



# Message Passing and Microkernels

CSE 536 Spring 2024  
jedimaestro@asu.edu



# Message passing vs. RPC

- Message passing
  - Procedure not directly invoked by name
  - Can be asynchronous, and typically is
  - Multicast and broadcast are pretty natural to the abstraction
  - Models of message passing (like  $\pi$  calculus) incorporate physics, are decentralized
- RPC
  - Have to specify name of procedure you're calling
  - Can be non-blocking, *i.e.*, asynchronous, but typically is not
  - Not really a way to multicast or broadcast
  - Need a central database of [type, instance] pairs

# Two cautionary notes

- Some people consider RPC to be a specific case of message passing
  - Message passing libraries often offer RPC API built on top of message passing
- Message passing means a lot of different things to a lot of different people
  - Object oriented people ... It's about modularity
  - Distributed computing people ... It's a beautiful model of concurrency
  - Microkernel people ... It's about robustness

# [https://en.wikipedia.org/wiki/Message\\_passing](https://en.wikipedia.org/wiki/Message_passing)

- “In computer science, message passing is a technique for invoking behavior (*i.e.*, running a program) on a computer.”
- “The invoking program sends a message to a process (which may be an actor or object) and relies on that process and its supporting infrastructure to then select and run some appropriate code.”
- “Message passing differs from conventional programming where a process, subroutine, or function is directly invoked by name.”

# Synchronous message passing

- When two objects are running at the same time, *e.g.*, in Java or Smalltalk
- “Synchronous messaging is analogous to a synchronous function call; just as the function caller waits until the function completes, the sending process waits until the receiving process completes.”
- *E.g.*, Circle, Square, and Rectangle are subclasses of Shape, send any Shape a message to calculate its own area

Isn't that just object oriented programming with  
polymorphism?

# Asynchronous message passing

- “With asynchronous message passing the receiving object can be down or busy when the requesting object sends the message.”
- “Continuing the function call analogy, it is like a function call that returns immediately, without waiting for the called function to complete.”
- Requires storing and retransmitting data
- Buffer gets full?
  - Block (can deadlock), *--or--*
  - Drop messages

