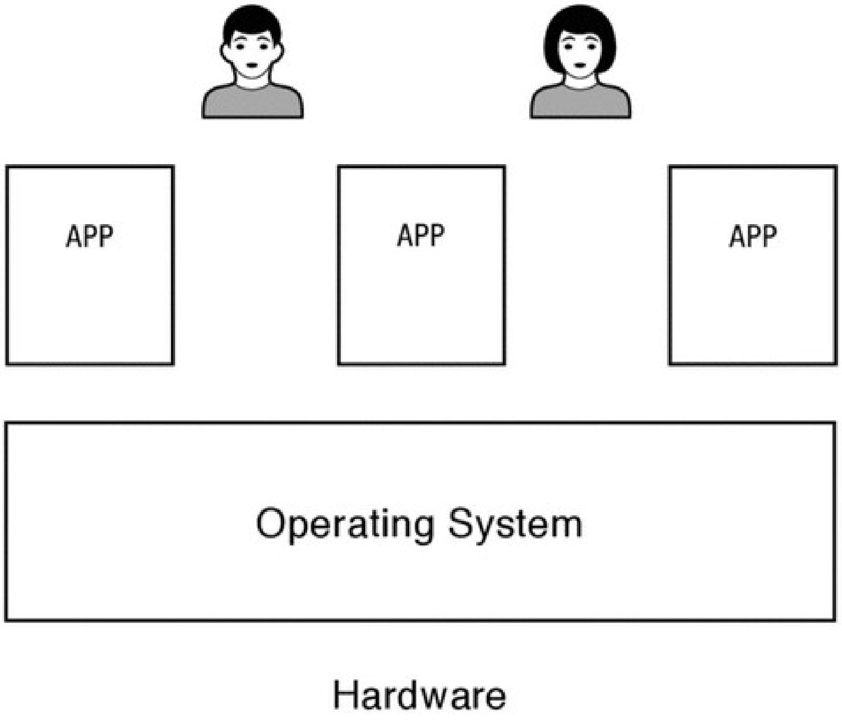
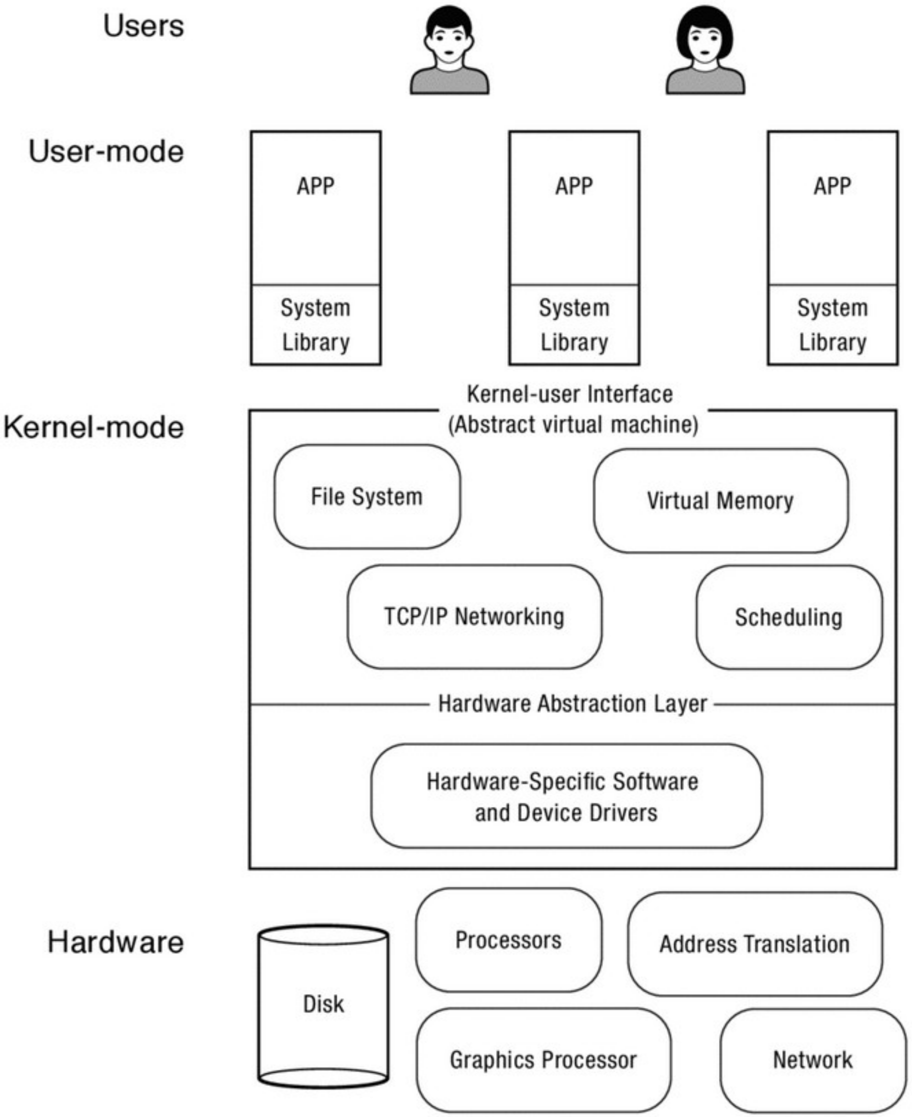
Operating Systems

- Low level software system that
  - manages all applications
  - implements an interface between applications and resources
  - manages available resources
- Resource manager
- Interface
- Virtual Machine

Operating System





Operating System

- OS kernel
  - Code executed in privileged mode
- User space
  - Code executed in non privileged mode
- Service / daemons
  - Application that executed in the background (server)
- Utility programs
  - Application provided by the OS and executed by the user (editor, shell, compiler()
- System calls
  - functions that implement parts of the OS services or utilities
  - can be used inside the kernel
  - Manage and change internal structure.

Operating System Roles

- Referee
  - management of resources shared between application
  - Isolation of applications from each other
  - Decide which applications get which resources
- Illusionist
  - Provides abstraction of physical hardware
    - I/O, available memory, ...
- Glue
  - Provide standard common services
    - I/O (disk, network, ...)
    - Service implementation

Operating System requirements

- Reliability
- Availability
- Security
- Portability
- Performance
- Adoption

Operating System requirements

- Reliability
  - Capacity to work correctly
  - Hide HW faults
  - Reduce SW bugs
  - Handle attacks
- Availability
  - Time the system is working
  - Affected by frequency of failures (MTBF) and time to restore system

- Buggy system that crashes frequently but never looses information
  - Reliable but not available
- Subverted system that appears to be working correctly
  - Available but not reliable

Operating System requirements

- Security
  - Guarantee that operations are not compromised by malicious attacks
  - Privacy
    - guarantee that data is only accessible to authorized users
- SW has bugs that can be exploited
- Administrator can be untrustworthy
- SW developer can be untrustworthy
- OS Design should minimize vulnerabilities
  - Computer virus
  - Downloaded code

Operating System requirements

- Portability
  - Applications are given an abstraction for the HW
    - Resources
    - Access
- Portable abstractions do not change with time
- Abstract virtual machines
  - WIN32
  - POSIX

Operating System requirements

- Performance
  - Efficient use of of available resources
- Overhead
  - added cost of implementing an abstraction
  - Efficiency
    - lack of overhead
- Fairness
  - how to divide resources (memory, CPU) to multiple applications
- Response Time
  - how long a task takes to run
- Throughput
  - rate at which tasks are completed
- Performance predictability

