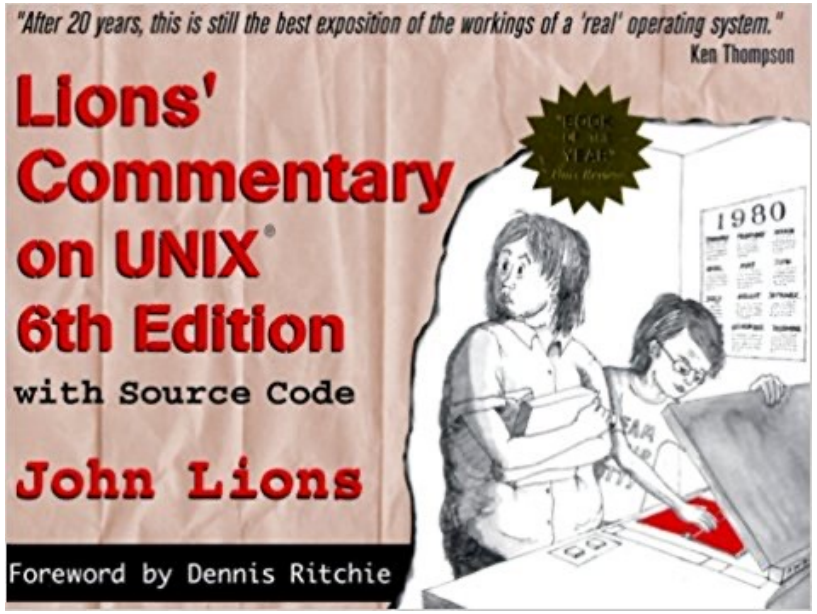
1969
1971 to 1973
1974 to 1975
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001 to 2004
2005
2006 to 2007
2008
2009
2010
2011
2012 to 2013

https://en.wikipedia.org/wiki/History_of_Unix

UNIX Dark Age

- Unix source code had been open for classroom use until 1979
- From then, AT&T closed it for profit by selling it
- Lion’s book (1976) had been illegally used to study UNIX




"After 20 years, this is still the best exposition of the workings of a 'real' operating system."
Ken Thompson
Lions' Commentary on UNIX® 6th Edition
with Source Code
John Lions
Foreword by Dennis Ritchie
1980

GNU (GNU's Not Unix)

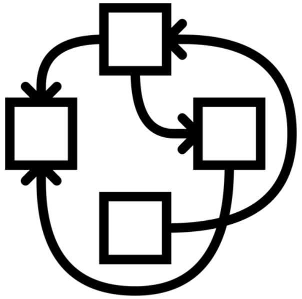
- Launched by Richard Matthew Stallman aka RMS at MIT in 1983
- The goal was to develop a UNIX-compatible free operating system
- GPL (GNU General Public License)
- Famous software packages (e.g., GCC, GDB, Make, GNU C Library, …)
- By the early 1990s, everything except its kernel had been developed