Another advantage of message passing is  
multicast and broadcast...



```
//scatter rows of first matrix to different processes
MPI_Scatter(a, N*N/size, MPI_INT, aa, N*N/size, MPI_INT, 0, MPI_COMM_WORLD);

//broadcast second matrix to all processes
MPI_Bcast(b, N*N, MPI_INT, 0, MPI_COMM_WORLD);

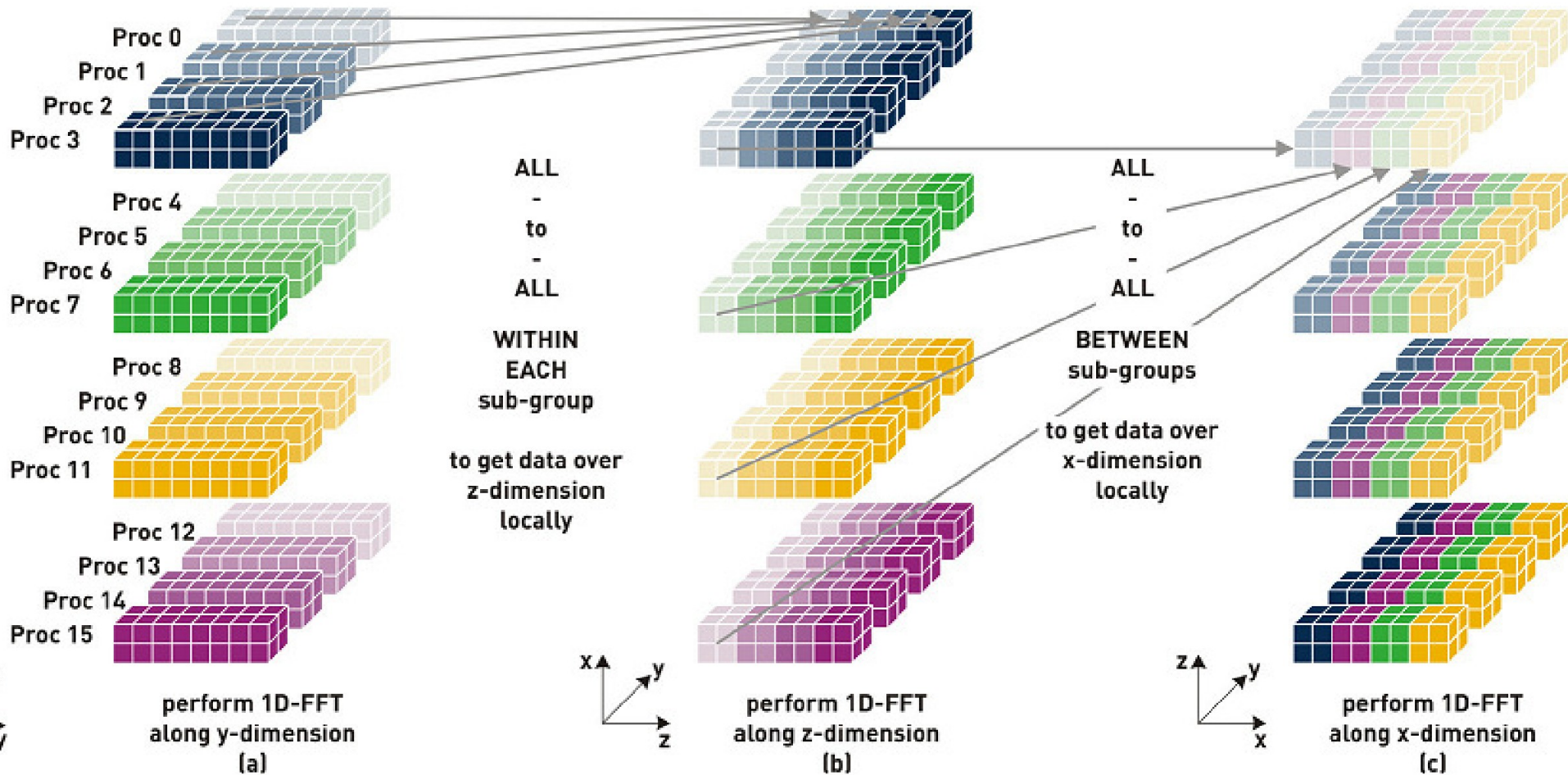
MPI_Barrier(MPI_COMM_WORLD);

