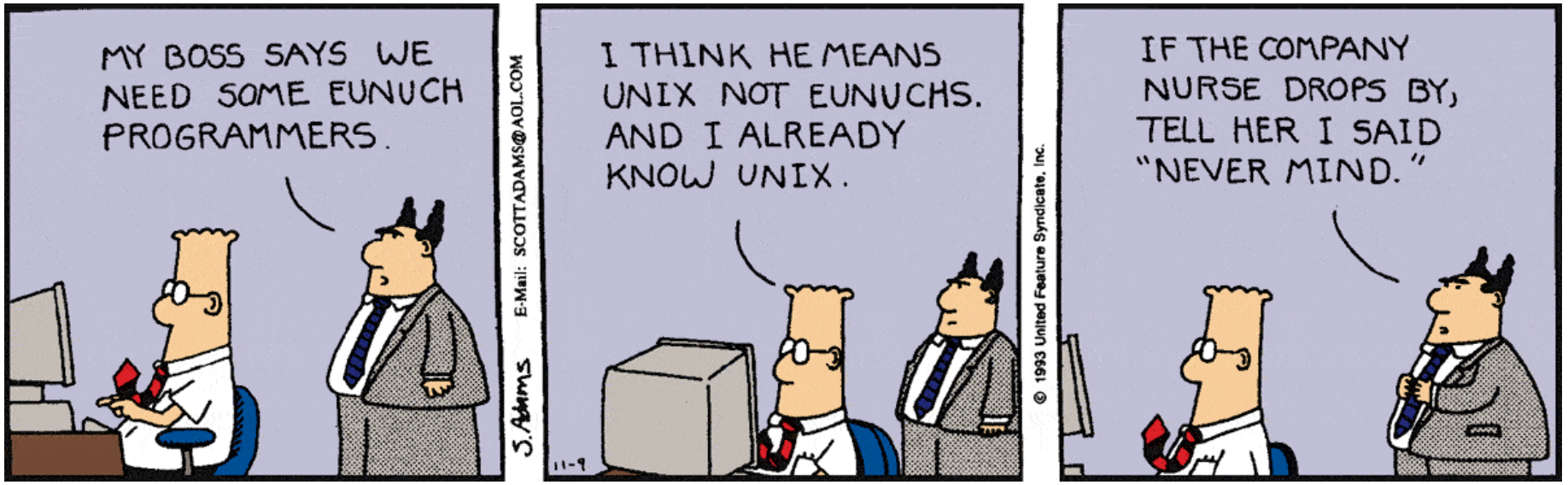
Introduction to UNIX Operating System

CS2600 Systems Programming

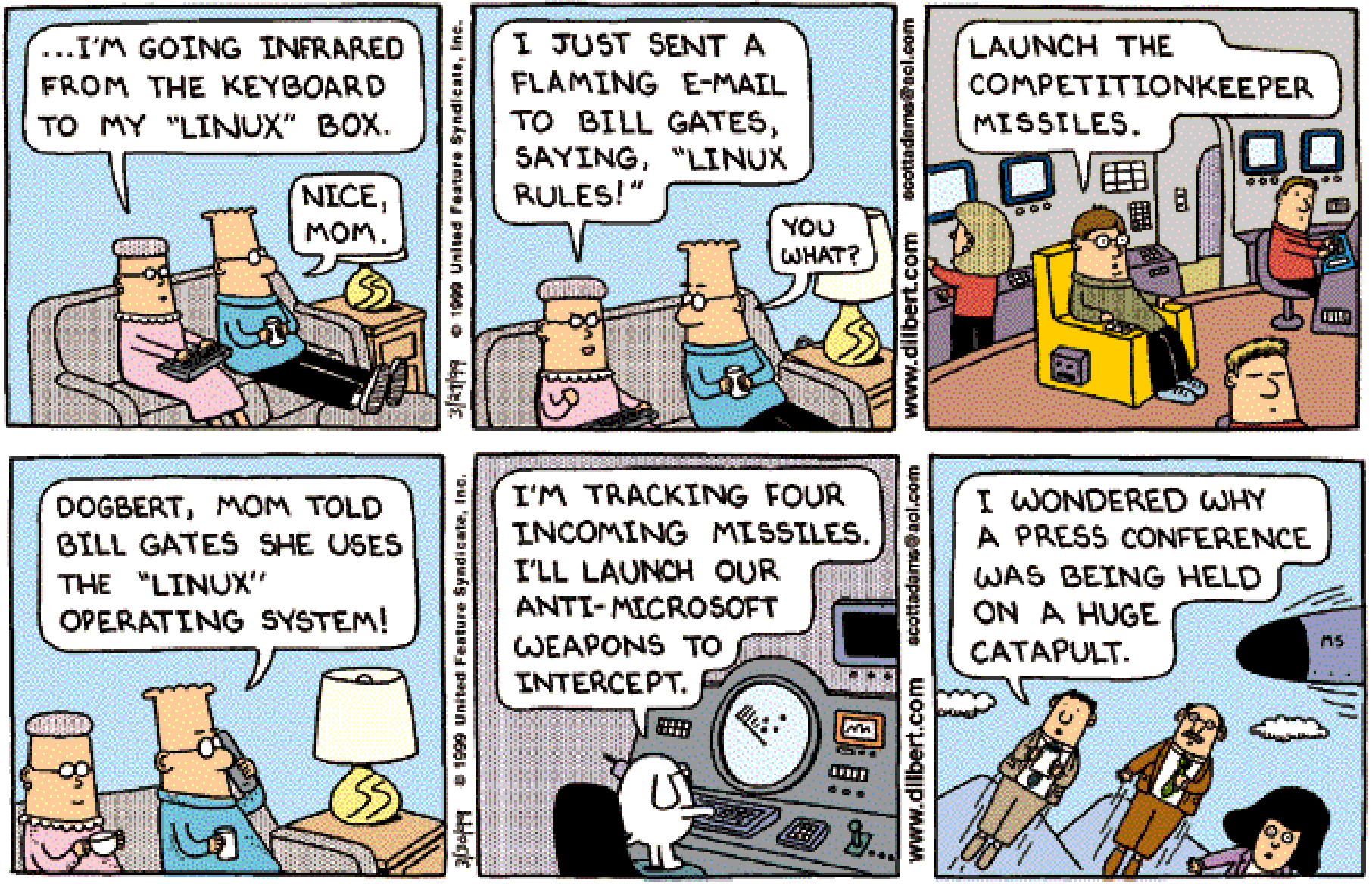
Dr. Amar Raheja

Lecture 2

Lighter Side of UNIX

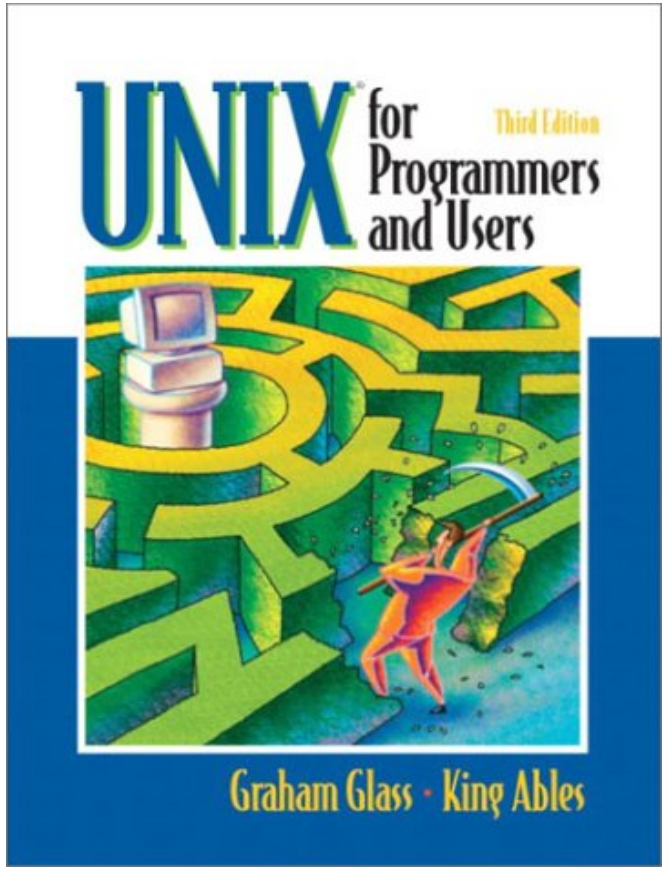


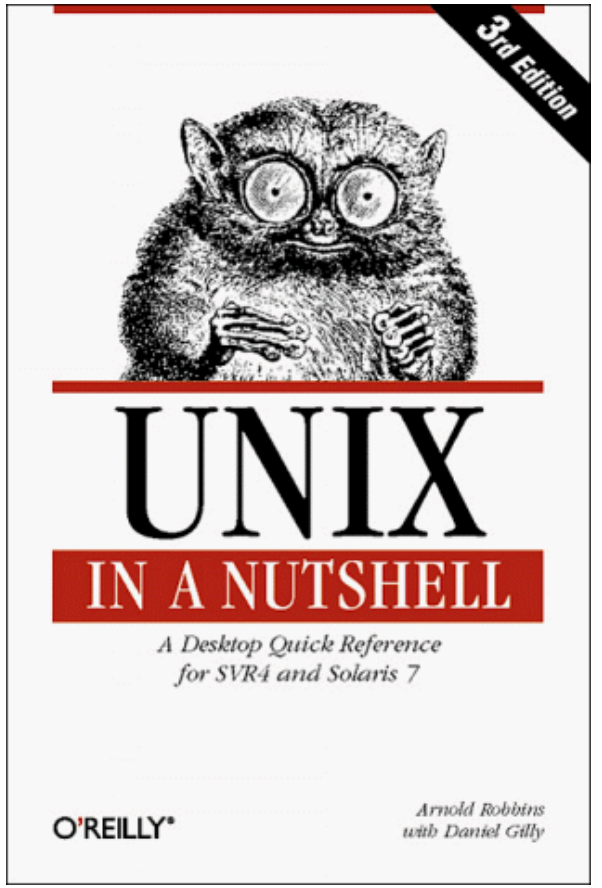
Lighter Side of UNIX (1 of 2)



