SYNOPSYS®

Things We Love About Unix

Alan Dewar, Senior Staff Engineer, Synopsys

2024-02-27

Outline

- History
- Philosophy
- OS Comparisons
- Features
- Your Views

History

Multics

- Multiplexed Information and Computing Service
- 1964: initial planning, development
- MIT, General Electric, Bell Labs
- Designed for security
- Time sharing
- Dynamic linking
- Hierarchical file system
- User-level command processor
- High-level systems language (PL/I)
- Active functions

History

Unix

- Originally Unics
  - Uniplexed Information and Computing Service
- Bell Labs withdraws from Multics
- Ken Thompson
  - Space Travel program
  - Operating system design
- Dennis Ritchie
  - C
- Doug McIlroy
  - Pipes
  - Various Unix tools
  - Manual

History

Initial Distribution

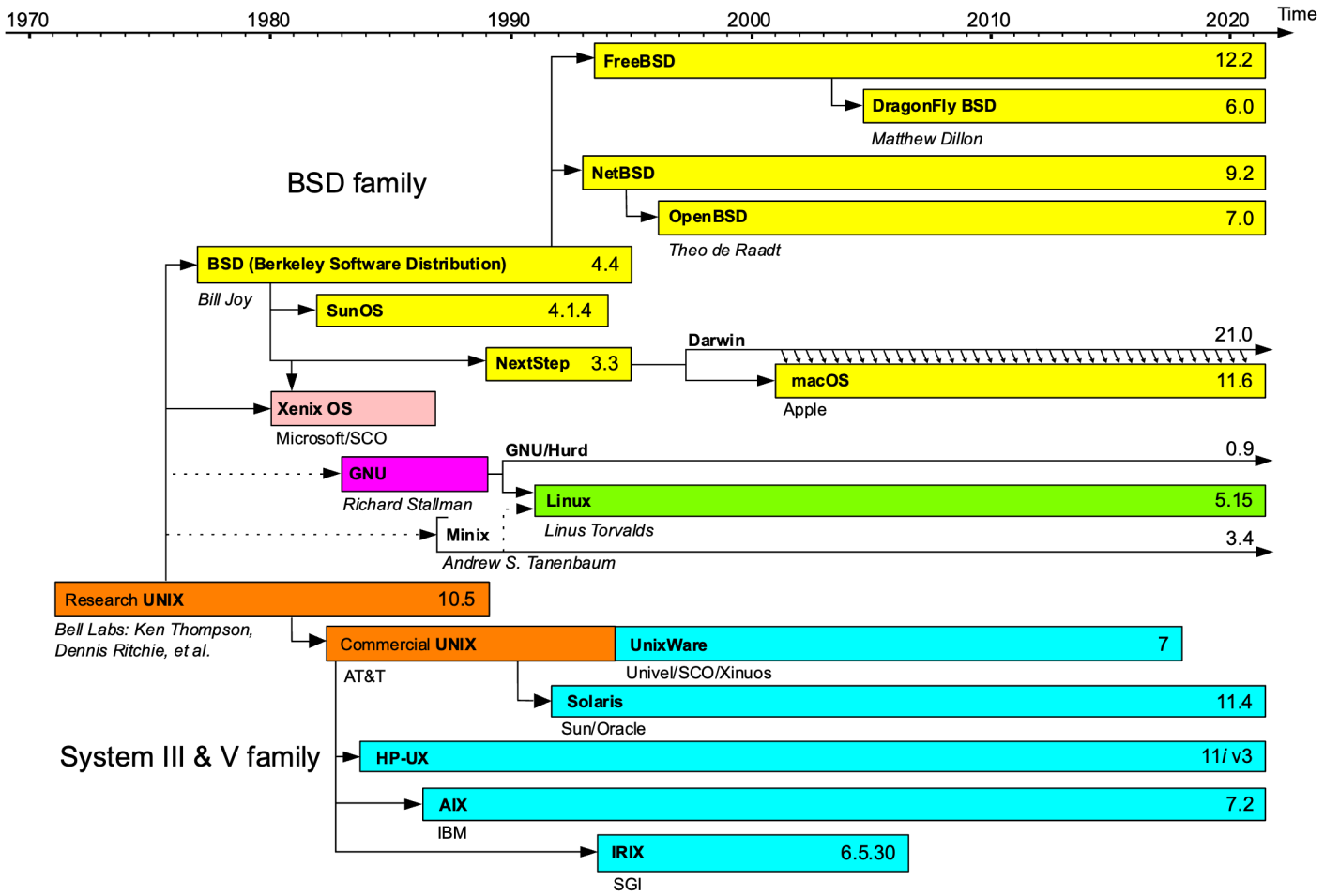
- AT&T antitrust consent decree (1954)
  - Not allowed to enter computer business
  - Required to license non-telephone technology
- CACM paper (1974)
- High demand from universities and research labs