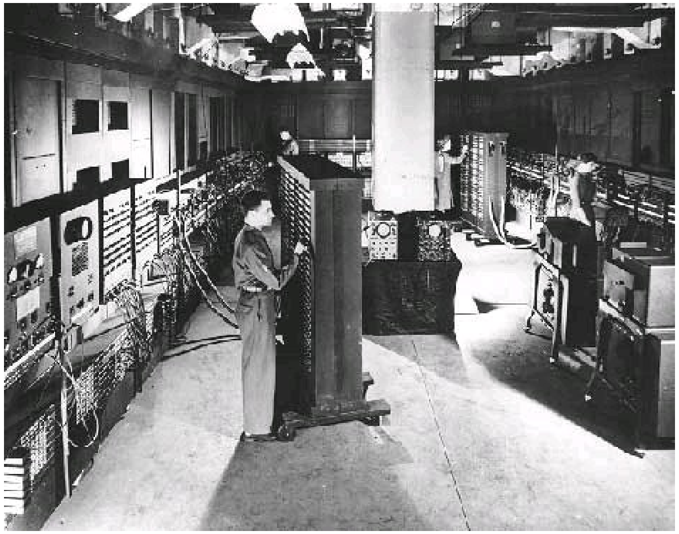
Operating System requirements

- Adoption rate
  - depend on
    - Availability of Applications
    - Availability of hardware

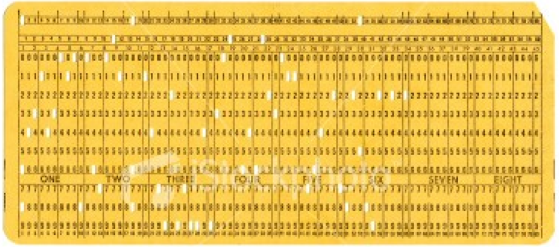
- Open vs closed
- Standardized API
  - more guaranteed of stability
  - more guarantees of portability
- Interoperability

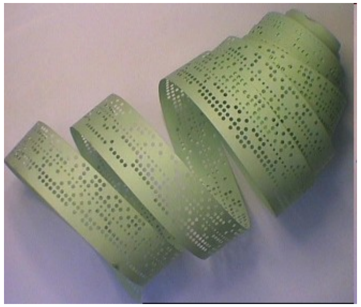
Past, Present Future

- 1945: ENIAC-Electronic Numerical INtegrator And Computer,
  - 19K valves e 1K relays
  - 200 KW
  - 167 m2



- Numeric computations
  - 5000 sums per second or 357 mult per second or 38 divs per second





PDP-7

- First computer to run UNIX
- Introduced in 1965
- Less expensive alternative to the PDP-4
  - price of only U.S.$72,000
- 18-bit word length
- Standard main memory was 4K words (equivalent to nine kilobytes)
  - 64K words (144 KB)

Motorola 68000

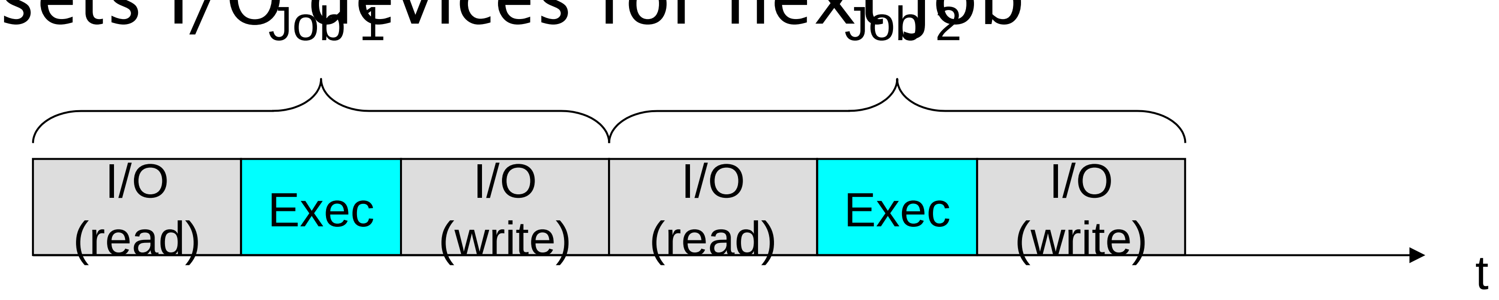
- Developed in the start of 80’s
- Used by HP, SUN, DEC, CGI, Apple
- 32-bit
  - instruction setm registers and internal data bus
  - 16-bit external data bus
- Virtual memory
- Frequency < 10Mhz

Speed/Price

		CPU	• RAm	Price	MIPS
1945	ENIAC			6 936 746	0.0000182
1965	DEC PDP-8		• 200kb	146 508	
1983	IBM PC/XT	i8088	16 kB – 256 kB	12 870	0.25
1984	Apple Macintosh	6800	128 KB	6100	0.50
1989	Sun SS1	Sparc	64 MB	18 598	10
1991	Dell 433P	i486DX	32Mb	5 645	11
1991	HP 730	HP-PA7000	32	28,029	78
2001	SUN Blade 1000	UltraSPARC	1 Gb	14 477	4200
2001		2xPentium 4	1Gb	1,447	2500

Past, Present Future

- Processor idle while reading data
- Batch processing
  - Load
  - Run
  - Unload
- While a job was running
  - OS sets I/O devices for next job



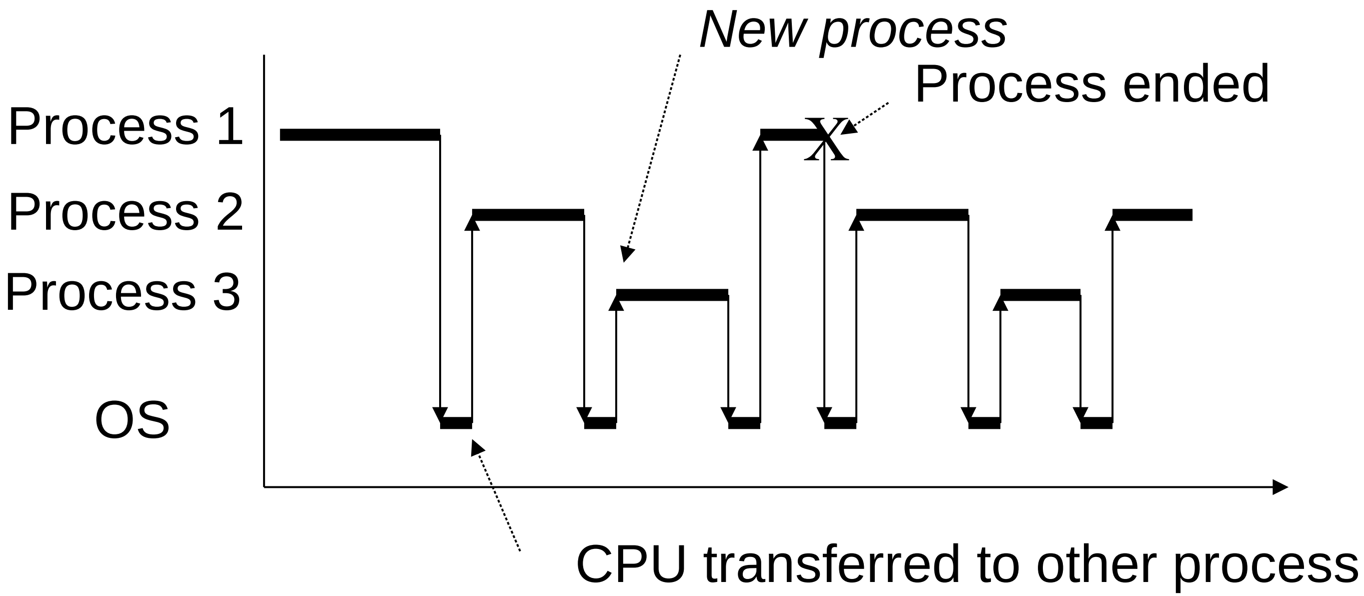
Past, Present Future

- Processor idle while reading data
- Multitasking / multiprocessing
  - multiple programs loaded into memory
  - OS selects program to execute
  - Blocking programs (I/O) do not use CPU
- Efficeint
  - if queue of tasks is long
  - I/O devices feed processor
- requires program isolation