Lighter Side of UNIX (2 of 2)

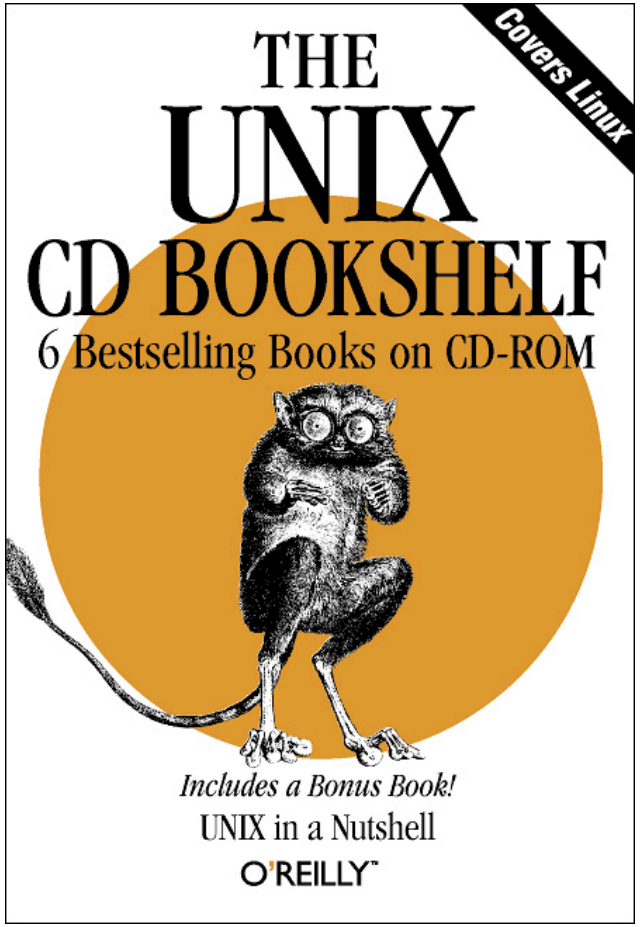
Books for UNIX

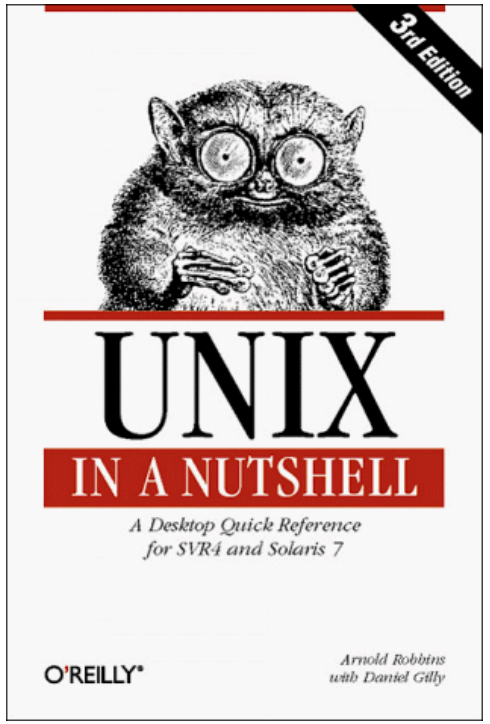


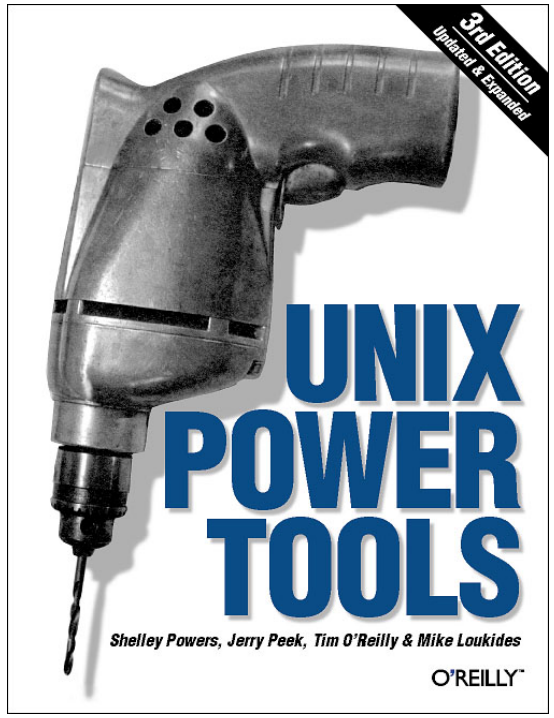


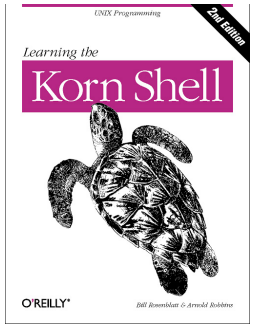
UNIX Books

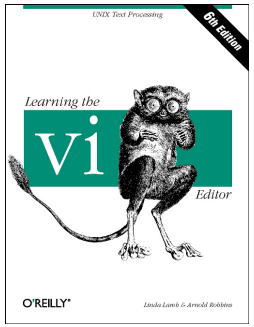
- Available free online through CPP

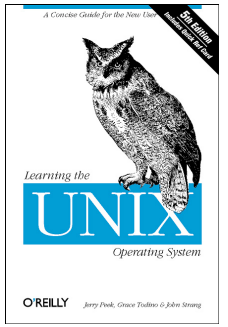


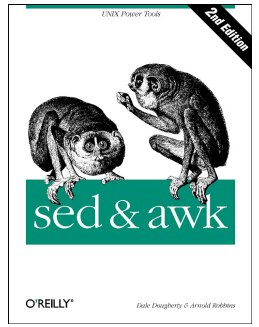








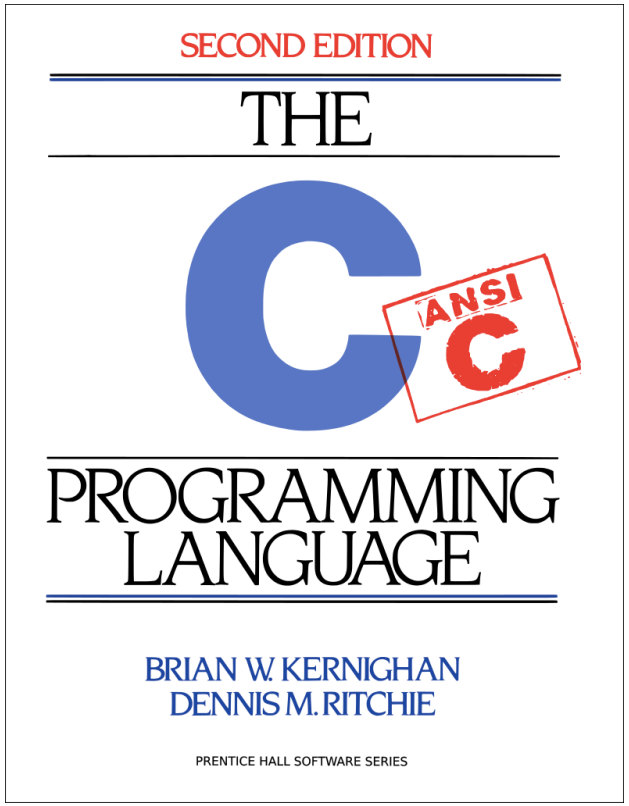


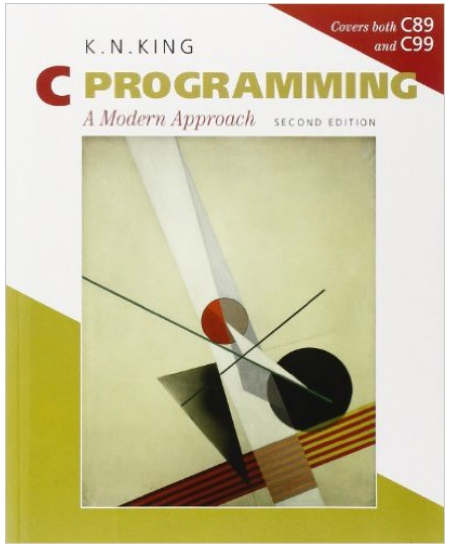


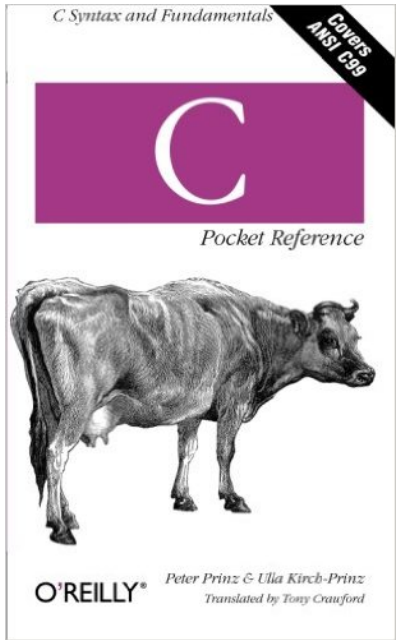
http://proquest.safaribooksonline.com

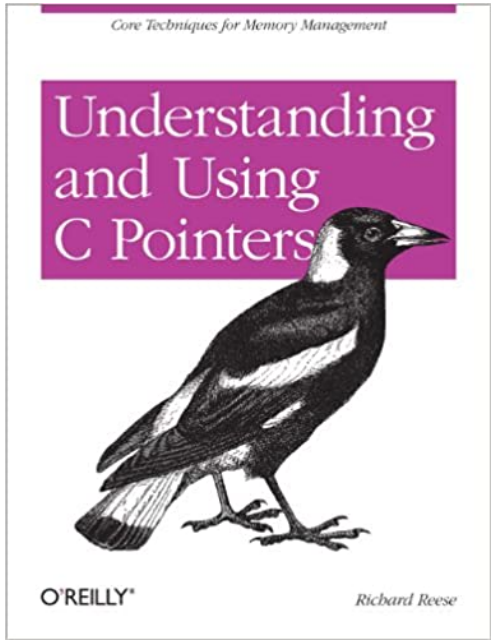
C books

- Ultimate C book



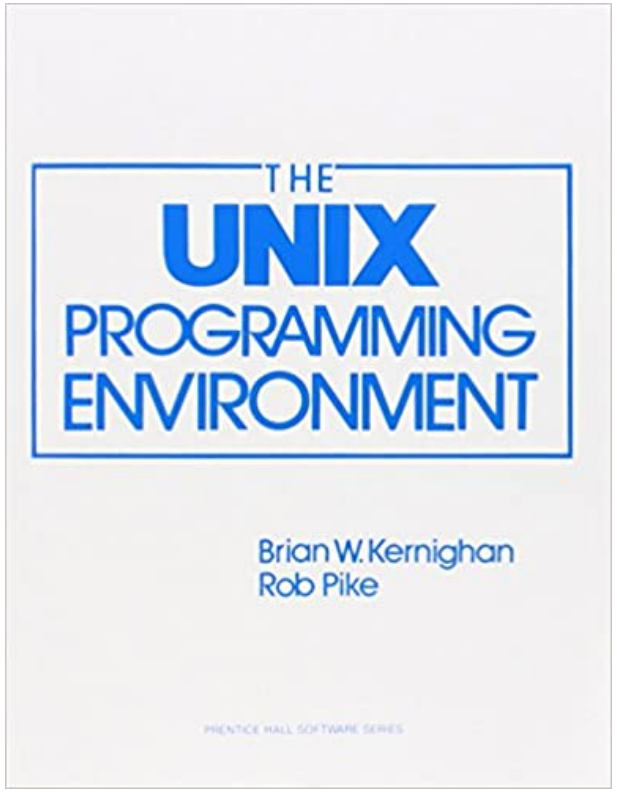


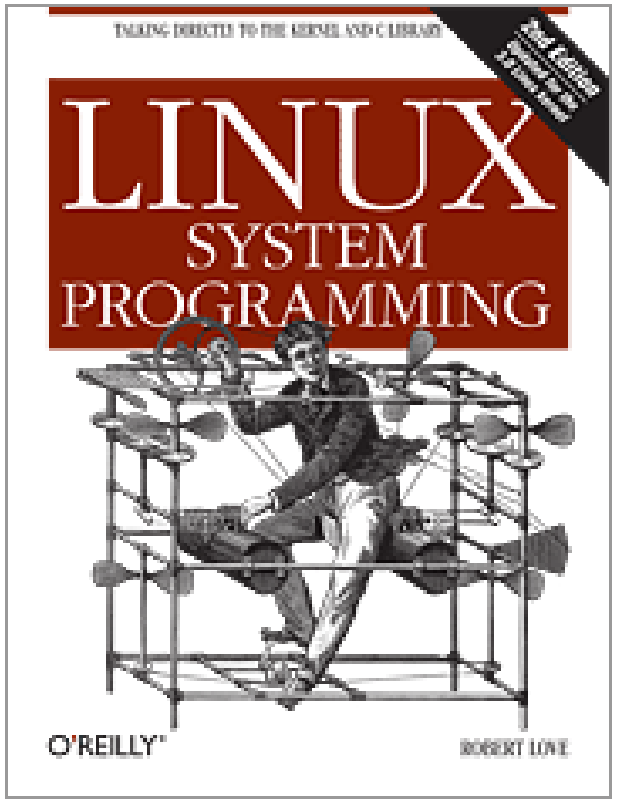


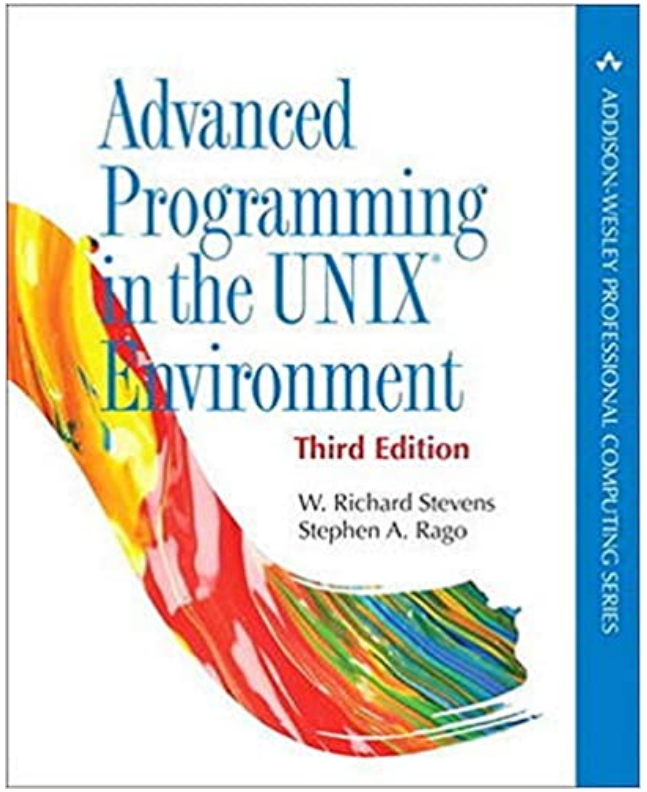


UNIX Systems Programming

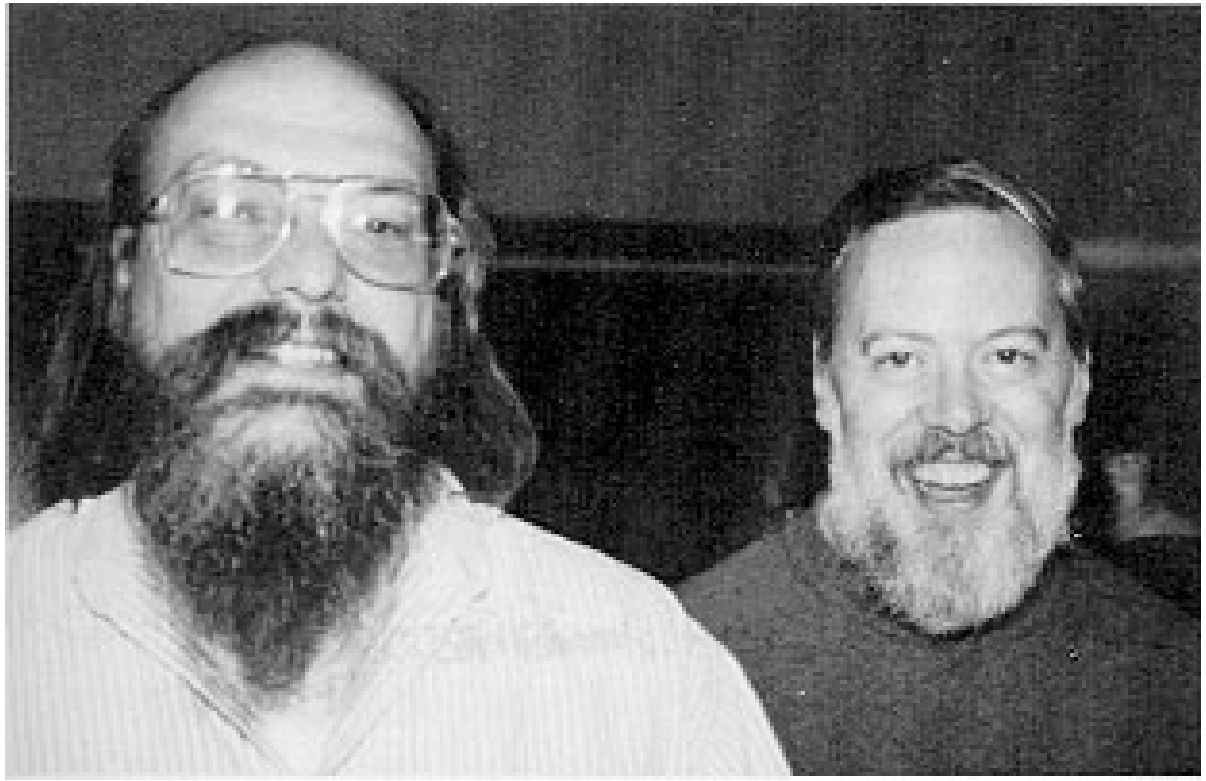