//perform vector multiplication by all processes
for (i = 0; i < N; i++) {
    for (j = 0; j < N; j++) {
        sum = sum + aa[j] * b[j][i];
    }
    cc[i] = sum;
    sum = 0;
}

MPI_Gather(cc, N*N/size, MPI_INT, c, N*N/size, MPI_INT, 0, MPI_COMM_WORLD);

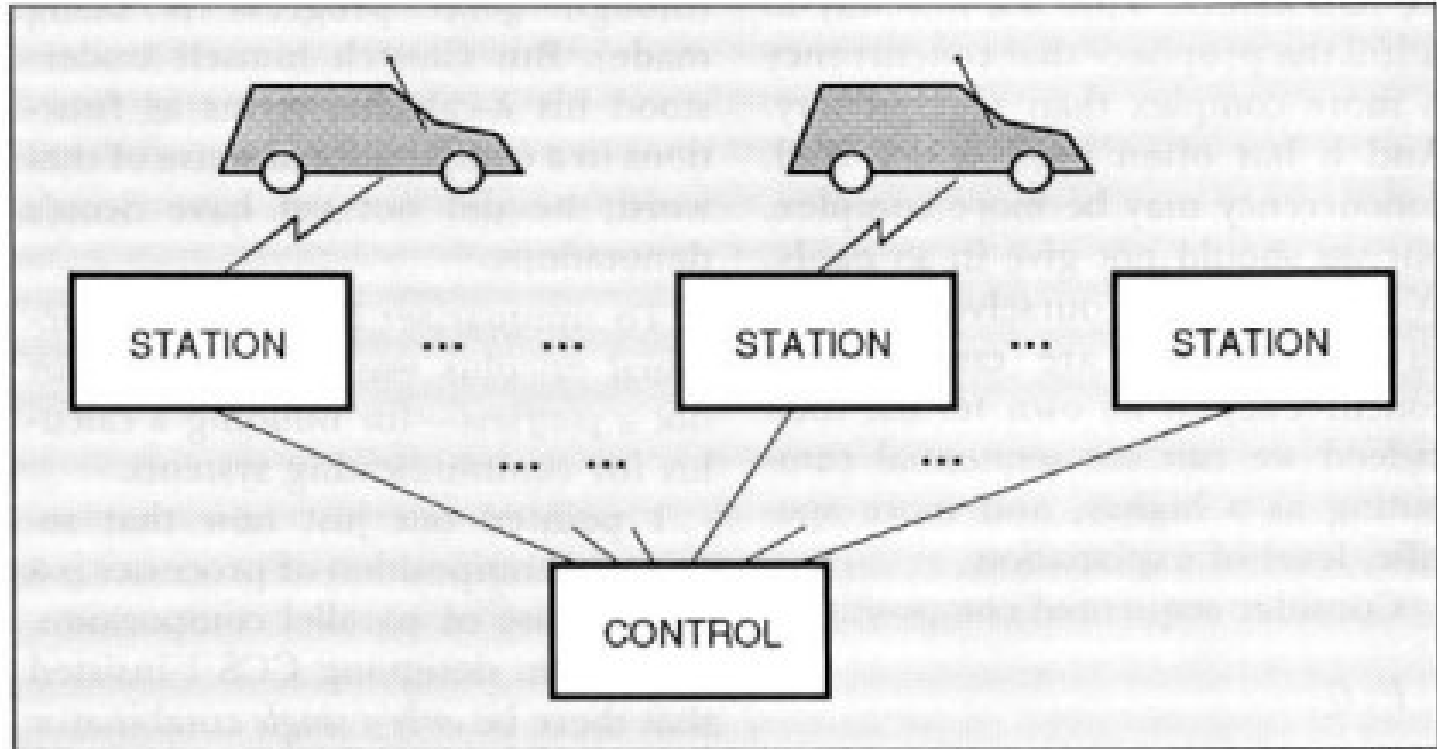
MPI_Barrier(MPI_COMM_WORLD);
MPI_Finalize();
```