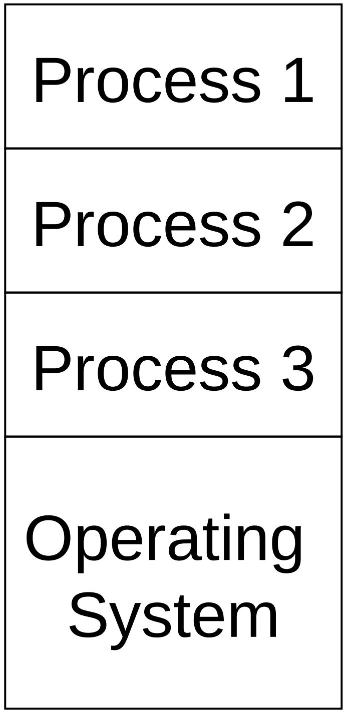
Past, Present Future

- Processor idle while reading data
- User can interact with the programs
  - multiple display/keyboards
  - make program idle most of the time
- Time-sharing
  - multiple programs loaded into memory
  - Programs from different users
  - OS selects program to execute
  - Blocking programs (I/O) do not use CPU
- Efficient
  - if programs require I/O
- Requires program isolation
- Requires user isolation

- CPU Usage



- Memory usage



History of UNIX

Past, Present Future

- Modern OS
  - Desktop
  - Smartphone
  - Server
  - Embedded
  - Virtual machines
  - Server clusters
  - Cloud

- Future OS
  - VLS datacenters
  - VLS Multicore systems
  - Mobile computing
  - Heterogenous systems
  - Large Storage
  - IoT

The UNIX Time-Sharing System

- Most important achievement
  - demonstrate that powerful operating system for interac-tive use need not be expensive
    - neither in equipment or in human effort:
- UNIX can run on hardware costing as
  - little as $40,000,
- the main system software development
  - less than two man years

The UNIX Time-Sharing System

- The PDP-11/45
  - $40,000
  - 16-bit word (8-bit byte)
  - 144Kbytes of core memory
  - 1M byte fixed-head disk
  - 2.5M bytes on removable disk car-tridges
  - console typewriter

The UNIX Time-Sharing System

- Most important job of UNIX is to provide a file system.
  - Ordinary Files
  - Directories
  - Special Files
- Removable File Systems
- Protection
- I/O Calls
  - designed to eliminate thedifferences between the various devices and styles ofaccess

The UNIX Time-Sharing System

- Processes and Images
  - An image is a computer execution environment. Iti
  - core image (memory)
  - general register values
  - status of open files, current directory, and the like.
- An image is the current state of a pseudo computer.

The UNIX Time-Sharing System

- Processes and Images
  - A process is the execution of an image.
  - While the processor is executing on behalf of a process
    - the image must reside in core
  - Segment organization
    - text segment begins at location 0 in the virtual address space.
      - During execution, this segment is write-protected and a single copy of it is shared among all processes executing the same program.
    - At the first 8K byte boundary above the program text segment
      - begins a non-shared, writable data segment, the size of which may be extended by a systemcall.
    - Starting at the highest address
      - stack segment, which automatically grows down-ward as the hardware’s stack pointer fluctuates.

The UNIX Time-Sharing System

- Processes communciation
  - Processes may communicate with related processes through pipes
  - Using he same system read and write calls that are used for file system I/O:
- Traps
  - The PDP-11 hardware detects a number of program faults
    - such as references to nonexistent memory, unimplemented instructions, and odd addresses used where an even address is required.
  - Such faults cause the processor to trapto a system routine
  - Programs which are looping,
    - may be halted by the use of the interrupt signal,

The UNIX Time-Sharing System

- 1969:
  - Bell withdraws from Multics;
  - Thompson starts pondering new os
  - ARPAnet
  - Thompson writes 1st Unix on PDP-7 in one month based on Multics
- 1970
  - Kernighan suggests name 'UNICS' (Uniplexed Information & Computing Service)
  - Thompson writes B language
  - DEC announces PDP-11 [fansite] (will sell 250k)
  - Bell orders $65k PDP-11/20 w/24kb RAM and 500k disk

The UNIX Time-Sharing System

- 1971
  - Unix rewritten for PDP-11/20
    - os uses 16k, files limited to 64kb
  - **adds pathnames**
  - Unix version 1 written in B
- 1972
  - Ritchie creates C language
  - Unix version 2 written mostly by Thompson in C
- 1973
  - Unix version 3 introduces **pipes text streams** as universal interface
  - multiprogramming added
  - #include and #define added to C
  - Unix kernel rewritten in C
  - Berkeley gets interested in Unix

The UNIX Time-Sharing System

- 1974
  - AT&T decides to supply Unix-source free to academia
- 1976
  - Xerox invents Ethernet
  - Unix licensed by 138 institutions
- 1977
  - Berkeley Software Distribution (Pascal and ex editor)
  - Unix v6 ported for first time
  - Unix clone 'Coherent'

The UNIX Time-Sharing System

- 1978
  - Unix Timesharing System version 7 (UTS)
  - K&R's "The C Programming Language"
  - 'Idris' is Unix clone for PDP-11 purposely incompatible
  - Bill Joy writes **vi** [story]
  - Second Berkeley Software Distribution (2BSD)
    - with **Pascal, vi, termcap, Mail, more, csh, ex**
  - **Berkeley adds virtual memory to Unix 7**
-

The UNIX Time-Sharing System

- 1979
  - **Unix version 7**
    - **Bourne shell, awk, lint, make, uucp; find, cpio, expr;**
    - **large file-systems,**
    - **unlimited users**
    - kernel is still just 40kb
  - Santa Cruz Operation (SCO) founded by Doug and Larry Michels to create Unix ports
  - 1979: Motorola introduces 68000 16-bit CPU
  - IBM prototypes PC using still-scarce 68000
  - Intel introduces 8088 as discount 8-bit version of 8086
  - IBM chooses 8088 for 'short run' PC
  - Unix 32V for VAX [source code]
  - 3BSD is 32-bit port of 2BSD w/ **virtual memory and C shell**