History

Unix Wars

History

Unix Wars



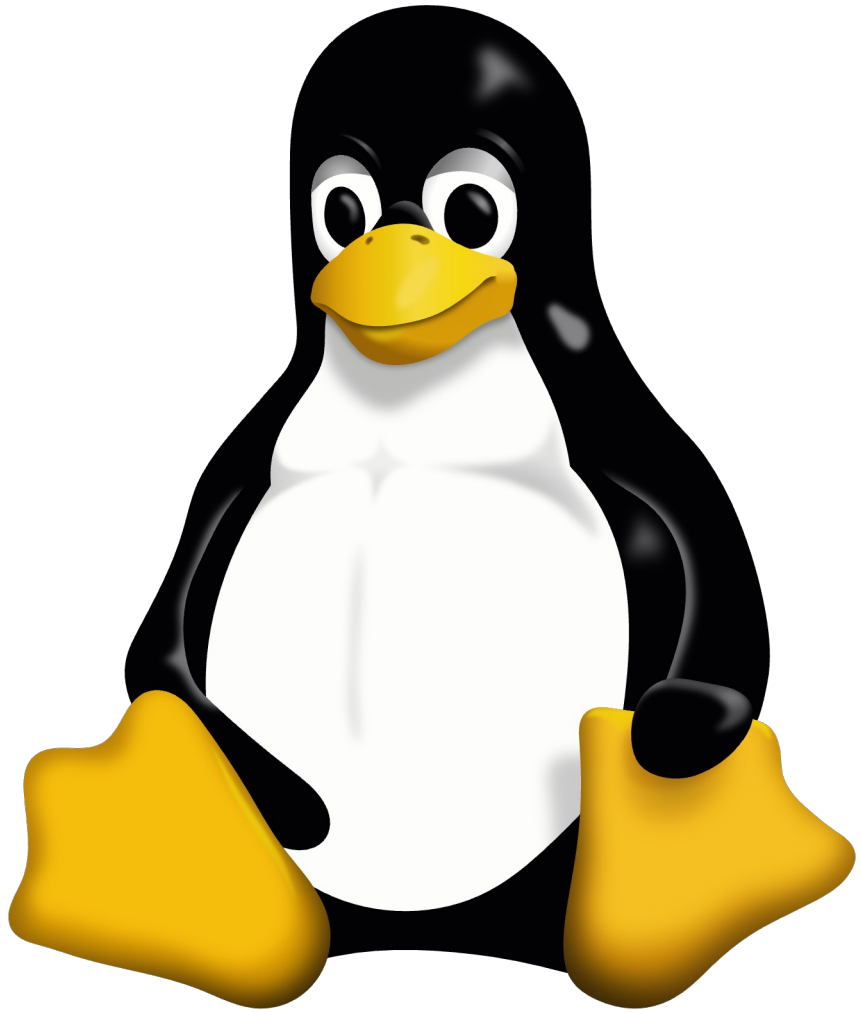
History

Unix Wars

- Berkeley Software Distribution (BSD)
  - Computer Systems Research Group (CSRG) at the University of California, Berkeley
  - Based on the original Unix source code
  - Ken Thompson 1975 sabbatical as visiting professor
- Bell System antitrust break-up
  - Unix System V
- Productization, differentiation lead to end of sharing
- Richard Stallman
  - Free Software Foundation, GNU manifesto
- 1985: POSIX standards (IEEE)
- Larry Wall
  - patch, perl