Infectious

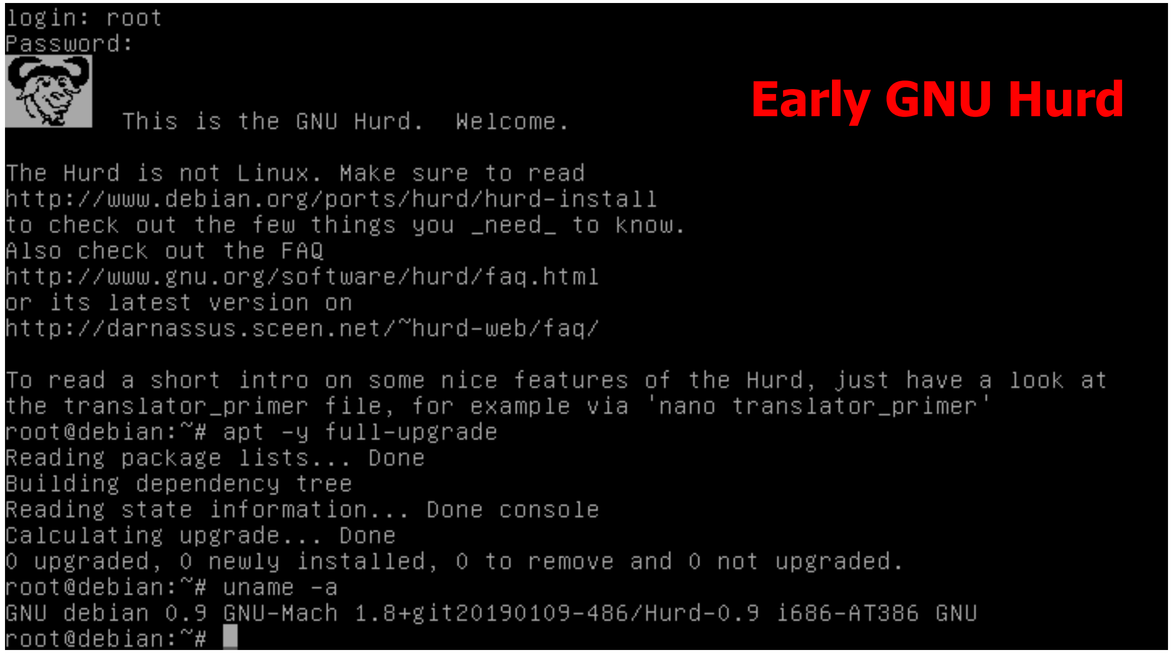
Freedom, not about the price

Richard Stallman
(Source: Wikipedia)

GNU Project



Hurd (Kernel)



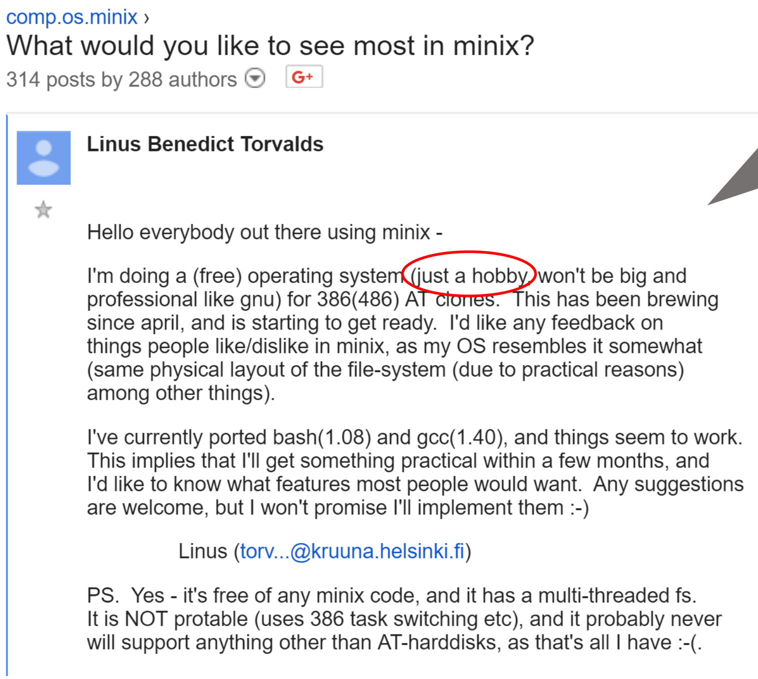
Early GNU Hurd

```
login: root
Password:

        This is the GNU Hurd.  Welcome.

The Hurd is not Linux. Make sure to read
http://www.debian.org/ports/hurd/hurd-install
to check out the few things you _need_ to know.
Also check out the FAQ
http://www.gnu.org/software/hurd/faq.html
or its latest version on
http://darnassus.sceen.net/~hurd-web/faq/

To read a short intro on some nice features of the Hurd, just have a look at
the translator_primer file, for example via 'nano translator_primer'
root@debian:~# apt -y full-upgrade
Reading package lists... Done
Building dependency tree
Reading state information... Done console
Calculating upgrade... Done
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
root@debian:~# uname -a
GNU debian 0.9 GNU-Mach 1.8+git20190109-486/Hurd-0.9 i686-AT386 GNU
root@debian:~# 
```

Birth of Linux Kernel (1991~)

- GPLed UNIX kernel began in 1991 by a Finnish student, Linus Torvalds
- Became much more popular than the GNU Hurd kernel
- Has grown up to 27.8 million lines of code (2020)

Version Number



comp.os.minix ›
What would you like to see most in minix?
314 posts by 288 authors

Linus Benedict Torvalds

Hello everybody out there using minix -

I'm doing a (free) operating system (just a hobby, won't be big and professional like gnu) for 386(486) AT clones. This has been brewing since april, and is starting to get ready. I'd like any feedback on things people like/dislike in minix, as my OS resembles it somewhat (same physical layout of the file-system (due to practical reasons) among other things).