- Books for Systems Programming







UNIX Heroes

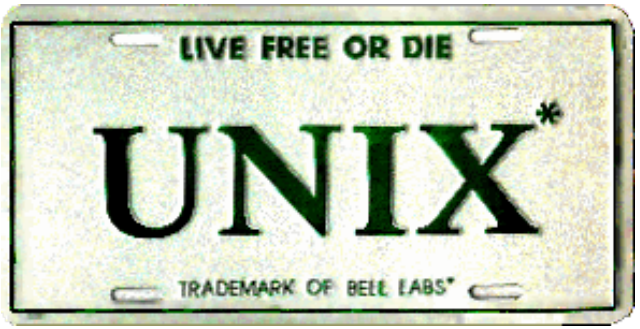


Ken Thompson

Dennis Ritchie

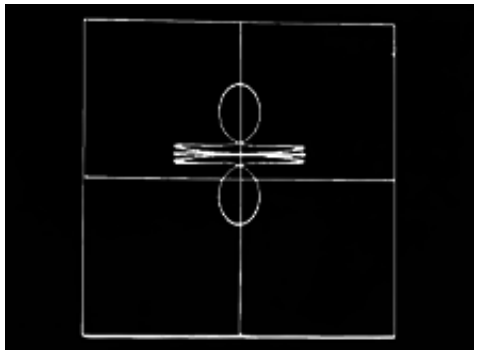
Birth of UNIX

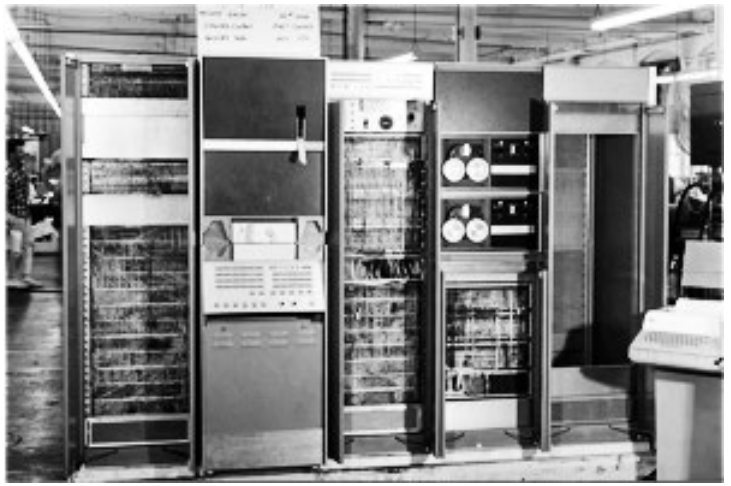
- Late 1960s, Bell Labs, MIT and General Electric to develop a time-sharing system, called Multiplexed Information and Computing Service (Multics)
  - allowing multiple users to access a mainframe simultaneously.
- Ken Thompson had worked on Multics
- Bell Labs researchers led by Thompson and Ritchie, including Rudd Canaday (in 1969), developed a hierarchical file system, the concepts of computer processes and device files, a command-line interpreter, and some small utility programs



Birth of UNIX (1 of 2)

- Ken Thompson programmed a game called *Space Travel*, but it needed a more efficient and less expensive machine to run on, and eventually he found a little-used PDP-7