<https://stackoverflow.com/questions/41575243/matrix-multiplication-using-mpi-scatter-and-mpi-gather>



Another advantage of message passing is  
mobility...

Can we use semaphores, mutexes, *etc.* for this?



<https://dl.acm.org/doi/pdf/10.1145/151233.151240>

# $\pi$ calculus

names  $x, y, z, \dots$

action terms  $A ::= \bar{x}z.P$   
 $xy.P$

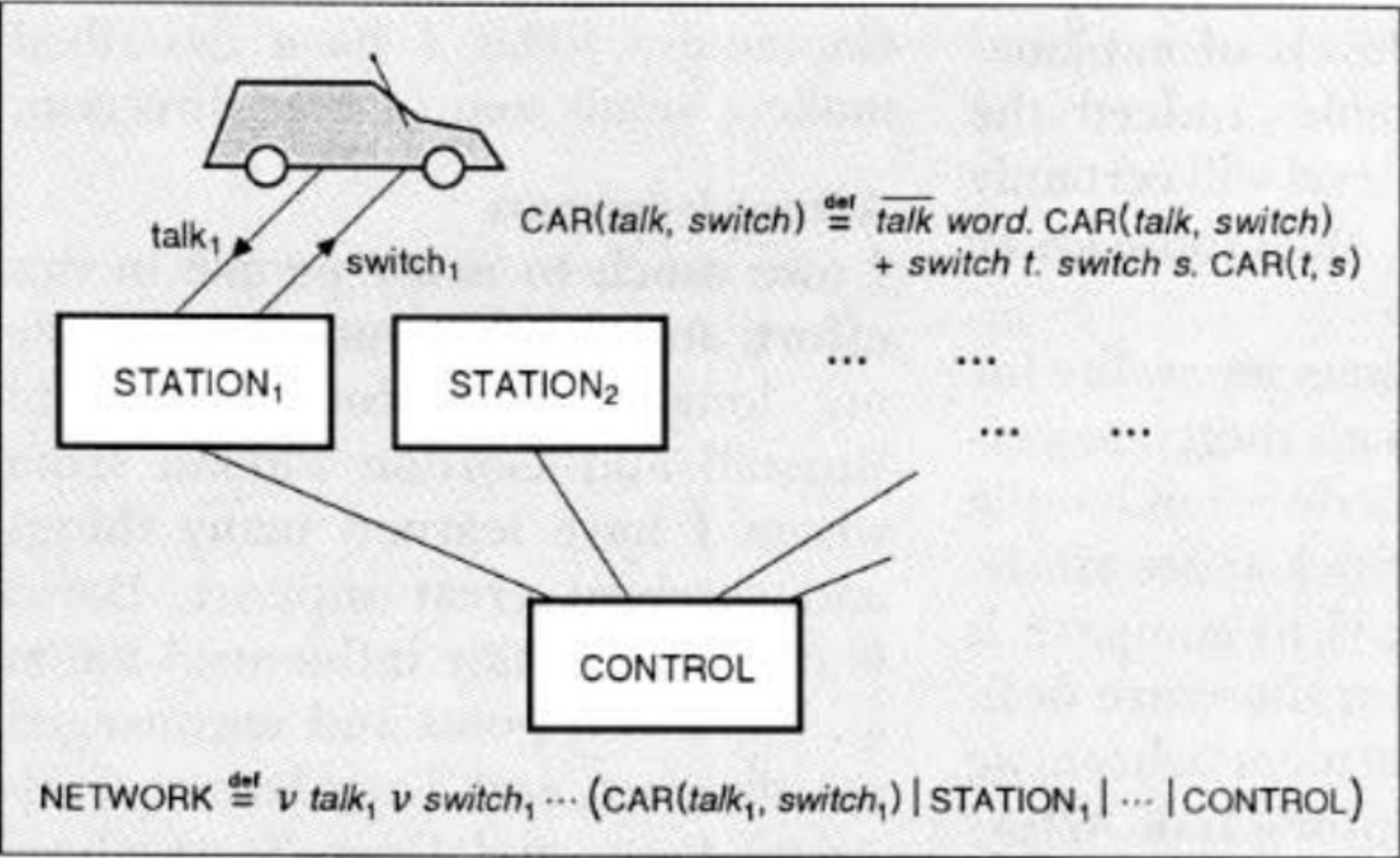
send  $z$  along  $x$   
receive any  $y$  along  $x$

terms  $P ::= A_1 + \dots + A_n$   
 $P_1 \mid P_2$   
 $\nu y.P$   
 $!P$

alternative action ( $n \geq 0$ )  
composition  
restriction  
replication

*(the occurrences of  $y$  are binding)*

basic rule of computation  $xy.P_1[y] \mid \bar{x}z.P_2 \rightarrow P_1[z] \mid P_2$



# In operating systems...

- Message passing is an Interprocess Communication (IPC) mechanism
- Special semantics and memory protection
  - Ordering of events is based on messages
    - No need for mutexes, shared memory and semaphores, *etc.*
  - Multicast and broadcast
  - Security and reliability benefits?



A little MacOS history...



[https://en.wikipedia.org/wiki/Altair\\_8800](https://en.wikipedia.org/wiki/Altair_8800)  
(1974)



[https://en.wikipedia.org/wiki/Apple\\_I](https://en.wikipedia.org/wiki/Apple_I)  
(1976)



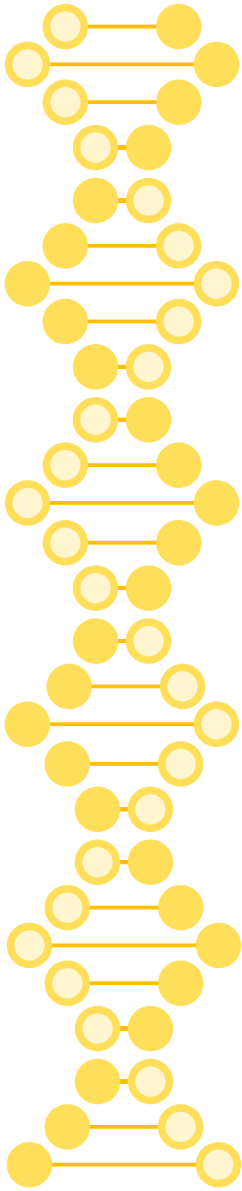
Because the Apple I did not include a case, customers needed to supply their own.