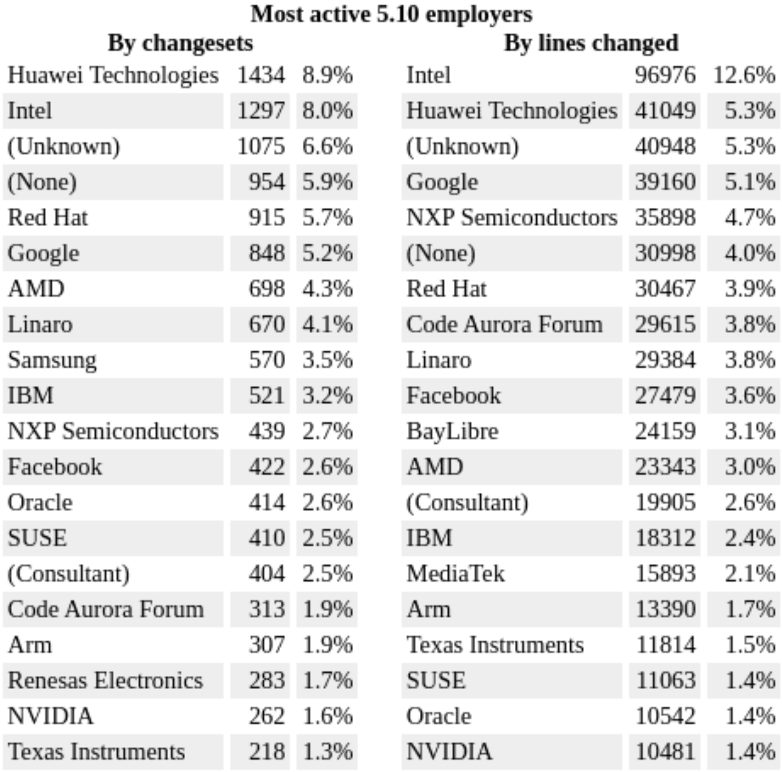
I've currently ported bash(1.08) and gcc(1.40), and things seem to work. This implies that I'll get something practical within a few months, and I'd like to know what features most people would want. Any suggestions are welcome, but I won't promise I'll implement them :-)

Linus (torv...@kruuna.helsinki.fi)

PS. Yes - it's free of any minix code, and it has a multi-threaded fs. It is NOT protable (uses 386 task switching etc), and it probably never will support anything other than AT-harddisks, as that's all I have :-(.

Linus's first email announcing Linux to comp.os.minix

Linus Torvalds (1969 ~)