- The resulting system, much smaller than the envisioned Multics system, was to become Unix
- In 1970, Peter G. Neumann coined the project name *Unics* (UNiplexed Information and Computing Service) as a pun on *Multics*





Birth of UNIX (1 of 2)

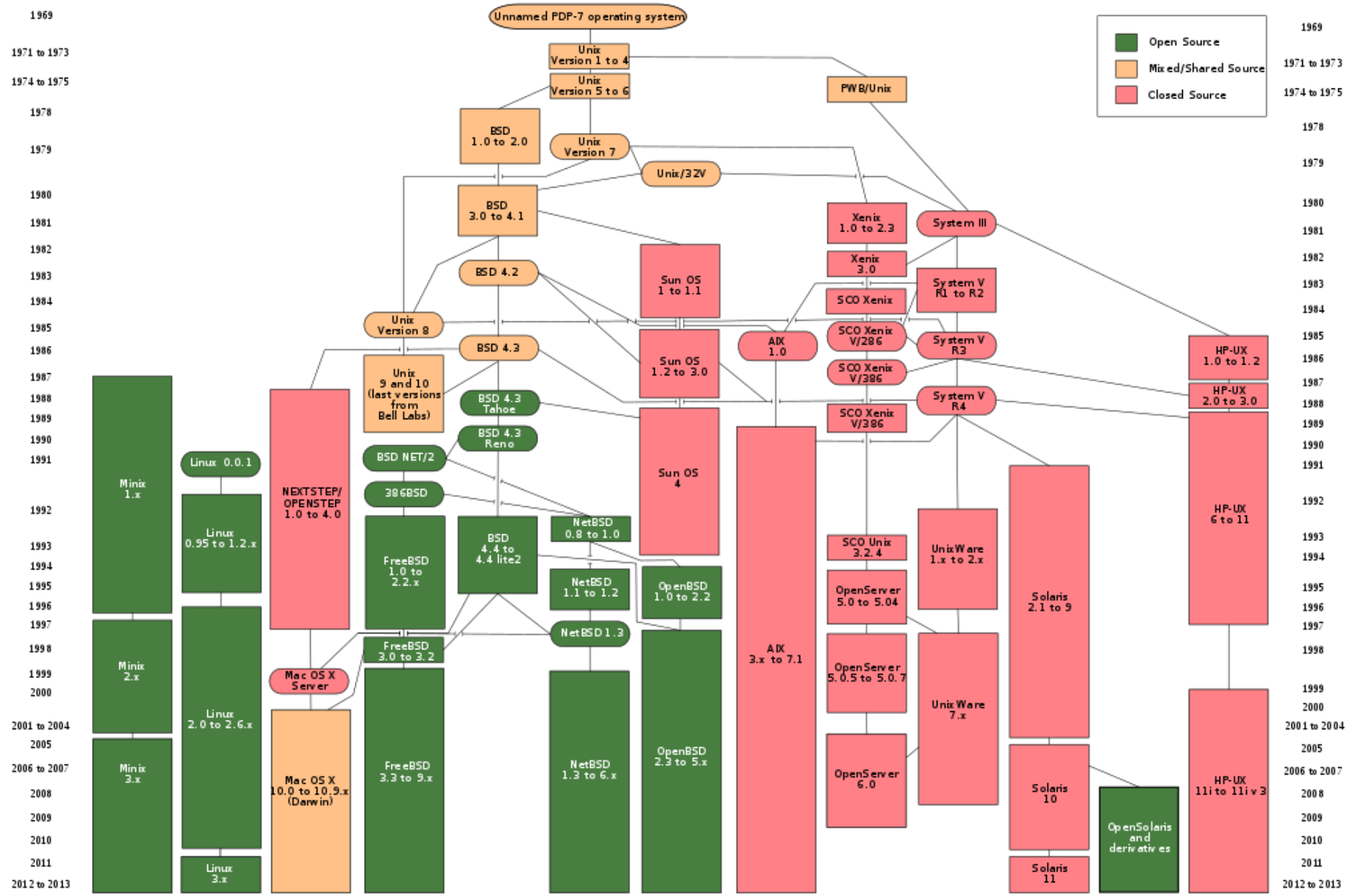
- In 1972, Unix was rewritten in the higher-level language C, contrary to the general notion at the time that an operating system's complexity and sophistication required it to be written in assembly language
- Although assembly did not disappear from the man pages until Version 8, the migration to C resulted in much more portable software, requiring only a relatively small amount of machine-dependent code to be replaced when porting Unix to other hardware computing platforms