The UNIX Time-Sharing System

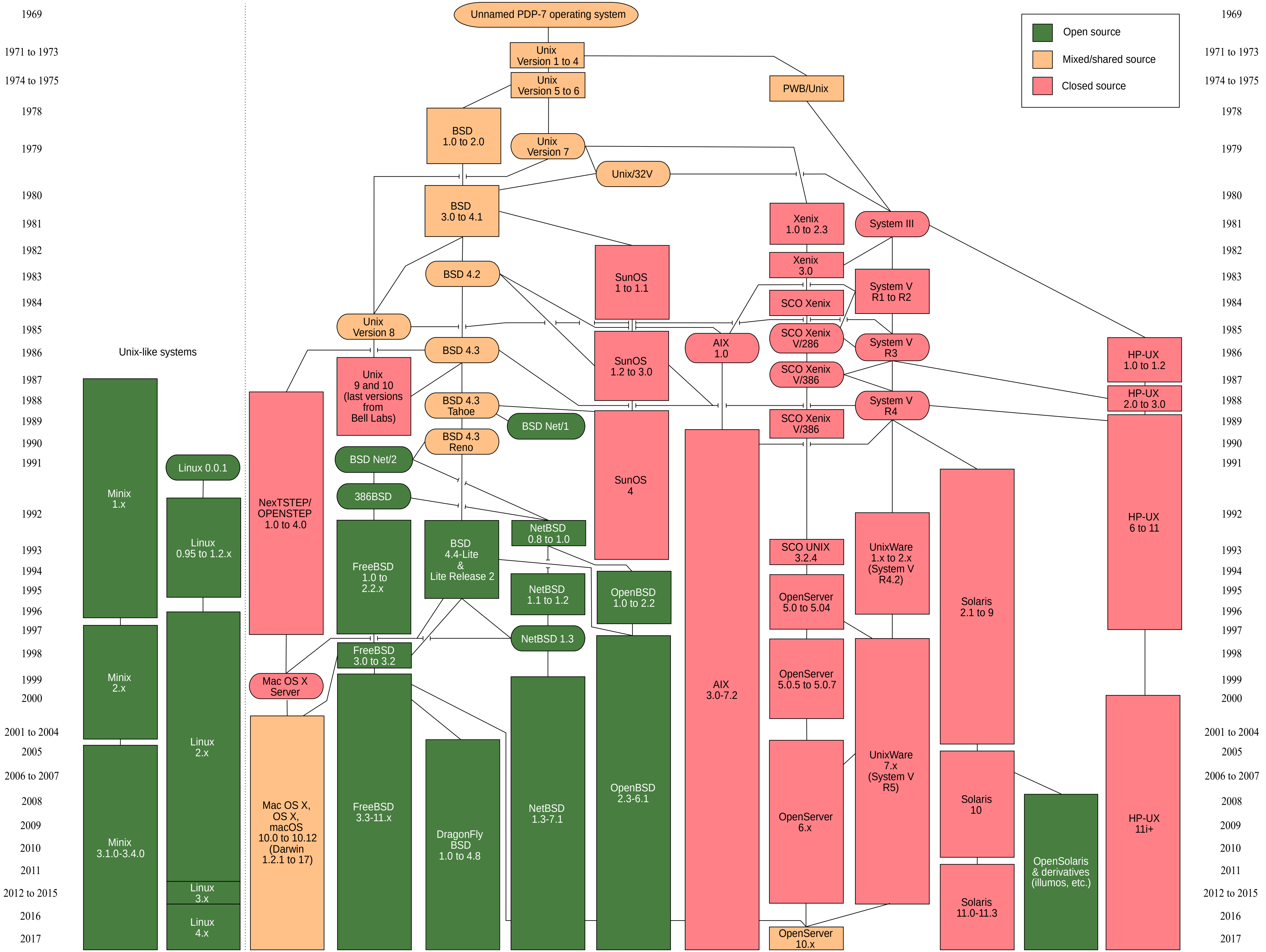
- 1980
  - Bell Labs finally shows interest in BSD Unix
  - Berkeley gets DARPA money because of VAX experience (Comp Systems Research Group)
  - users group for unlicensed (binary-only) users called '**usr**' (later UniForum)
  - Microsoft's 1st-ever os 'Xenix' announced for 8086-et-al,
    - based partly on BSD
    - heavily used at MS for software development and documentation
  - 4BSD incl Franz Lisp, **DARPA enhancements**, curses

The UNIX Time-Sharing System

- 1981
  - '4.1BSD includes speed tweaks and Delivermail
  - no-date: Unix lookalikes include: Onyx, Zeus, Cromix, Omnyx
  - 37yo professor Andy Tanenbaum of Free University in Amsterdam starts work on Minix teaching-system in response to AT&T's stricter license
- 1982
  - Intel intros 80286 with protected memory mode
  - AT&T's 1st commercial Unix, System III
  - AT&T 6300 w/PC Unix
  - no-date: Inix
  - MS Xenix 3.0 will be based on System III but include some 4.1BSD?
  - IBM's version for AT will be called IBM Xenix 1.0;
  - '4.1aBSD' adds TCP/IP and rlogin, etc sendmail?
  - Microsoft promises Xenix ports for PDP-11, 8086, Z8000, and 68000

The UNIX Time-Sharing System

- 1983
  - Apple Lisa runs MS Xenix?
  - DEC abandons PDP-10, MIT shifts to VAX w/Unix
  - MIT launches Project Athena
    - networked workstations to replace Multics
  - 4.2BSD released
  - Stallman announces GNU project
- 1984
  - AT&T divested, Unix becomes commercial product;
    - source code restricted
  - 750 universities have Unix licenses
  - X/Open formed by 5 European computer manufacturers: Bull, ICL, Siemens, Olivetti, and Nixdorf (BISON)
  - MS Xenix System V begins new numbering scheme: IBM calls it IBM Xenix 2.0, SCO calls it SCO Xenix 2.0
  - Earliest SCO copyrights on Xenix 3.x
  - Stallman starts GNU Emacs


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 2015
2016
2017
Unix-like systems
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
SunOS 1 to 1.1
Xenix 3.0
BSD 4.2
System V R1 to R2
SCO Xenix
Unix Version 8
SCO Xenix V/286
System V R3
AIX 1.0
BSD 4.3
SunOS 1.2 to 3.0
HP-UX 1.0 to 1.2
Unix 9 and 10 (last versions from Bell Labs)
SCO Xenix V/386
BSD 4.3 Tahoe
System V R4
HP-UX 2.0 to 3.0
BSD Net/1
SCO Xenix V/386
BSD 4.3 Reno
Minix 1.x
Linux 0.0.1
BSD Net/2
NeXTSTEP/ OPENSTEP 1.0 to 4.0
SunOS 4
386BSD
NetBSD 0.8 to 1.0
HP-UX 6 to 11
Linux 0.95 to 1.2.x
FreeBSD 1.0 to 2.2.x
BSD 4.4-Lite & Lite Release 2
SCO UNIX 3.2.4
UnixWare 1.x to 2.x (System V R4.2)
Solaris 2.1 to 9
NetBSD 1.1 to 1.2
OpenBSD 1.0 to 2.2
OpenServer 5.0 to 5.04
AIX 3.0-7.2
Minix 2.x
Linux 2.x
NetBSD 1.3
FreeBSD 3.0 to 3.2
Mac OS X Server
OpenServer 5.0.5 to 5.0.7
Mac OS X, OS X, macOS 10.0 to 10.12 (Darwin 1.2.1 to 17)
FreeBSD 3.3-11.x
DragonFly BSD 1.0 to 4.8
NetBSD 1.3-7.1
OpenBSD 2.3-6.1
OpenServer 6.x
UnixWare 7.x (System V R5)
Solaris 10
OpenSolaris & derivatives (illumos, etc.)
HP-UX 11i+
Minix 3.1.0-3.4.0
Linux 3.x
Linux 4.x
Solaris 11.0-11.3
OpenServer 10.x

The UNIX Time-Sharing System

- 1985
  - Stallman founds Free Software Foundation
  - POSIX standard (Portable Operating System) to reconcile BSD and AT&T variants
  - X-Windows distributed as free
  - X/Open Portability Guide
  - Intel 80386DX can address 4 gigabyte of memory
- 1986
  - 4.3BSD adds speed tweaks
  - Andy Tanenbaum rewrites Unix from scratch
    - for a teaching system on PC/XT/AT/386

The UNIX Time-Sharing System

- 1988
  - AT&T invests in Sun, threatening Unix dominance
  - Open Software Foundation (IBM, DEC, HP, et al)
    - builds on Mach and IBM's AIX; develops Motif
  - NeXT chooses Mach kernel
- 1989
  - open-source components of 4.3BSD released as Networking Release 1
  - AT&T System V v4.0 unifies Xenix, SunOS, 4.3BSD
  - Bill and Lynne Jolitz conceive free 386BSD

The UNIX Time-Sharing System

- 1990
  - 4.3BSD-Reno includes improved virtual memory via Mach
  - Linus takes first C programming class,
    - learns Digital Unix
- 1991
  - Berkeley includes Jolitz's free-but-incomplete 386 BSD on updated Networking Release 2
  - BSD Networking Release 2 includes all but six Unix files
  - linux 0.01 (64kb) 0.02 0.11