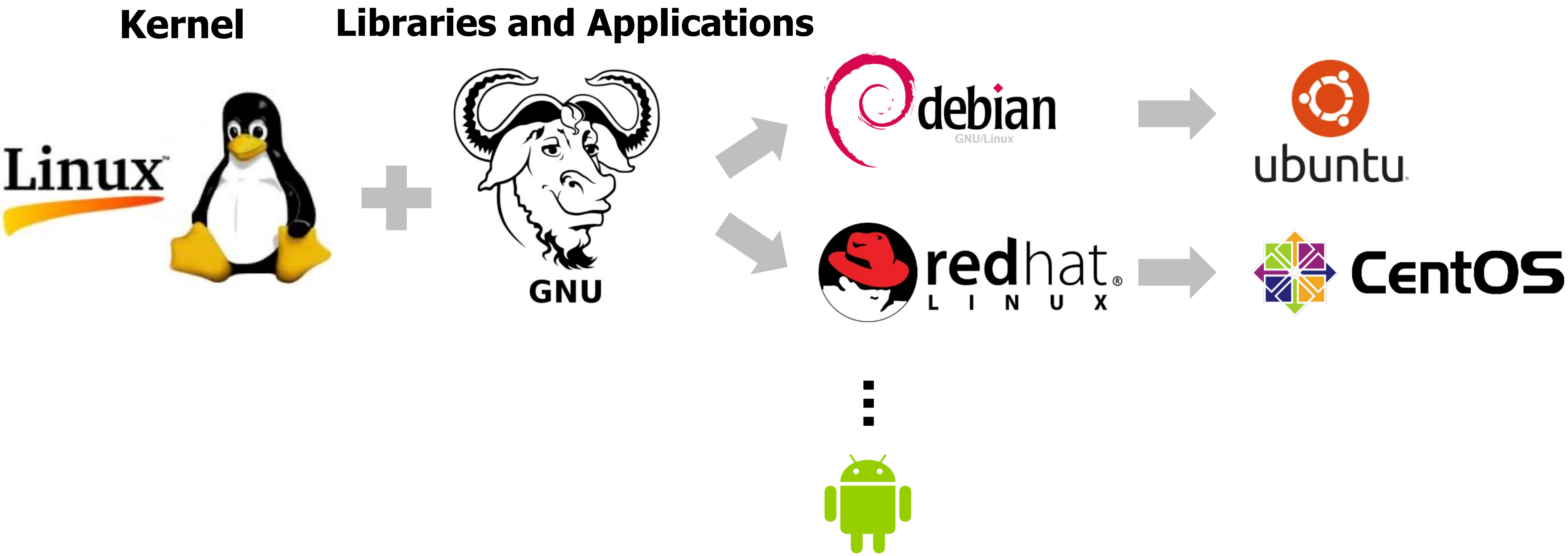
Most active 5.10 employers

By changesets			By lines changed		
Huawei Technologies	1434	8.9%	Intel	96976	12.6%
Intel	1297	8.0%	Huawei Technologies	41049	5.3%
(Unknown)	1075	6.6%	(Unknown)	40948	5.3%
(None)	954	5.9%	Google	39160	5.1%
Red Hat	915	5.7%	NXP Semiconductors	35898	4.7%
Google	848	5.2%	(None)	30998	4.0%
AMD	698	4.3%	Red Hat	30467	3.9%
Linaro	670	4.1%	Code Aurora Forum	29615	3.8%
Samsung	570	3.5%	Linaro	29384	3.8%
IBM	521	3.2%	Facebook	27479	3.6%
NXP Semiconductors	439	2.7%	BayLibre	24159	3.1%
Facebook	422	2.6%	AMD	23343	3.0%
Oracle	414	2.6%	(Consultant)	19905	2.6%
SUSE	410	2.5%	IBM	18312	2.4%
(Consultant)	404	2.5%	MediaTek	15893	2.1%
Code Aurora Forum	313	1.9%	Arm	13390	1.7%
Arm	307	1.9%	Texas Instruments	11814	1.5%
Renesas Electronics	283	1.7%	SUSE	11063	1.4%
NVIDIA	262	1.6%	Oracle	10542	1.4%
Texas Instruments	218	1.3%	NVIDIA	10481	1.4%