Early Beginnings of UNIX

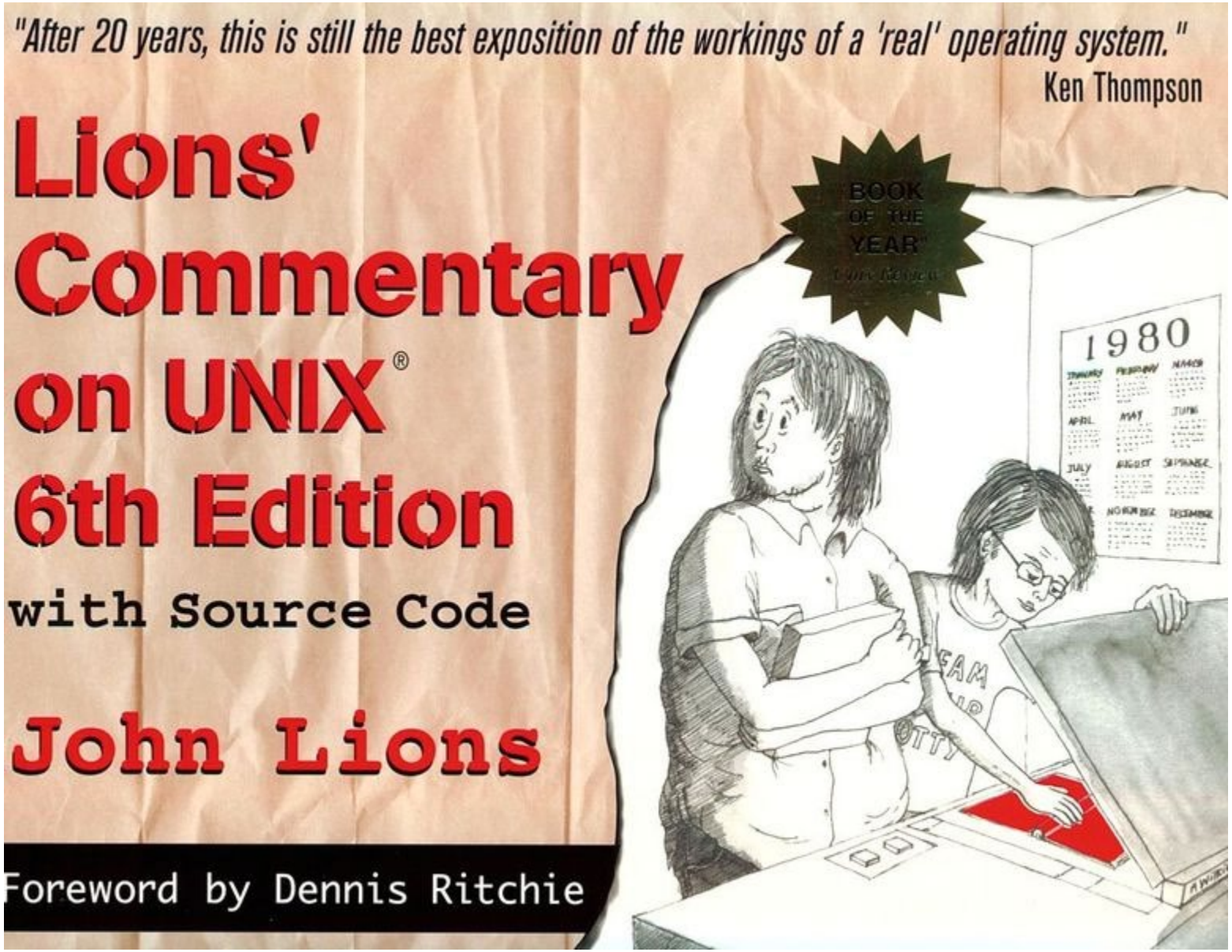
- UNICS: 1969 – PDP-7 minicomputer
- PDP-7 goes away, rewritten on PDP-11
- V1: 1971
- V3: 1973 (pipes, C language)
- **V6**: 1976 (rewritten in C, base for BSD)
- V7: 1979 (Licensed, portable)

PDP-11

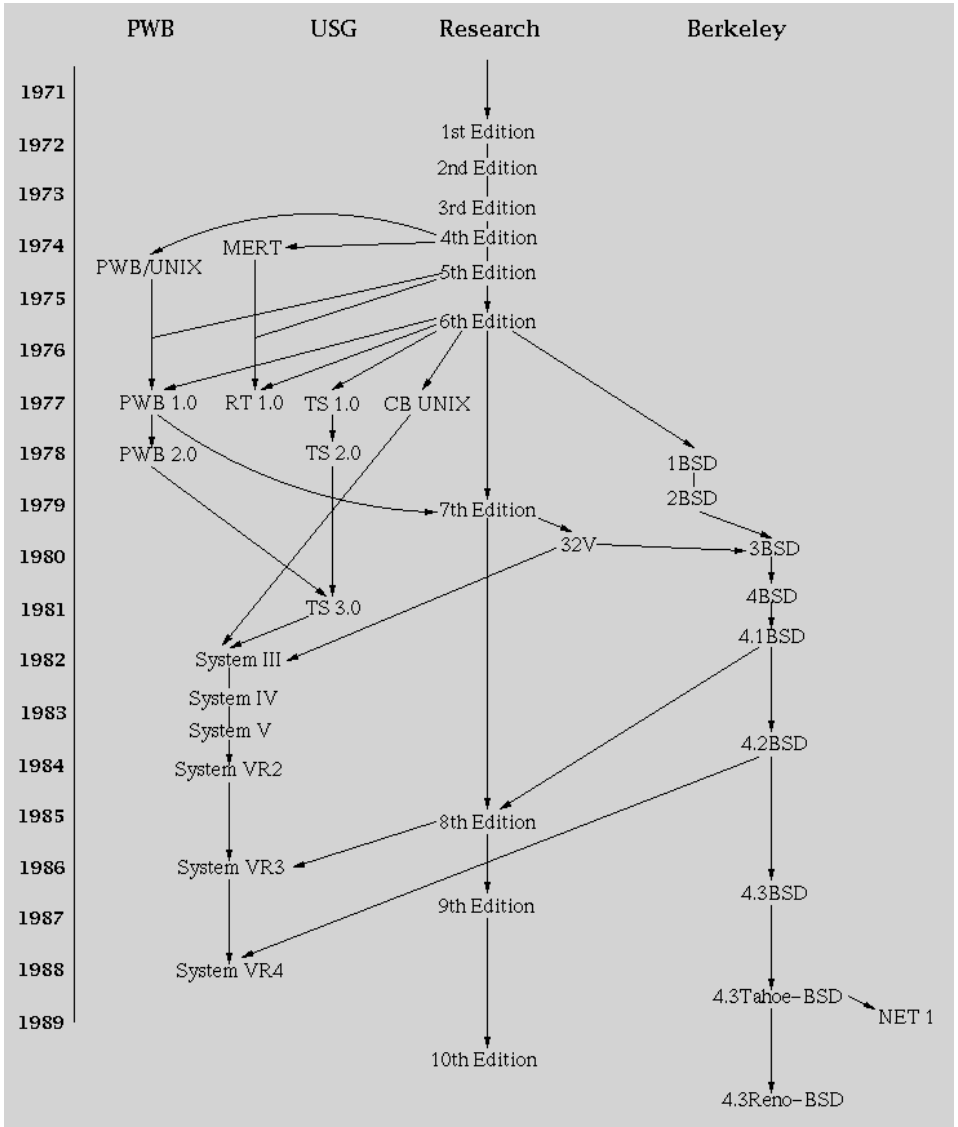
Evolution


Unnamed PDP-7 operating system
Unix Version 1 to 4
Unix Version 5 to 6
PWB/Unix
BSD 1.0 to 2.0
Unix Version 7
Unix/32V
BSD 3.0 to 4.1
Xenix 1.0 to 2.3
System III
Xenix 3.0
BSD 4.2
Sun OS 1 to 1.1
System V R1 to R2
SCO Xenix
Unix Version 8
SCO Xenix V/286
System V R3
BSD 4.3
Sun OS 1.2 to 3.0
AIX 1.0
HP-UX 1.0 to 1.2
SCO Xenix V/386
HP-UX 2.0 to 3.0
Unix 9 and 10 (last versions from Bell Labs)
BSD 4.3 Tahoe
System V R4
BSD 4.3 Reno
Sun OS 4
BSD NET/2
386BSD
NetBSD 0.8 to 1.0
FreeBSD 1.0 to 2.2.x
BSD 4.4 to 4.4 lite2
SCO Unix 3.2.4
UnixWare 1.x to 2.x
NetBSD 1.1 to 1.2
OpenBSD 1.0 to 2.2
OpenServer 5.0 to 5.04
Solaris 2.1 to 9
NetBSD 1.3
FreeBSD 3.0 to 3.2
OpenServer 5.0.5 to 5.0.7
UnixWare 7.x
HP-UX 6 to 11
Mac OS X Server
AIX 3.x to 7.1
OpenServer 6.0
Solaris 10
OpenSolaris and derivatives
Minix 1.x
Linux 0.0.1
NEXTSTEP/ OPENSTEP 1.0 to 4.0
Linux 0.95 to 1.2.x
Minix 2.x
Linux 2.0 to 2.6.x
Mac OS X 10.0 to 10.9.x (Darwin)
FreeBSD 3.3 to 9.x
NetBSD 1.3 to 6.x
OpenBSD 2.3 to 5.x
Minix 3.x
Linux 3.x
Solaris 11
HP-UX 11i to 11i v3
Open Source
Mixed/Shared Source
Closed Source
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

Big Reason for V6 Success



UNIX Derivative Systems



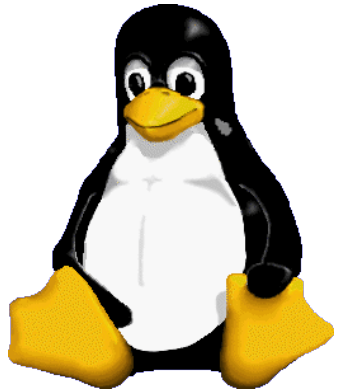
- Programmers Work Bench(PWB),
- Multi-Environment Real-Time (MERT)
- BSD: Adds many important features (networking, job control).
- AT&T enters the computer business with System III, V

Commercial Success

- AIX 
- SunOS, Solaris 
- Ultrix, Digital Unix 
- HP-UX 
- Irix 
- UnixWare -> Novell -> SCO -> Caldera ->SCO
- Xenix: -> SCO 
- Standardization (Posix, X/Open)

Send in the Clones

- **Linux**
  - Written in 1991 by Linus Torvalds
  - Most popular UNIX variant
  - Free with GNU license



- **BSD Lite**
  - FreeBSD (1993, focus on PCs)
  - NetBSD (1993, focus on portability)
  - OpenBSD (1996, focus on security)
  - Free with BSD license
  - Development less centralized

Lou Gerstner gives his keynote address at the eBusiness Conference and Expo in New York.