[https://en.wikipedia.org/wiki/Apple\\_II](https://en.wikipedia.org/wiki/Apple_II)  
(1977)



[https://en.wikipedia.org/wiki/Apple\\_Lisa](https://en.wikipedia.org/wiki/Apple_Lisa)  
(1983)





MEXICO'S EARTHQUAKE TRAGEDY

# Newsweek

THE WEEKLY NEWS MAGAZINE

## A WHIZ KID'S FALL



**How Apple Computer Dumped Its Chairman**

**Shooed out in Silicon Valley**

**An Exclusive Interview With Steve Jobs**

# STEVE JOBS: OUT FOR REVENGE

By Paul Patton



Jobs at his NeXT factory in Fremont, Calif. Each circuit board is assembled "unfurnished by human hands."

He has named his computer NeXT. "What we want," he tells the audience, "is to create the next computing revolution. We want to push the envelope." The same NeXT means his claim to the new standard in the industry—a PC with unprecedented power and versatility and an innovative programming system—but it is also an unqualified reference to continuity about the next chapter in the story of Steve Jobs.

In 1976, at the age of 31, Silicon Paul Jobs co-founded Apple Computers with Stephen G. Wozniak, five years his senior, whom Jobs had known since he was a sophomore at Menlo Park High School, in California's Silicon Valley. Within five years, Apple had become a billion-dollar company. Then in 1985 Jobs was booted out—by John Sculley, whom Jobs himself had hired two years before to be the company's chief executive. Ever since, working in silence, Jobs had been preparing a comeback. Now, at age 34, no longer the boy-wonder of the computer industry, he was starting over.

IN SILICON VALLEY, A COMPUTER IS CALLED A "box," a sign that the guts may be less important than the skin. The guts of Jobs's new machine are housed in a ribbed black magnesium case. Keyboard and monitor are separate, connected by cables, the 17-inch screen dramatically cantilevered over a swiveling support. "Computers," Jobs likes to say, "are the metaphor of our time. They should have a certain higher aesthetic."

Not that what is inside the NeXT box is unimportant: a Paul Patton is at work on a book about American design,

new special memory system that uses a laser to move and read up to 200 volumes' worth of information on a single disk, a sound system of CD quality, a powerful array of sophisticated processing chips, and intensive software.

"What other computer can you sit down to," he says, referring to the "Digital Libraries" feature of the NeXT machine, "and blast through the complete works of William Shakespeare in just a couple of seconds?"

Although it looks like a personal computer, the NeXT machine is much more powerful than any PC on the market. It has the capabilities of computers known as workstations, previously used mostly by engineers and scientists. Like most workstations, it employs Unix, an aging but powerful basic software system. But as EISA it costs much less than most workstations of comparable power.

Business has traditionally favored the Unix system, and it still accounts for just 8 percent of the computer market today. But, thanks in part to NeXT, Unix is expected to move from double to triple its share in the next five years. Jobs has exceeded its expectations in a new variation of the software, a "user shell" that will make it, he says, "usable by more mortals."

Called NeXTing, the software employs a "finder" key approach that allows novices as well as experts to combine pre-existing sections of instructions, or "objects," to create the programs they need—an innovative technique known in the industry as "object-oriented programming."

What may be most revolutionary about NeXT, however, is not its technology but the fact that it is the first computer to be sold primarily on the strength of its design.