Recent Contributions from Big Companies

GNU/Linux Operating Systems

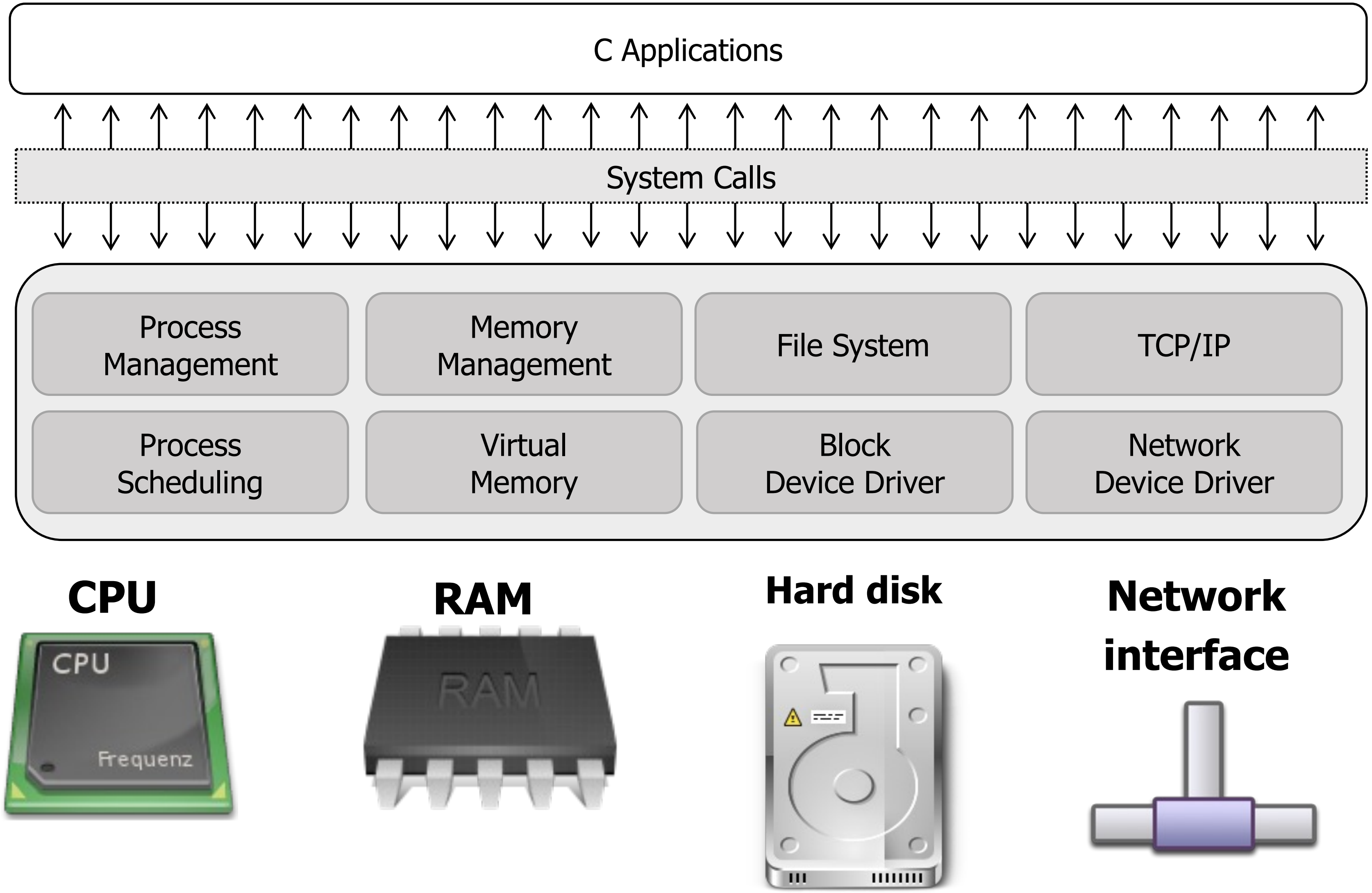
- Many companies and organizations develop their own **GNU/Linux operating system** based on the Linux kernel and the GNU project