Linux

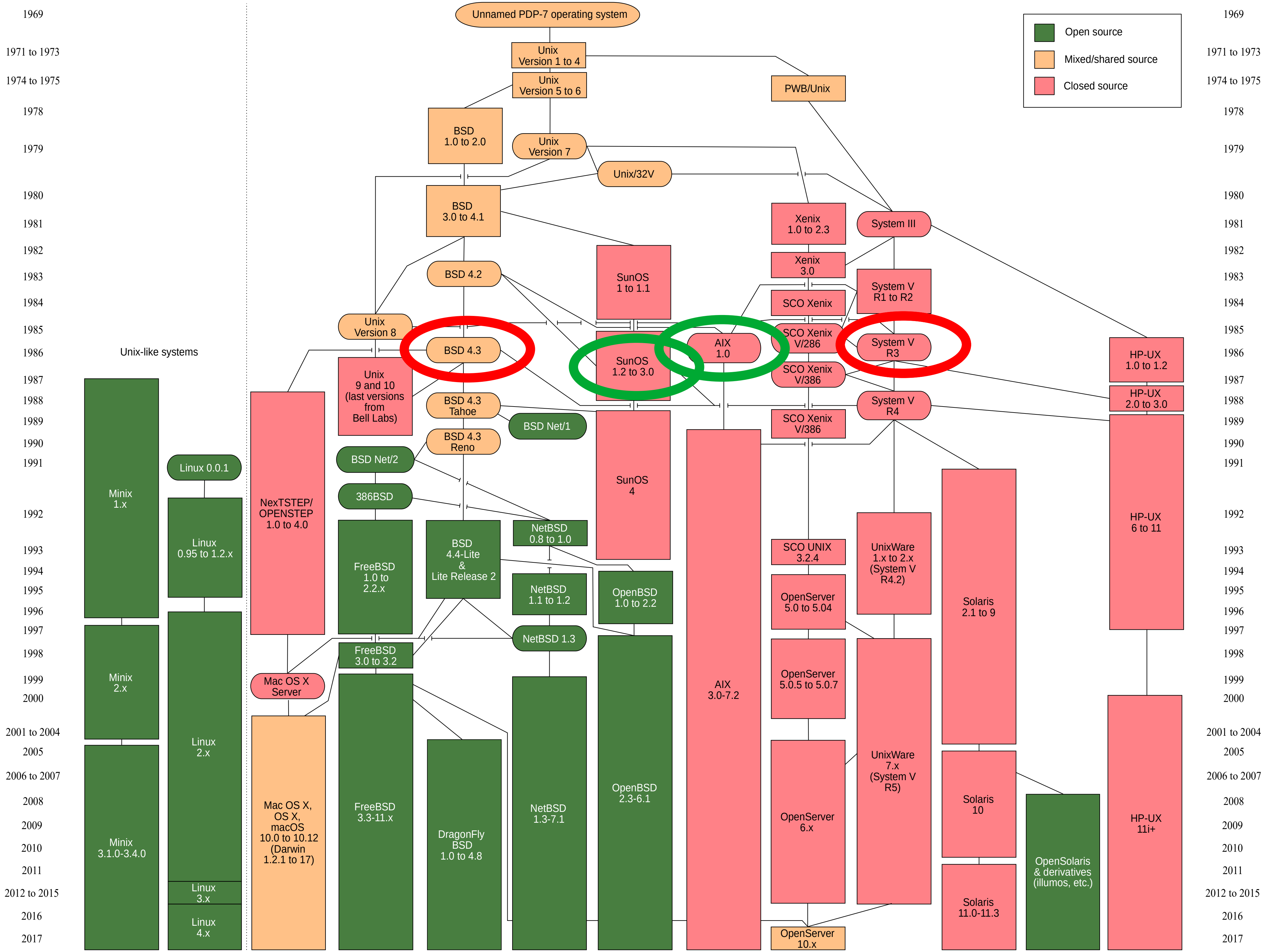
- In 1991, Linus Torvalds announced the Linux project
  - A free Unix kernel for x86 systems
  - Used Gnu tools from the very beginning
- By 1993, Linux had both internet capability and X capability
  - Just in time for the big internet boom

Standartization

June 1986 USENIX conference in Atlanta

- Many AT&T staff wore buttons which read,
  - “System V: Consider it Standard,”
- and a number of major vendors were promoting products based on System V.

- On the other hand, System V did not yet have TCP/IP networking built innd
  - BSD 4.2 did;
- vendors of engineering workstations were nearly all using BSD, and buttons and posters that said
  - “4.2 > V”


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 2015
2016
2017
Unix-like systems
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
SunOS 1 to 1.1
Xenix 3.0
BSD 4.2
System V R1 to R2
SCO Xenix
Unix Version 8
SCO Xenix V/286
AIX 1.0
System V R3
BSD 4.3
SunOS 1.2 to 3.0
HP-UX 1.0 to 1.2
Unix 9 and 10 (last versions from Bell Labs)
SCO Xenix V/386
HP-UX 2.0 to 3.0
BSD 4.3 Tahoe
System V R4
BSD Net/1
SCO Xenix V/386
BSD 4.3 Reno
Minix 1.x
Linux 0.0.1
BSD Net/2
SunOS 4
NeXTSTEP/ OPENSTEP 1.0 to 4.0
386BSD
HP-UX 6 to 11
Linux 0.95 to 1.2.x
FreeBSD 1.0 to 2.2.x
BSD 4.4-Lite & Lite Release 2
NetBSD 0.8 to 1.0
SCO UNIX 3.2.4
UnixWare 1.x to 2.x (System V R4.2)
NetBSD 1.1 to 1.2
OpenBSD 1.0 to 2.2
OpenServer 5.0 to 5.04
Solaris 2.1 to 9
AIX 3.0-7.2
Minix 2.x
NetBSD 1.3
FreeBSD 3.0 to 3.2
Mac OS X Server
OpenServer 5.0.5 to 5.0.7
Linux 2.x
Mac OS X, OS X, macOS 10.0 to 10.12 (Darwin 1.2.1 to 17)
FreeBSD 3.3-11.x
DragonFly BSD 1.0 to 4.8
NetBSD 1.3-7.1
OpenBSD 2.3-6.1
UnixWare 7.x (System V R5)
OpenServer 6.x
Solaris 10
OpenSolaris & derivatives (illumos, etc.)
HP-UX 11i+
Minix 3.1.0-3.4.0
Linux 3.x
Linux 4.x
Solaris 11.0-11.3
OpenServer 10.x

- 1987, AT&T announced that it had purchased a large percentage of Sun Microsystems
  - Sun would receive preferential treatment as AT&T/UNIX Systems Labs developed new software.
- Sun announced that its next system would
  - not be a further extension of SunOS (BSD)
  - but would be derived from AT&T’s System V4.
- DEC, in particular, sensed that AT&T was no longer the benign, benevolent progenitor of UNIX

Open Software Foundation

- January 7, 1988 meeting
  - Apollo, DEC, Gould, Hewlett-Packard, Honeywell-Bull, InfoCorp, MIPS, NCR, Silicon Graphics, UniSoft, Unisys
- semi-secret meetings that included
  - HP, IBM, Bull (France), and Nixdorf and Siemens (Germany)
- May 1988,
  - announced the formation of the Open Software Foundation to be dedicated to the production
    - of an operating system, a user interface, a distributed environment,
    - and free cotton candy.

- 1991–92 the worldwide economy worsened.
  - Bull, DEC, IBM, and the computer side of Siemens all lost money.
  - AT&T resold its share of Sun.
- Sun (ATT) had adopted Motif (OSF);
- 1993 USL sold UNIX to Novell,
- OSF abandoned several of its previously announced products
  - shrink-wrapped software and the distributed management environment);
- Bull, Philips, and Siemens withdrew from sponsorship of OSF.