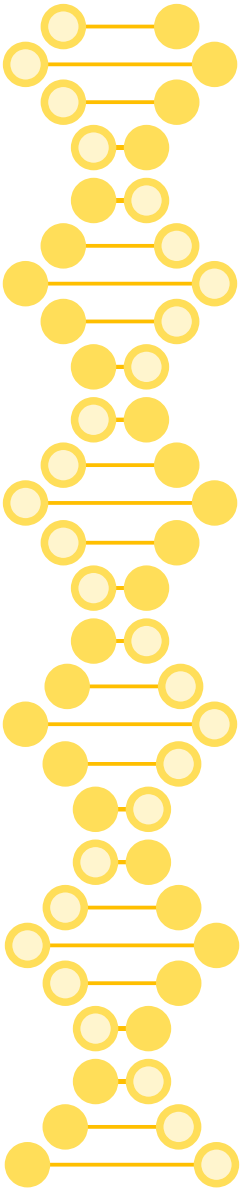
Critics and champions alike—and Jobs has plenty of each—agree that he has always been at his best as a salesman and evangelist for the computer. Former Intel man, who worked with Jobs at Apple and at NeXT, says, "In some ways, Steve gets philosophical the way the Greeks did. He always shapes the best. He has these aesthetic notions of perfect shapes and perfect sounds. It is almost Platonic."

Jobs's obsession with detail, with appearance, is part of his legend. When a small imperfection showed up on the first samples of the computer's case, he flew to Chicago to work with the die maker. At NeXT's automated factory in Fremont, Calif., he had the machines repeatedly re-programmed to achieve the uniform gray he wanted.

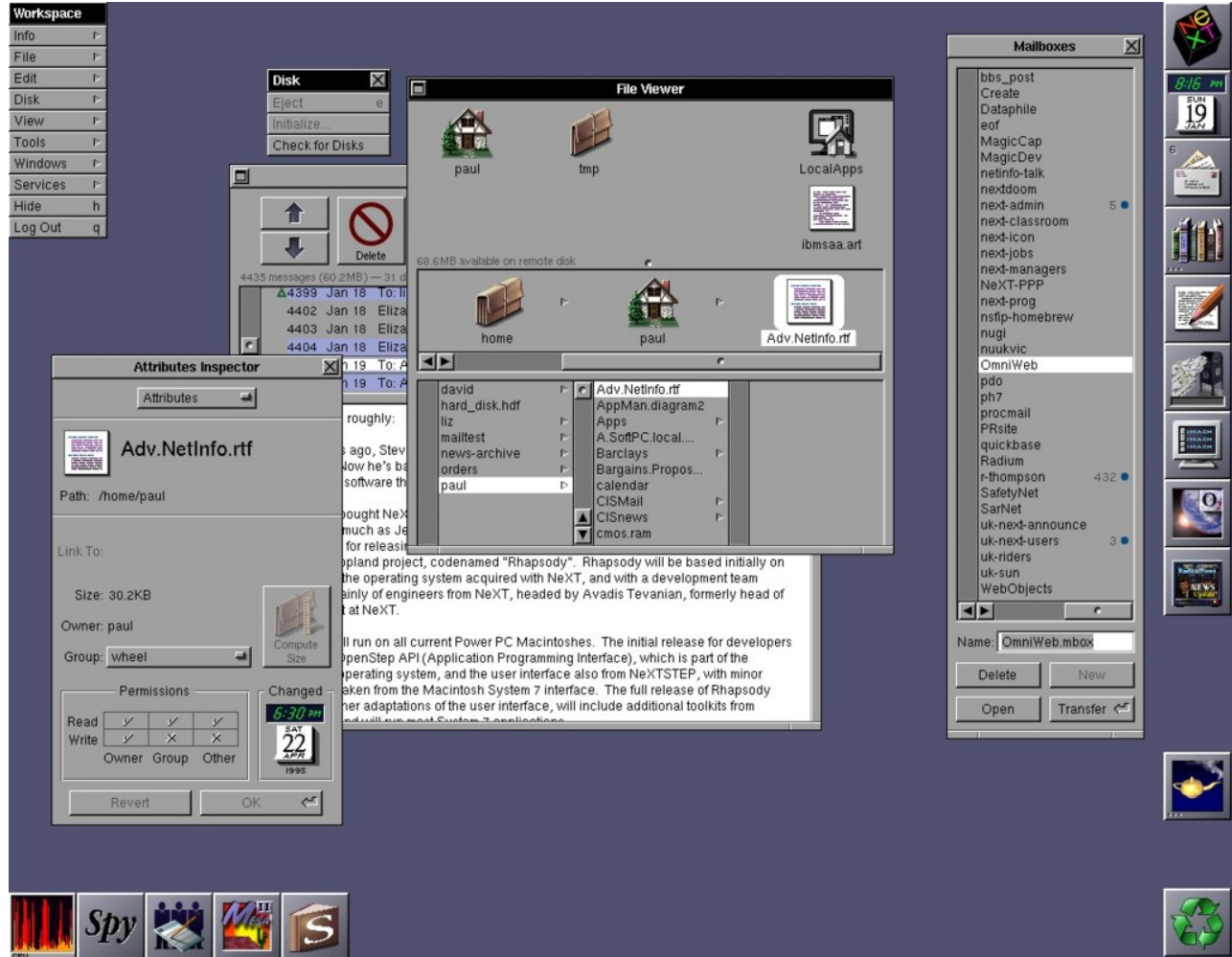
Colleagues may use words like Platonic to describe his aesthetic but for Jobs the technology is "laser" and "whizy"—as opposed to "laser" and "bram-damaged." He has never claimed to be an is- (Continued on Page 12)