Modern Unix

- *BSD
- macOS
- Linux
- Solaris
- HP-UX
- AIX
- Windows
  - Windows Subsystem for Linux (WSL)
  - cygwin
  - RMS: “a step backward in the campaign for freedom”



Calgary Unix Users' Group (CUUG)

- Early Diamond Sponsors
  - DEC (Digital UNIX)
  - HP (HP-UX)
  - IBM (AIX)
  - Sun (SunOS, Solaris)
- Open Systems
  - Linux
  - OpenBSD

What's in a Name?

- UNIX
  - Trademark
    - AT&T -> Novell -> The Open Group
  - Single Unix Specification (SUS)
    - Extends POSIX
- Unix
  - Generic term
- GNU
  - GNU's Not Unix
- Linux
  - GNU/Linux

Unix Philosophy

Doug McIlroy: Bell System Technical Journal, 1978

- Make each program do one thing well.
  - To do a new job, build afresh rather than complicate old programs by adding new “features”.
- Expect the output of every program to become the input to another, as yet unknown, program.
  - Don't clutter output with extraneous information. Avoid stringently columnar or binary input formats. Don’t insist on interactive input.
- Design and build software, even operating systems, to be tried early, ideally within weeks.
  - Don’t hesitate to throw away the clumsy parts and rebuild them.
- Use tools in preference to unskilled help to lighten a programming task,
  - even if you have to detour to build the tools and expect to throw some of them out after you’ve finished using them.

Unix Philosophy

Doug McIlroy: Summarizing

- Write programs that do one thing and do it well.
- Write programs to work together.
- Write programs to handle text streams, because that is a universal interface.