•1985-93's – FSF and Gnu

- Once you give people something for free, they don't want to have to pay for it
- The Free Software Foundation (1985)
  - Gnu (Gnu's not Unix)
  - Create free versions of popular tools
    - 1986 – gcc, 1987 – most tools
  - Overall goal was to develop a free kernel (It hadn't happened by 1993)
- Squabbling continued and Unix suffered

Free Software Foundation

- Founded by Richard Stallman
  - Wrote original version of gcc and gdb
  - Software should be free, because it should be free
    - As in “free speech, not free beer”
  - Wrote the General Public License (GPL)
    - You are free to do whatever you want as long as the source code goes with it no matter what
    - Controversy: Anything derived from a GPL’d work must itself be GPL’d

Open Source Movement

- Software should be free just because – FSF viewpoint
- Software should be free because free software is better – Open Source Movement viewpoint
  - Every problem can be eliminated if more people look at it

•Linux's Success

- Internet culture and newsgroup postings caused a group of similar minded people to contribute and create a worthwhile kernel
- A competing free attempt had problems
  - The free BSD attempt was mired in a lawsuit (3 files were copied illegally)
  - The Berkeley development group disbanded

OS architectures

- A general purpose OS is composed of:
  - Process manager
    - multiplexes the CPU time between the multiple execution units (processes)
  - memory manager
    - controls, manages and multiplexes the access to physical and virtual memory
  - Inter-process comunication
    - Implements and handles mechanism for processes to comunicate
  - I/O manager
    - manages comunication with perifheral (keyboard/screen, disk, network)
  - User interface
    - command line interpreter
    - GUI

OS organization

- Layered
- Monolithic
- Micro kernel
- Distributed
- VM based

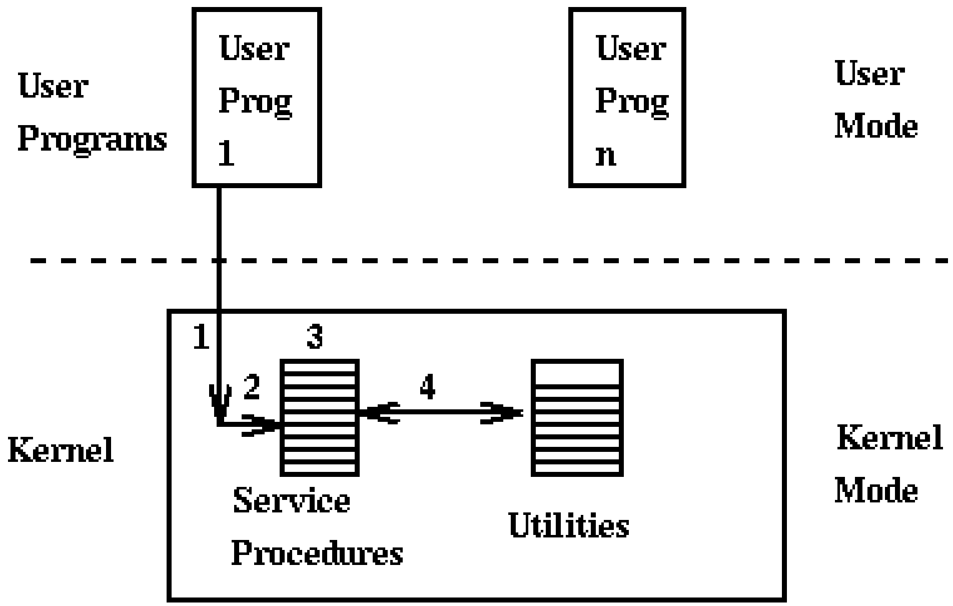
Monolithic architecture

- OS composed of a single module
  - Although using data abstraction
  - Although using layered approach
- All data and code use same memory space
  - low security mechanisms (one driver can mess other drivers)
  - Difficult to evolve (reboot of system needed)
- Easy to implement
- Low overhead

Monolithic architecture

- DOS
- First Unix versions

MONOLITHIC ARCHITECTURE

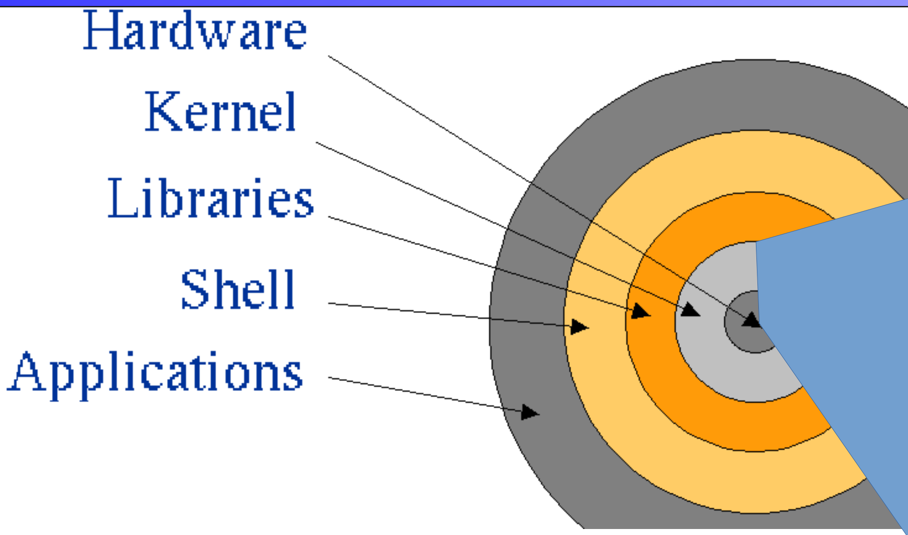


1. System call (User->Kernel Mode)
2. Check parameters
3. Call service routine
4. Service Routine call utilities

Reschedule/Return to user

Layered OS

- Components are divided into layers
  - grouping similar components
- Each layer only interacts with:
  - the bottom layer - requesting services
  - to top layer - answering requests
- Higher level layer
  - Applications
- lowest level layer
  - hardware
- Advantaged
  - good structure, well defined interface, ...
- Disadvantages
  - can be slow, may be difficult to define layers.