IBM to spend $1 billion on Linux in 2001

By Joe Wilcox
Staff Writer, CNET News.com
December 12, 2000, 8:50 a.m. PT

update IBM chief executive Louis Gerstner said Tuesday that his company will spend $1 billion on Linux next year.

Gerstner made the announcement at the eBusiness Conference and Expo in New York, where IBM also revealed a Linux supercomputer win with Shell Oil.

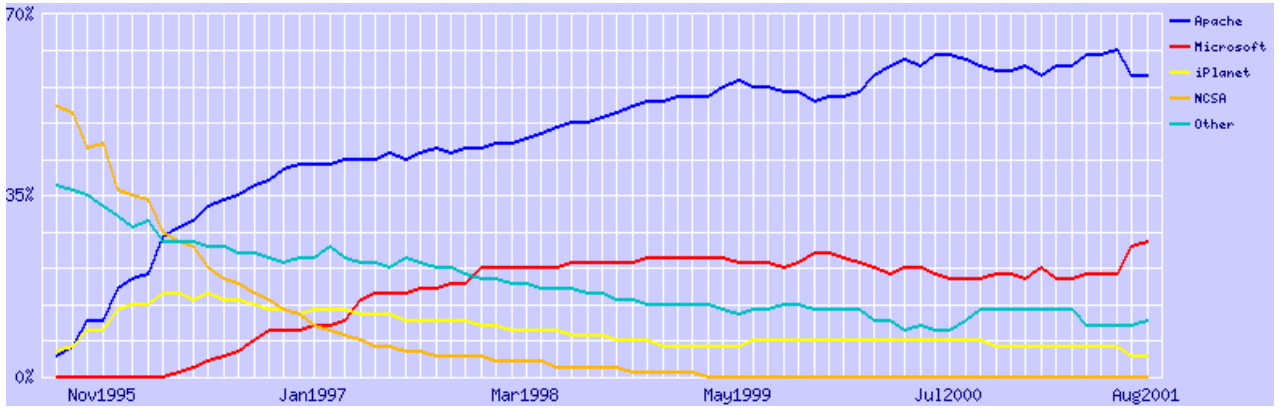
Info appliance makers adopt Linux

Just buzz or actual benefits? More info appliance makers are choosing Linux.

Intel	To use Linux for Intel-branded Web appliances
TiVo	Runs personal video recorder services on Linux
National Semiconductor	Offers Linux choice for its Web Pad platform
Sony	PlayStation 2 development system based on Linux
Transmeta	Bundling Linux for mobile applications with new chip
Lineo	Offers Linux development system for embedded info devices

SOURCE: Company announcements

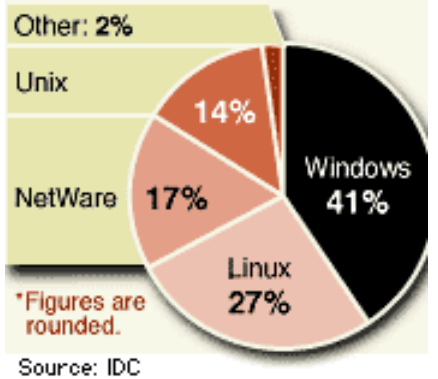
Today: Unix is Big



Server share

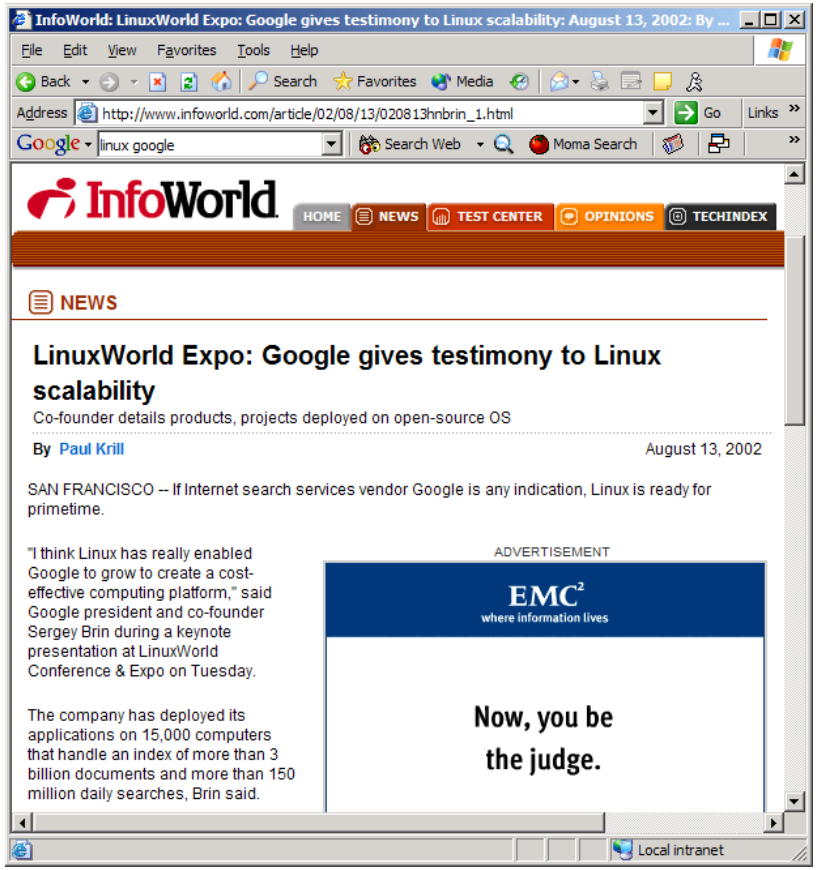
Four companies dominated the market for server operating systems last year.

Market share*



Source: IDC

Linux at Google & Elsewhere



Disney, DreamWorks, Pixar Go Linux

Posted by Hemos on Wed Jul 27, '05 03:19 PM

from the moving-into-the-future dept.

robinsrowe writes *"Most of the major studios use Linux -- such as DreamWorks with more than 1,500 Linux desktops and 3,500 Linux servers. The MovieEditor Conference is an all-day event on computer-based filmmaking in downtown Los Angeles on August 3rd. Studio technology chiefs and other experts discuss ongoing work using Linux in feature animation and visual effects. Presented in collaboration with LinuxMovies.org."*

Darwin

- Apple abandoned old Mac OS for UNIX
  - Purchased NeXT in December 1996
  - Unveiled in 2000
  - Based on 4.4BSD-Lite
  - Aqua UI written over Darwin
  - Open Source

- Darwin forms the core set of components upon which macOS (previously OS X and Mac OS X), iOS, watchOS, tvOS, and iPadOS are based.

Why did UNIX succeed?

- Technical strengths!
- Research, not commercial
- PDP-11 was popular with an unusable OS
- AT&T's legal concerns
  - Not allowed to enter computer business but needed to write software to help with switches
  - Licensed cheaply or free
- Rewritten in C to be made portable to various hardware platforms

The Open Source Movement

- Has fueled much growth in UNIX
  - Keeps up with pace of change
  - More users, developers
  - More platforms, better performance, better code
- Many vendors switching to Linux