Ousted from Apple, the computer legend is back on the fast track with a \$10,000 PC loaded with power.

<https://en.wikipedia.org/wiki/NeXTcube>  
(1990)



# <https://en.wikipedia.org/wiki/NeXTSTEP>





# Have you heard of any of these?

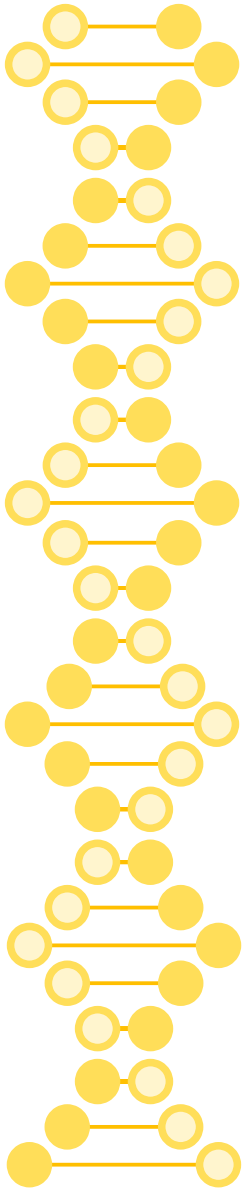
- Pixar
- Doom
- Quake
- The World Wide Web





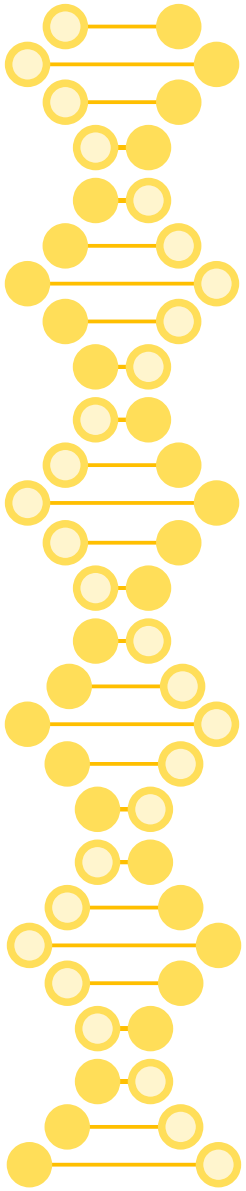
30 AMMO    100% HEALTH    2 3 4 ARMS    88% ARMOR    88 / 200    30 / 50    6 / 50    260 / 300





# NextSTEP

- Combination of Mach and FreeBSD
- Objective-C
- An object oriented application layer known as “kits”
- The “Dock” in the GUI
- First app store