Kernel

Libraries and Applications

Operating System Kernel



C Applications

System Calls

Process Management

Memory Management

File System

TCP/IP

Process Scheduling

Virtual Memory

Block Device Driver

Network Device Driver

CPU

RAM

Hard disk

Network interface

Software Layers and Control Flows

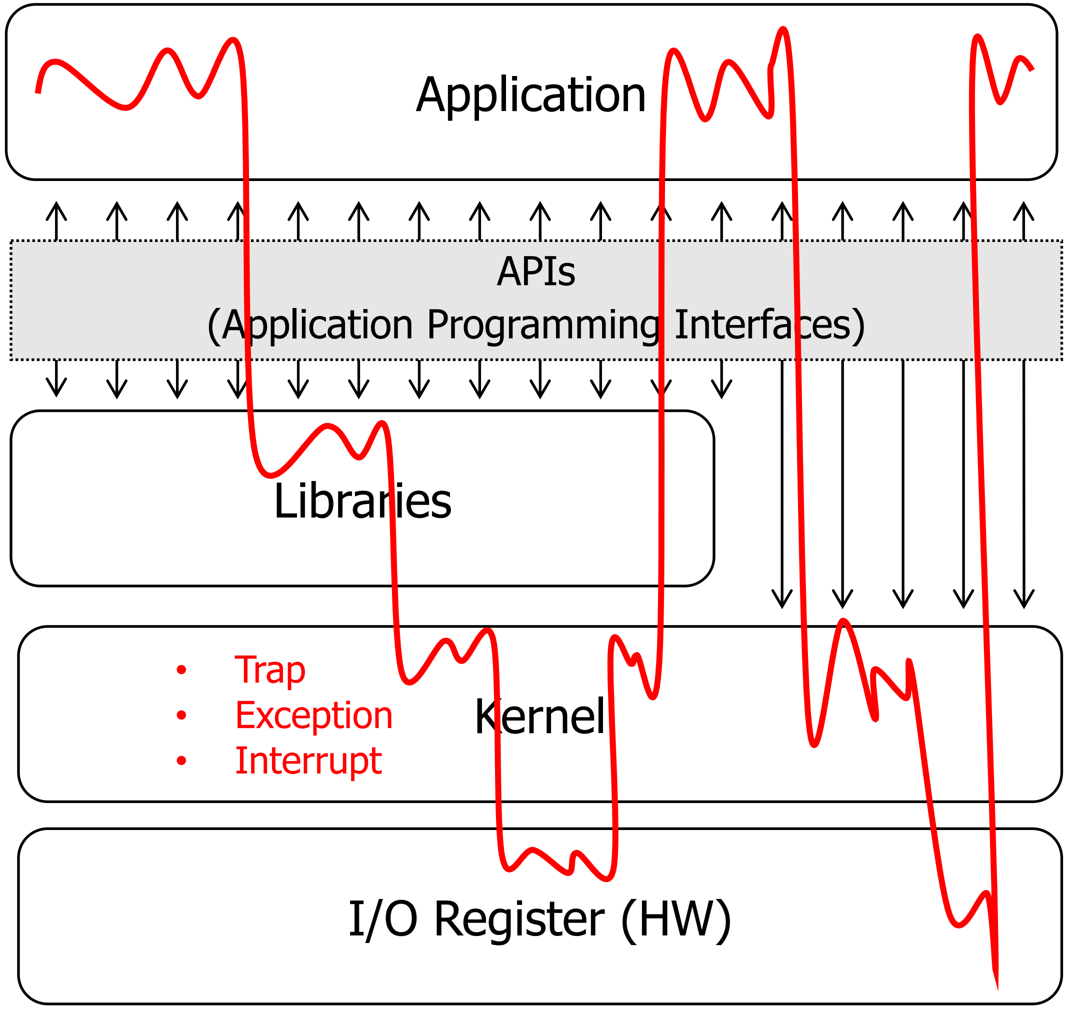
- Limited access to address space
- Cannot access hardware registers directly

User Level

CPU mode changes between user mode and kernel mode (privileged mode)

Kernel Level

- Unlimited access to address space
- Can directly handle hardware registers



Library Function vs System Call

```
#include <stdio.h>

int main(void)
{
    printf("Hello World\n");
    return 0;
}
```

Library function

- Library Functions
  - Serviced in user level
  - May call system calls inside

```
#include <unistd.h>

int main(void)
{
    write(STDOUT_FILENO, "Hello World\n", 12);
    return 0;
}
```

System call

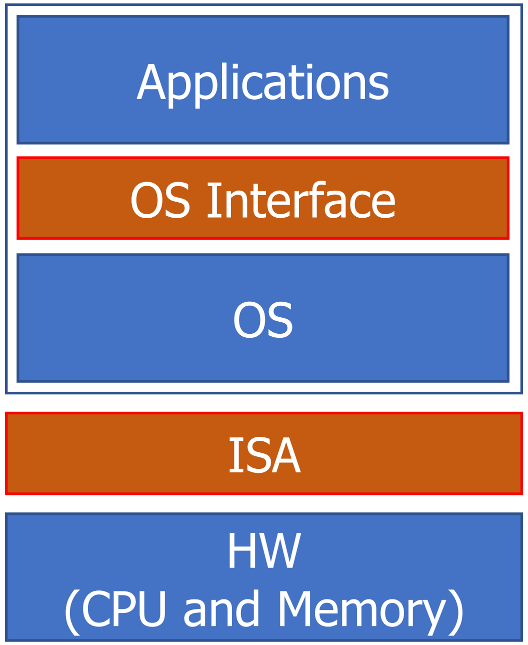
- System Calls
  - Serviced inside the kernel