Article about Open Source

washingtonpost.com

The Open Source Threat

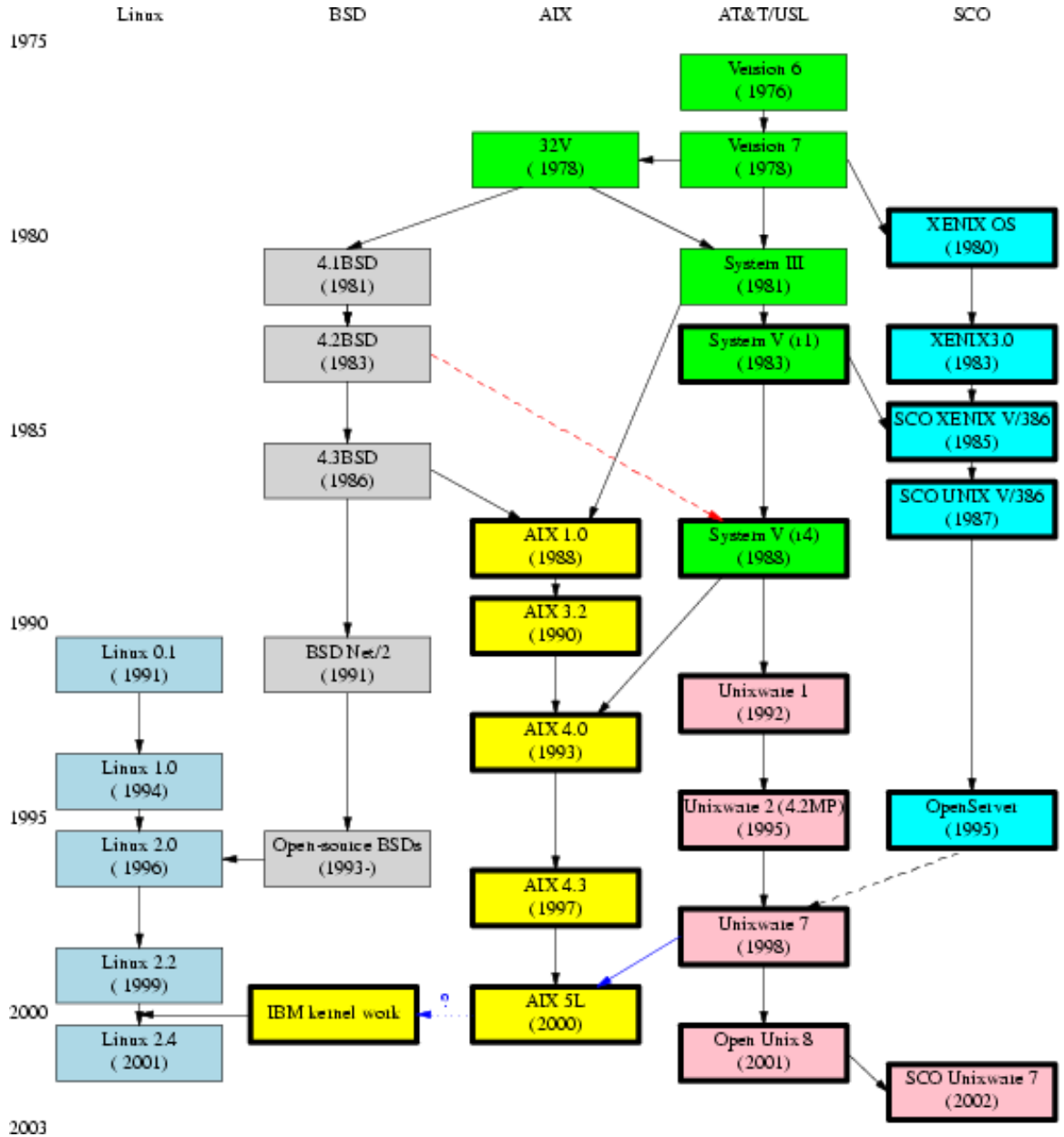
By Cynthia L. Webb
washingtonpost.com Staff Writer
Tuesday, September 7, 2004; 9:54 AM

Open-source software, namely **Linux**, is nipping more sharply at the heels of **Microsoft**, leading the software giant to defend itself more fiercely than ever against the insurgent rise of freely distributed, collaboratively coded programs.

The Redmond, Wash.-based software giant acknowledged Linux is a growing challenge to its business in its 10-K filing with the **Securities and Exchange Commission**. Microsoft "is facing growing pressure from open-source software across every segment of its business -- a competitive threat that could have significant consequences for its financial future going forward," eWeek reported. "While Microsoft often mentions Linux and open-source software as a potential threat to its business, it seems to be treating the threat far more seriously and describing it as more pervasive than in previous official filings."

Linux "is making inroads in servers and PCs," Australian IT said in its coverage of the filing. Here's what Microsoft had to say: "To the extent open source software products gain increasing market acceptance, sales of our products may decline, which could result in a reduction in our revenue and operating margins." More from the filing: "We continue to watch the evolution of open-source software development and distribution and continue to differentiate our products from competitive products, including those based on open-source software. We believe that Microsoft's share of server units grew modestly in fiscal 2004, while Linux distributions rose slightly faster on an absolute basis." And Microsoft's filing also offers this survey of its competitors: "**IBM**'s endorsement of Linux has accelerated its acceptance as an alternative. ... Linux's competitive position has also benefited from the large number of compatible applications now produced by many leading commercial software developers as well as non-commercial software developers," Microsoft said.

UNIX vs. Linux



Linux Distributions

- Slackware – the original
- Debian – collaboration of volunteers
- Red Hat / Fedora – commercial success
- Ubuntu – currently most popular, based on Debian. Focus on desktop
- Gentoo – portability
- Knoppix – live distribution

The UNIX Philosophy (1 of 4)

- Small is beautiful
  - Easy to understand
  - Easy to maintain
  - More efficient
  - Better for reuse
- Make each program do one thing well
  - More complex functionality by combining programs
  - Make every program a filter

The UNIX Philosophy (2 of 4)

- Portability over efficiency
  - Most efficient implementation is rarely portable
  - Portability better for rapidly changing hardware
- Use flat ASCII files
  - Common, simple file format (yesterday's XML)
  - Example of portability over efficiency
- Reusable code
  - Good programmers write good code;
    great programmers borrow good code

The UNIX Philosophy (3 of 4)

- Scripting increases leverage and portability

```
print $(who | awk '{print $1}' | sort | uniq) | sed ‘s/  / , /g’
```

- List the logins of a system’s users on a single line.

who	**755**
awk	**3,412**
sort	**2,614**
uniq	**302**
sed	**2,093**

9,176 lines

- Build prototypes quickly (high level interpreted languages)

The UNIX Philosophy (4 of 4)

- Avoid captive interfaces
  - The user of a program isn't always human
  - Look nice, but code is big and ugly
  - Problems with scale
- Silence is golden
  - Only report if something is wrong
- Think hierarchically

UNIX Highlights / Contributions

- Portability (variety of hardware; C implementation)
- Hierarchical file system; the file abstraction
- Multitasking and multiuser capability for minicomputer
- Inter-process communication
  - Pipes: output of one programmed fed into input of another
- Software tools
- Development tools
- Scripting languages
- TCP/IP

The Operating System

- The government of your computer
- Kernel: Performs critical system functions and interacts with the hardware
  - Manages the resources of your computer, including peripherals, files, memory
  - Coordinates all programs running on the computer
- Systems utilities: Programs and libraries that provide various functions through systems calls to the kernel

Kernel Basics

- The kernel is …
  - a program loaded into memory during the boot process, and always stays in physical memory.
  - it is an interface between application and actual data processing at the hardware level.
  - The kernel connects the system hardware to the application software.
  - responsible for managing CPU and memory for processes, managing file systems, and interacting with devices.

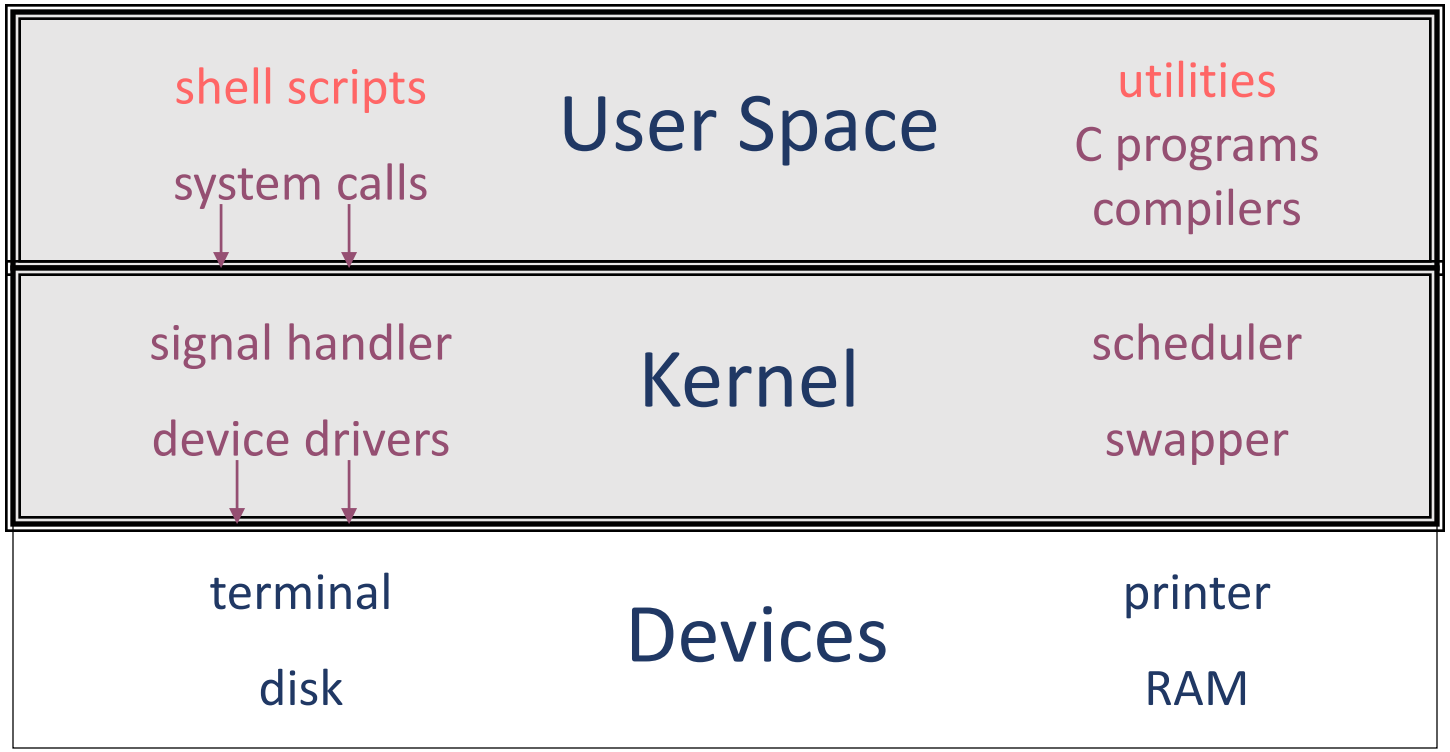
Main Kernel Tasks (1 of 2)

- **Resource allocation**
  - The kernel's primary function is to manage the computer's resources and allow other programs to run and use these resources.
  - These resources are- CPU, Memory and I/O devices.
- **Process Management**
  - A **process** defines which memory portions the application can access.
  - The main task of a kernel is to allow the execution of applications and support them with features such as hardware abstraction.
- **Memory Management**
  - The kernel has full access to the system's memory. It allows processes to safely access this memory as they require it.
  - **Virtual addressing** helps kernel to create virtual partitions of memory in two disjointed areas, one is reserved for the kernel (**kernel space**) and the other for the applications (**user space**).
- **I/O Device Management**
  - To perform useful functions, processes need access to the peripherals connected to the computer, which are controlled by the kernel through **Device Drivers**.
  - A device driver is a computer program that enables the operating system to interact with a hardware device. It provides the operating system with information of how to control and communicate with a certain piece of hardware.