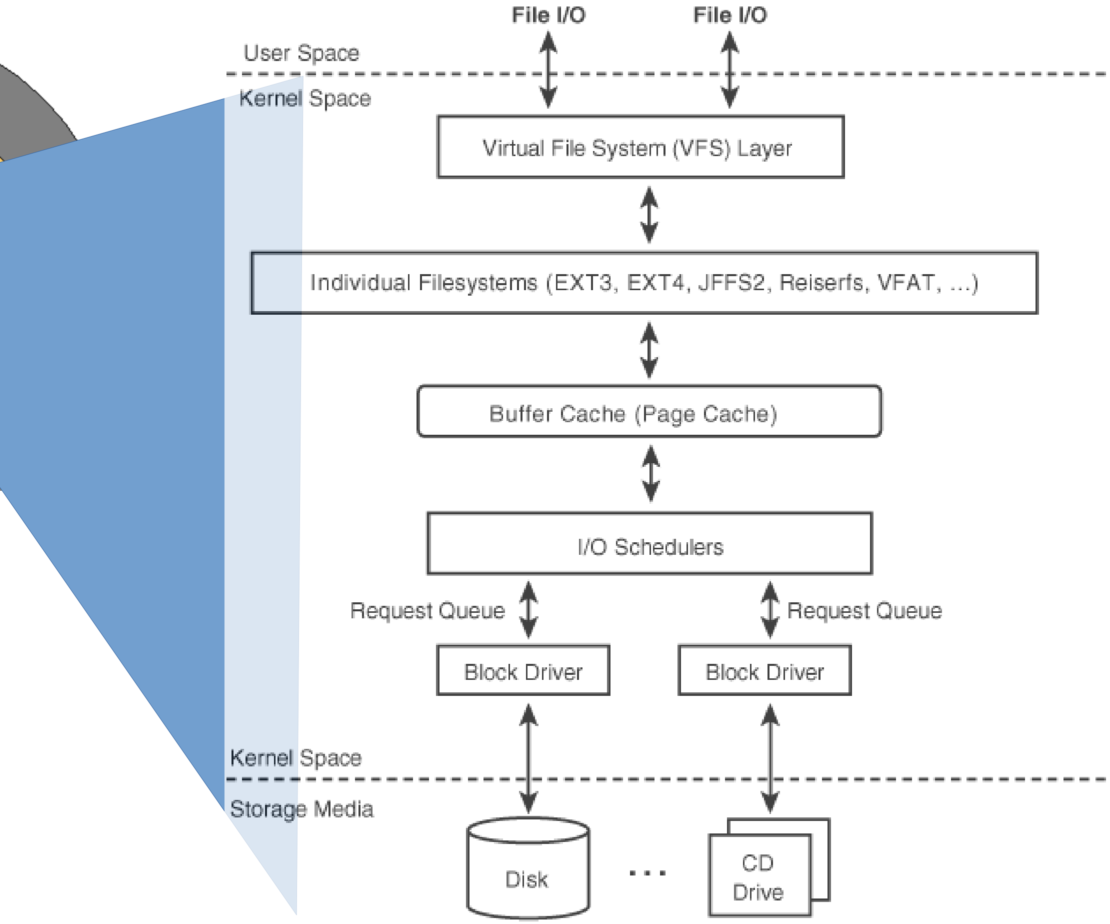
Hardware

Kernel

Libraries

Shell

Applications



File I/O

File I/O

User Space

Kernel Space

Virtual File System (VFS) Layer

Individual Filesystems (EXT3, EXT4, JFFS2, Reiserfs, VFAT, ...)

Buffer Cache (Page Cache)

I/O Schedulers

Request Queue

Request Queue

Block Driver

Block Driver

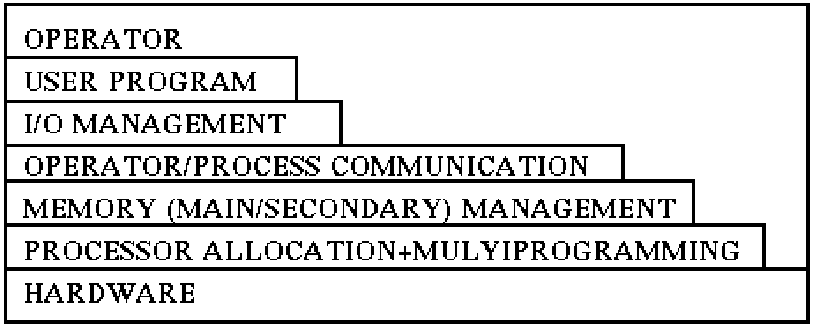
Kernel Space

Storage Media

Disk

. . .

CD Drive



OPERATOR

USER PROGRAM

I/O MANAGEMENT

OPERATOR/PROCESS COMMUNICATION

MEMORY (MAIN/SECONDARY) MANAGEMENT

PROCESSOR ALLOCATION+MULYIPROGRAMMING

HARDWARE

LAYERED SYSTEM (THE System, Dijkstra)

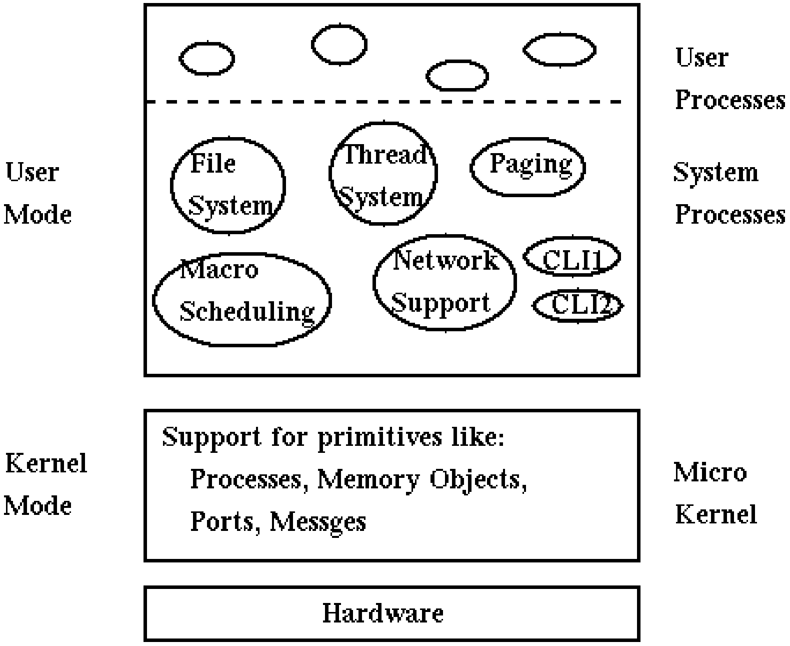
Microkernel

- Removes from kernel as much functionality as possible,
  - limiting the amount of code executed in privileged mode
  - allow easy modifications and extensions
- Most microkernels provide
  - process management
  - memory management
  - message passing between other services
- Security and protection can be enhanced
  - most services are performed in user mode, not kernel mode.
- System expansion can also be easier,
  - only involves adding more system applications, not rebuilding a new kernel.

Microkernel

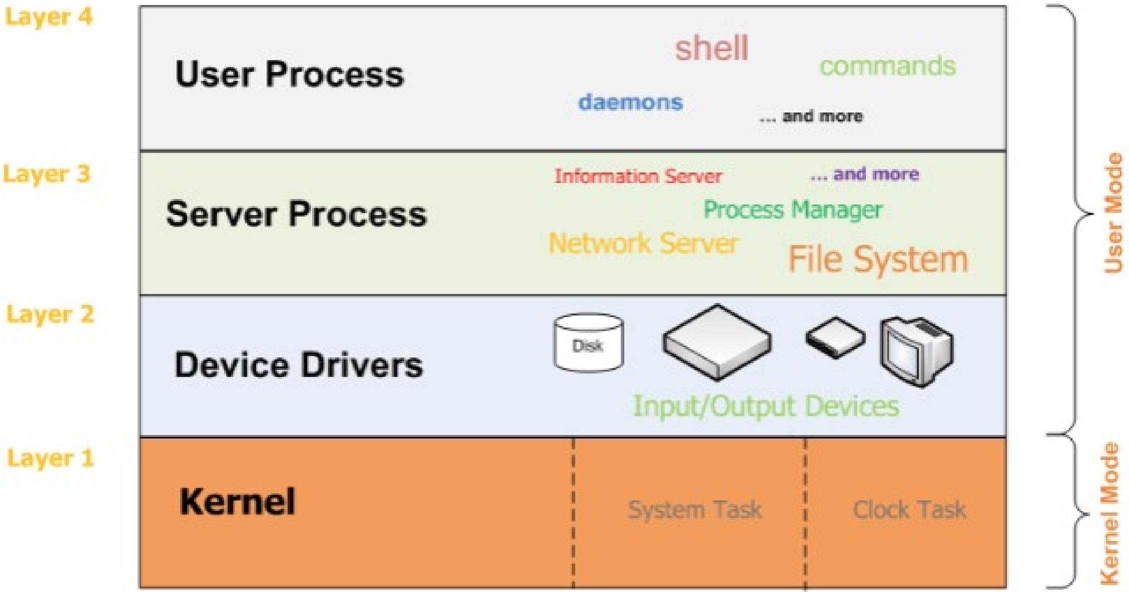
- Windows NT was originally microkernel
  - but suffered from performance problems relative to Windows 95.
  - NT 4.0 improved performance by moving more services into the kernel
- Multiple OSs can be buils on top of a micro-kernel
  - Each operating system will make use of different system processes.