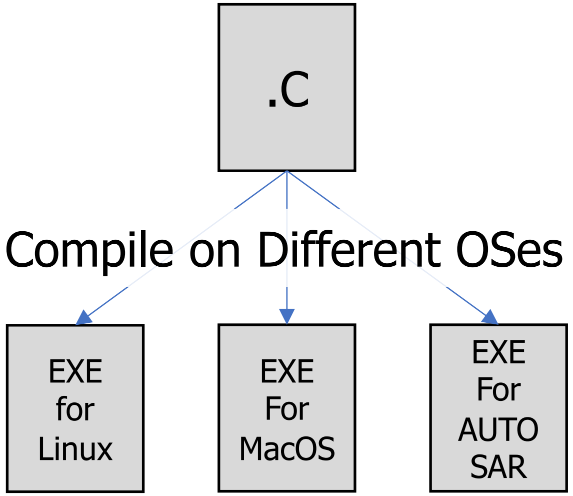
User Mode / Kernel Mode Transition

- Trap (Synchronous)
  - Calling a system call
- Exception (Synchronous)
  - Jump to an exception handler in kernel
  - E.g.) divide by zero
- Interrupt (Asynchronous)
  - Jump to an interrupt handler in kernel
  - E.g.) Hardware interrupt

OS Interfaces

- Standards for library functions and system calls (or APIs)
  - Applications are "portable" across different OSes with the same OS interface
  - Source-level portability, not binary-level portability





Single Source Code for Multiple OSes

POSIX Standards

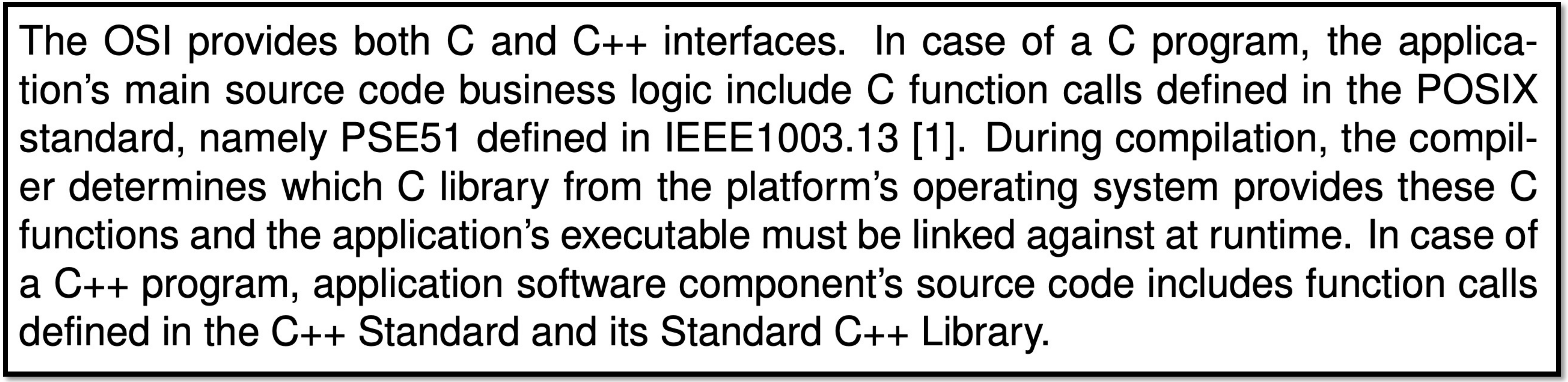
- Portable Operating System Interface (POSIX)
  - The most popular OS interface for UNIX-like OSes

- 1003.1-2017 - IEEE Standard for Information Technology--Portable Operating System Interface (POSIX(TM)) Base Specifications, Issue 7
  - https://ieeexplore.ieee.org/document/8277153
  - Baseline standard

- 1003.13-2003 - IEEE Standard for Information Technology - Standardized Application Environment Profile (AEP) - POSIX(TM) Realtime and Embedded Application Support
  - https://ieeexplore.ieee.org/document/1342418
  - Minimal standard (subsets) for real-time embedded applications

Adaptive AUTOSAR based on POSIX 1003.13

- AUTOSAR Adaptive Platform
  - Platform software standards for autonomous driving systems
  - Based on POSIX 1003.13



The OSI provides both C and C++ interfaces. In case of a C program, the application's main source code business logic include C function calls defined in the POSIX standard, namely PSE51 defined in IEEE1003.13 [1]. During compilation, the compiler determines which C library from the platform's operating system provides these C functions and the application's executable must be linked against at runtime. In case of a C++ program, application software component's source code includes function calls defined in the C++ Standard and its Standard C++ Library.

Specification of Operating System Interface for AUTOSAR Adaptive Platform

Summary

- Firmware vs. OS-based Systems
- History of UNIX and Linux Operating Systems
- The Kernel Concept and System Calls
- User Mode vs. Kernel Mode
- POSIX Standard