Unix Philosophy

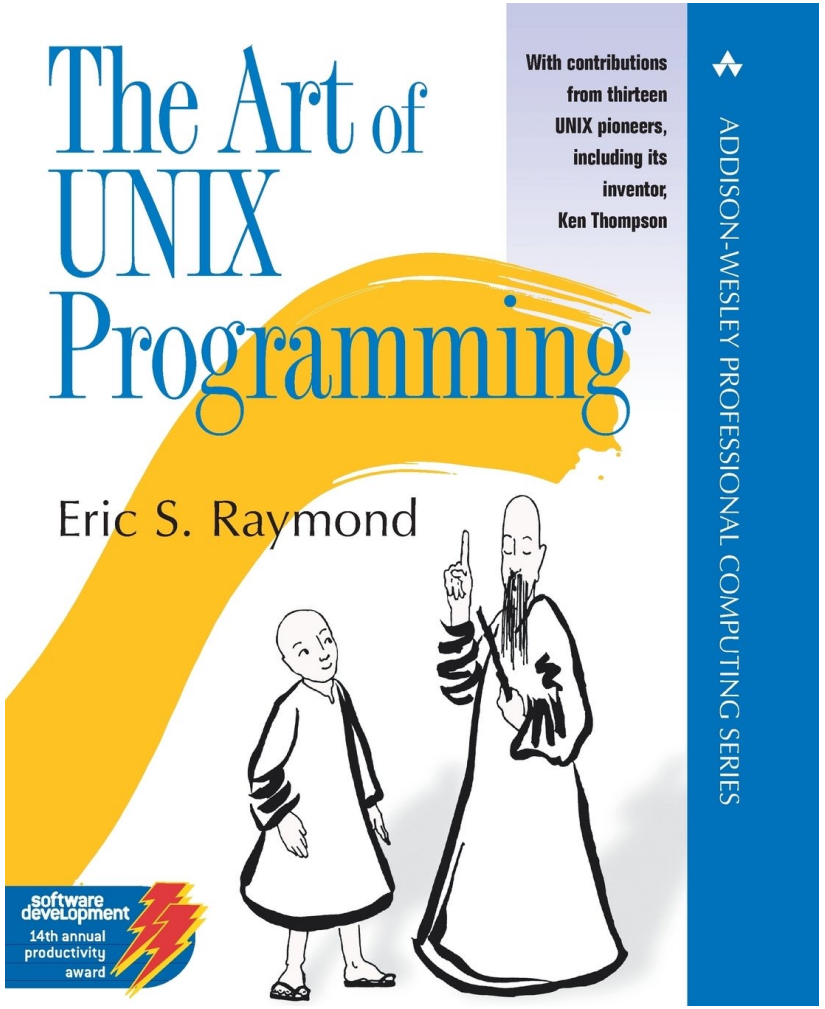
Ritchie and Thompson: CACM, 1974

- Make it easy to write, test, and run programs
- Interactive use instead of batch processing
- Economy and elegance of design due to size constraints (“salvation through suffering”)
- Self-supporting system: all Unix software is maintained under Unix

Unix Philosophy

Eric S. Raymond: The Art of UNIX Programming

- Build modular programs
- Write readable programs
- Use composition
- Separate mechanisms from policy
- Write simple programs
- Write small programs
- Write transparent programs
- Write robust programs
- Make data complicated when required, not the program
- Build on potential users' expected knowledge
- Avoid unnecessary output
- Write programs which fail in a way that is easy to diagnose
- Value developer time over machine time
- Write abstract programs that generate code instead of writing code by hand
- Prototype software before polishing it
- Write flexible and open programs
- Make the program and protocols extensible



OS Comparisons

MacOS (pre-MacOS X)

- Desktop focus
- Single user
- Cooperative multi-tasking
- Single address space

OS Comparisons

MacOS X

- Based on BSD Unix

OS Comparisons

OS/2

- Single user
- Pre-emptive multi-tasking

OS Comparisons

Microsoft: DOS, 3.1, 95, NT, 2000, XP, Server 2003, …

- Pre-emptive multi-tasking
- Expensive spawn
- Special-purpose programs for documents, databases
- Registry
- Incompatible changes
- Backward-compatibility -> loose security
- DLL hell
- Unifying metaphor: “The customer must be locked in”

OS Comparisons

BeOS

- POSIX
- Initially specific hardware
- Lost to Microsoft anti-competitive actions and to Linux

OS Comparisons

MVS

- IBM mainframe batch processing
- JCL
- EBCDIC
- SUS

OS Comparisons

VM/CMS

- IBM
- Hosts many virtual machines

OS Comparisons

Linux

- Ease of install
- Dual-boot
- Other file systems

OS Comparisons

Solaris, AIX, HP-UX

- $$$

Unix Features

Ecosystem

- Multi-platform
  - Especially including inexpensive personal hardware
- Free
- Open
- Stable
- Scalable
- Community support
- Programming tools
- Customizable
- Secure
  - Some more so than others

Unix Features

Technical

- Shell
- Scripting
- Everything is a file
- Networking
- Multiple privilege groups
- Multi-tasking
- Time sharing
- Inter-Process Communication (IPC)

Other UNIX

What Do You Love About Unix?

Image: https://geekfamilylife.com/valentines-day-tutorial-linux-administrator-life/

References

- https://multicians.org/
- https://en.wikipedia.org/wiki/Ken_Thompson
- https://en.wikipedia.org/wiki/Dennis_Ritchie
- https://en.wikipedia.org/wiki/Douglas_McIlroy
- https://en.wikipedia.org/wiki/Unix_wars
- https://en.wikipedia.org/wiki/Windows_Subsystem_for_Linux
- https://en.wikipedia.org/wiki/Unix_philosophy
- http://www.catb.org/esr/writings/taoup/html/

Thank You