Microkernel

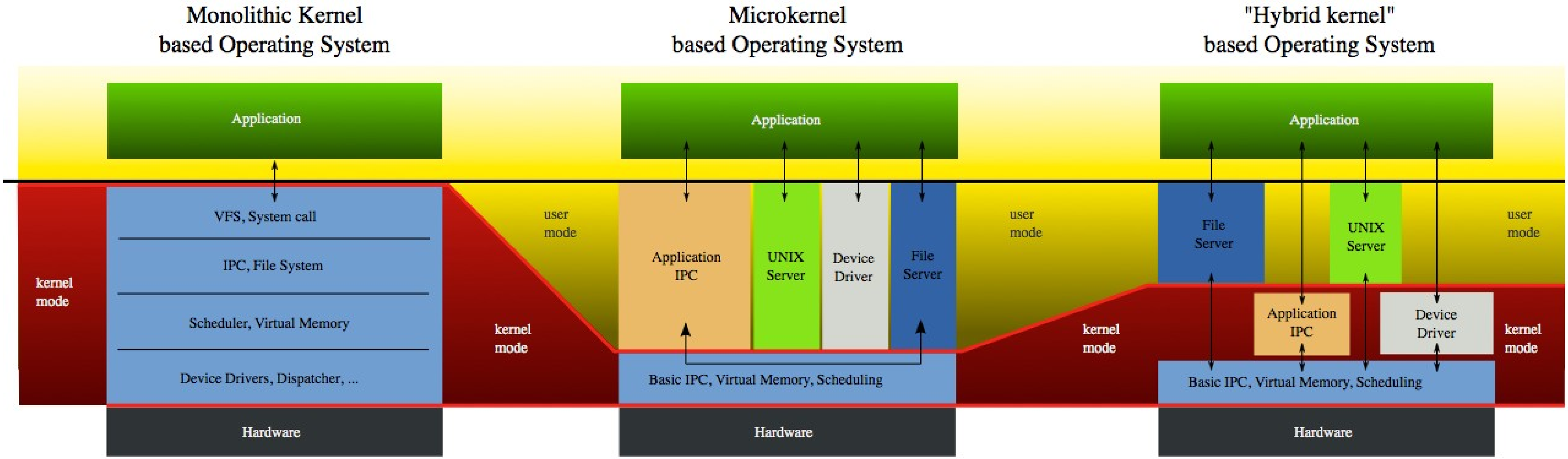


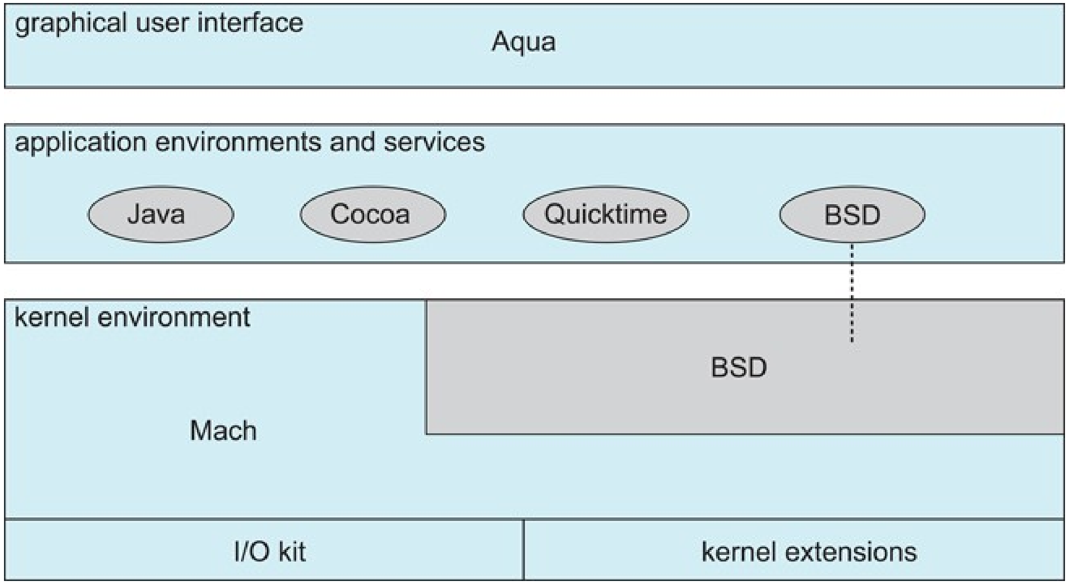
Micro–Kernel Architecture

Minix Layered Micro Kernel Architecture



Hibrid kernel



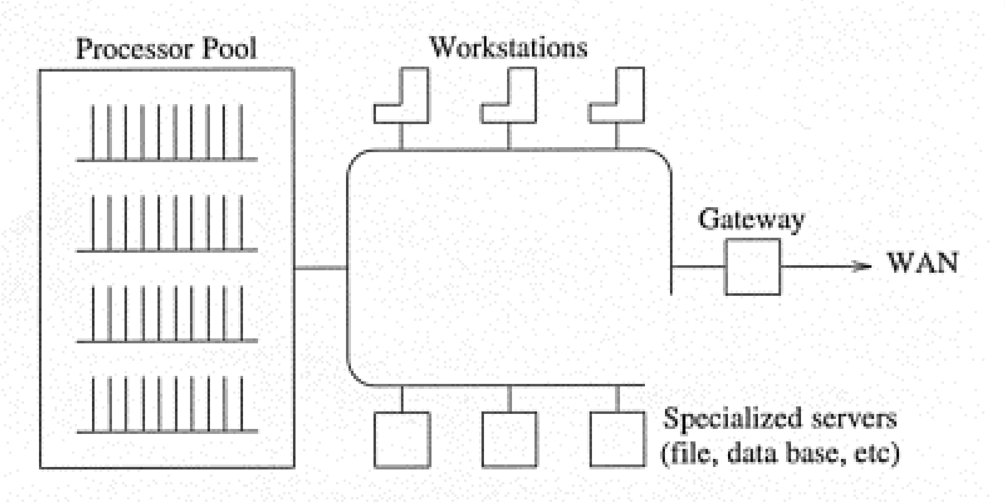


Distributed OS

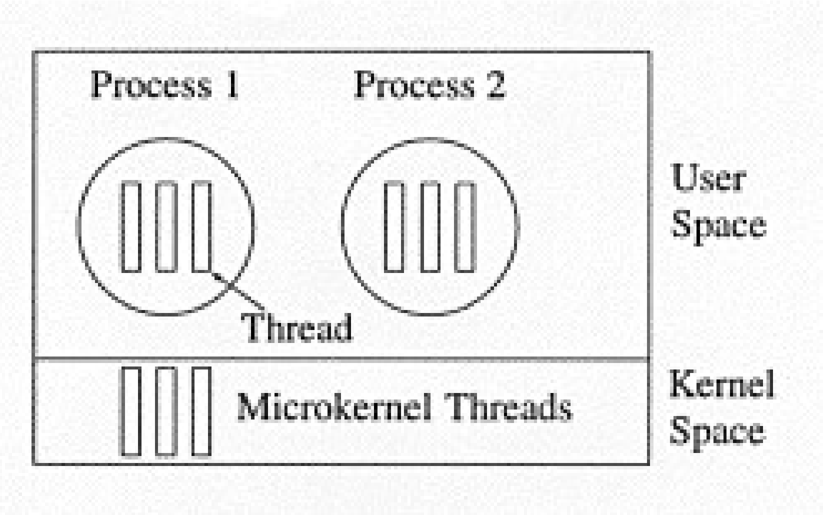
- Each component/service is a separated process
  - on the same machine
  - on different machines
- Components interactions
  - Messages / Remote procedures
- Distributed File System
- Distributed Memory
- Distributed Processes

Amoeba

- The Bullet Server
  - Used for file storage
- The Directory Server
  - Used for file naming
  - Maps from names to capabilities
- The Replication Server
  - Used for fault tolerence and performance
- The Run Server
  - Run server manages the processor pools
- The Boot Server
  - Ensures that servers are up and running
  - If it discovers that a server has crashed,
    - it attempts to restart it, otherwise selects another processor to provide the service



Processor Pool
Workstations
Gateway
WAN
Specialized servers
(file, data base, etc)



Process 1
Process 2
User
Space
Thread
Kernel
Space
Microkernel Threads