Main Kernel Tasks (2 of 2)

- **Inter- Process Communication**
  - Kernel provides methods for Synchronization and Communication between processes called Inter- Process Communication (IPC).
  - There are various approaches of IPC say, semaphore, shared memory, message queue, pipe (or named fifo), etc.
- **Scheduling**
  - In a Multitasking system, the kernel will give every program a slice of time and switch from process to process so quickly that it will appear to the user as if these processes were being executed simultaneously.
  - The kernel uses **Scheduling Algorithms** to determine which process is running next and how much time it will be given. The algorithm sets **priority** among the processes.
- **System Calls and Interrupt Handling**
  - A **system call** is a mechanism that is used by the application program to request a service from the operating system. System calls include close, open, read, wait and write.
  - To access the services provided by the kernel we need to invoke the related kernel functions. Most kernels provide a C Library or an API, which in turn invokes the related kernel functions.
- **Security or Protection Management**
  - Kernel also provides protection from faults (**error control**) and from malicious behaviors (**Security**).
  - One approach toward this can be **Language based protection system**, in which the kernel will only allow code to execute which has been produced by a trusted language compiler.

UNIX Structural Layout

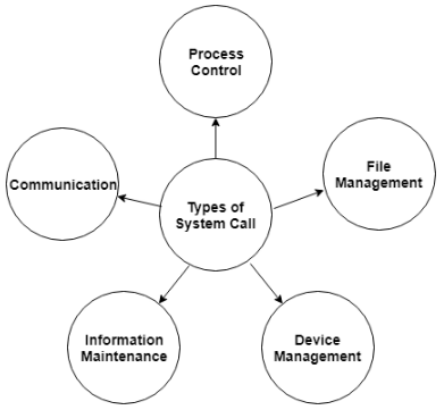


Kernel Subsystems

- Process management
  - Schedule processes to run on CPU
  - Inter-process communication (IPC)
- Memory management
  - Virtual memory
  - Paging and swapping
- I/O system
  - File system
  - Device drivers
  - Buffer cache

Unix System Calls

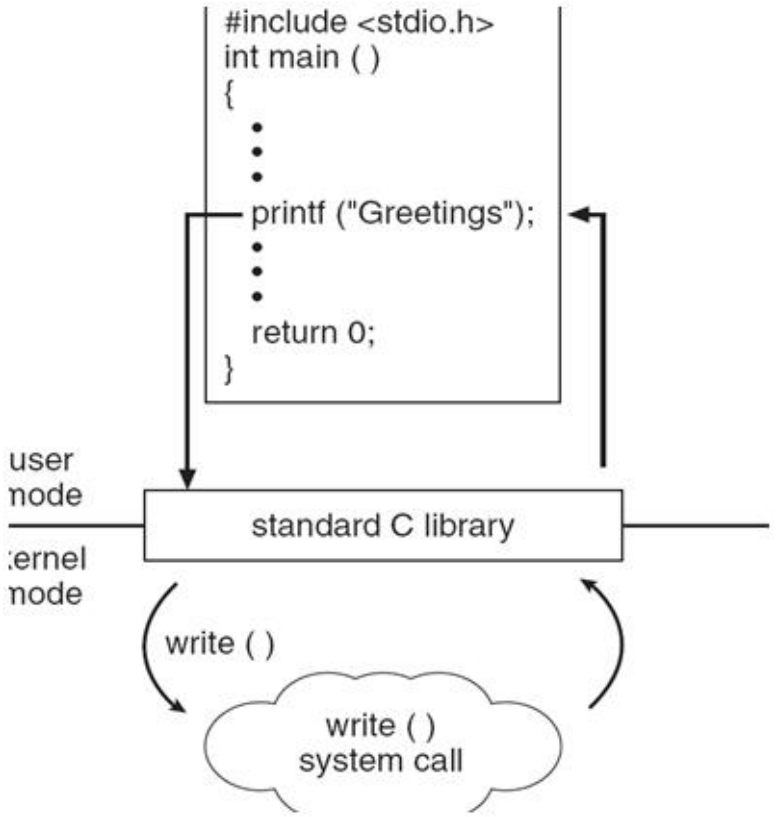
- Interface to the kernel (Kernel API)
- Over 1,000 system calls available on Linux
- 5 categories
  - File management
    - e.g. mkdir(), open(), read()
  - Process control
    - e.g. fork(), exit(), nice()
  - Information management
    - e.g. getuid(), time(), alarm(), sleep()
  - Device management
    - e.g. ioctl(), read(), write()
  - Communication
    - e.g. pipe(), shmget(), mmap()



	Windows	Unix
Process Control	CreateProcess() ExitProcess() WaitForSingleObject()	fork() exit() wait()
File Manipulation	CreateFile() ReadFile() WriteFile() CloseHandle()	open() read() write() close()
Device Manipulation	SetConsoleMode() ReadConsole() WriteConsole()	ioctl() read() write()
Information Maintenance	GetCurrentProcessID() SetTimer() Sleep()	getpid() alarm() sleep()
Communication	CreatePipe() CreateFileMapping() MapViewOfFile()	pipe() shmget() mmap()
Protection	SetFileSecurity() InitlializeSecurityDescriptor() SetSecurityDescriptorGroup()	chmod() umask() chown()

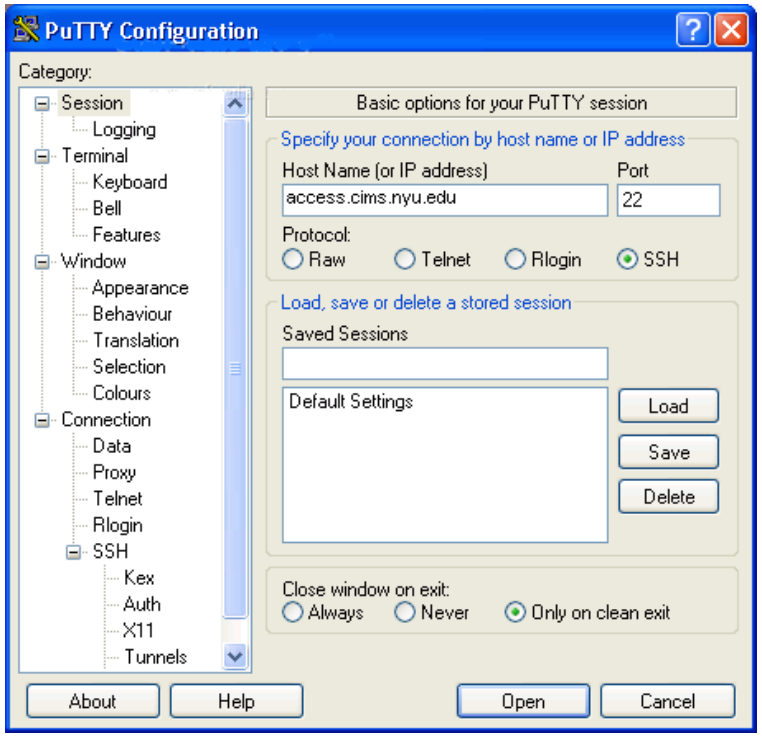
System Calls using C

- The standard C library provides a portion of the system call interface for many version of UNIX and Linux.
- Example shows a C program invoking printf() statement.
- The C library intercepts this call and invokes the necessary system call(s) in the OS.
- The C library takes the value returned by write() and passes it back to the user program



Remote Login

- Use Secure Shell (ssh) from terminal window on Linux or Mac
- Windows
  - e.g. PuTTY



- Command to type at prompt

  ssh username@login.cpp.edu

First Exercise: Logging In

- Need an account and password first
  - Enter at **`login:`** prompt
  - Password not echoed
  - After successful login, you will see a shell prompt
- Entering commands
  - At the shell prompt, type in commands
    - Typical format: **command** *options arguments*
    - Examples: **`who`**, **`date`**, **`ls`**, **`cat`** ***`myfile`***, **`ls -l`**
  - Case sensitive
- **`exit`** to log out (Control D)