# [https://en.wikipedia.org/wiki/Mach\\_\(kernel\)](https://en.wikipedia.org/wiki/Mach_(kernel)) (1985 to 1994)

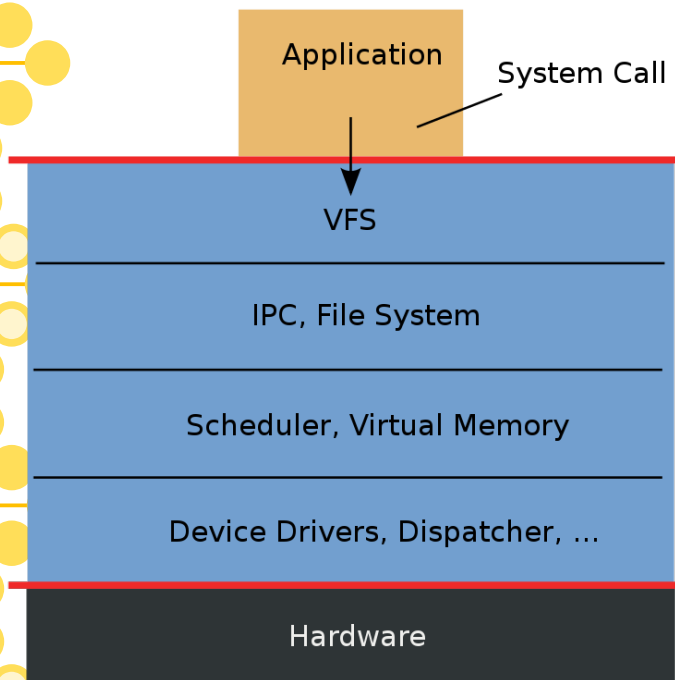
- Microkernel replacement for UNIX
- Started as a monolithic kernel and evolved into a microkernel
- Basis for...
  - GNU Hurd
  - XNU
    - macOS, iOS, iPadOS, tvOS, and watchOS

# UNIX in the 1980s

- Everything is all about pipes
- Networking, device drivers, *etc.*
  - A lot of complexity being added
- Aleph kernel at Univ. of Rochester ... OS is modular and communicates over pipes
  - Added shared memory
- Mach based on message passing

<https://en.wikipedia.org/wiki/Microkernel>

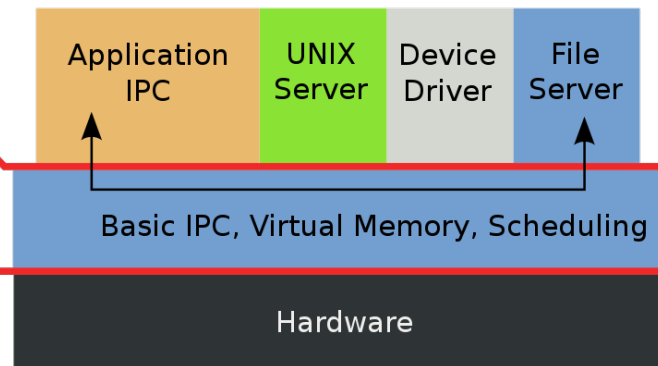
## Monolithic Kernel based Operating System



## Microkernel based Operating System

user mode

kernel mode





# Supposed benefits of microkernels

- More modular
  - A crash in part of the OS doesn't crash the system?
  - Plug and play parts of the OS?
- Makes multiple CPUs and distributed computing easier
  - Based on message passing
- More secure?
  - Filesystem, *etc.* can be in userspace
  - What about transitivity?

# Truth about microkernels

- Context switches will always be expensive
  - TLB flushes, virtualization
- As far as I can tell, the Mach messaging layer in MacOS is there for historical reasons only
- The most impressive things about “message passing” on MacOS are object-oriented stuff happening in the GUI (AFAIK --- prove me wrong)

# Message passing is still important

- Easy way to do IPC and concurrency
  - Advantages over...
    - Pipes?
    - Stream sockets?
    - Datagram sockets?
    - Shared memory?
  - Beautiful academic theories if you need them
    - Mobility
- Supercomputing
  - RPC can't broadcast and multicast