

## CS134a: Modern OS

- An interesting point of view
- Two rounds of research
- 1980's: modularity and micro-kernels
  - Mach (CMU; Rashid, ...)
  - Amoeba (Vrije; Tannenbaum, ...)
  - X-Kernel (U. Arizona; Peterson, ...)
  - V (Stanford; Cheriton, ...)
- 1990's: minimalism
  - ExoKernel (MIT; Kaashok, ...)
  - Vino (Harvard; Seltzer, ...)
  - SPIN (U. Washington; Bershad, ...)



---

---

---

---

---

---

---

---

## Systems Software Research is Irrelevant

- Rob Pike from Bell Labs (02/21/ 00)
- “This talk is a polemic that distills the pessimistic side of my feelings about systems research these days.[...] I think the situation is genuinely bad and requires action.”
- “Systems software research has become a sideline to the excitement in the computing industry.”
- “When did you last see an exciting noncommercial demo?”



---

---

---

---

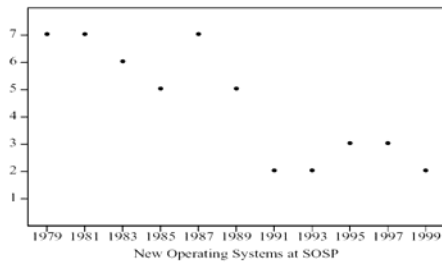
---

---

---

---

## Systems Software Research is Irrelevant (2)



- SOSP (Symposium on Operating Systems Principles)



---

---

---

---

---

---

---

---

### Systems Software Research is Irrelevant (3)

- Who needs new operating system, anyway?
- There are a lot of papers in file systems, performance, security, etc.



---

---

---

---

---

---

---

---

### Systems Software Research is Irrelevant (4)

1990	2000
<i>Hardware</i>	
33 MHz Mips R3000	600 MHz Alpha or Pentium III
32 megabytes of RAM	512 megabytes of RAM
10 Mbps Ethernet	100 Mbps Ethernet
<i>Software</i>	
Unix	Unix
X Windows	X Windows
Emacs	Emacs
TCP/IP	TCP/IP
	Netscape
<i>Language</i>	
C	C
C++	C++
	Java
	Perl (a little)



---

---

---

---

---

---

---

---

### Systems Software Research is Irrelevant (5)

- Q: Where is the innovation?
- A: Microsoft (compare 1990 Microsoft software with 2000)
- Claim: Microsoft was not innovative but a good copier
- Counter-claim: Java is to C++ what Microsoft was to MacOS
- Linux
- Innovation?
- No, it's just another Unix
- *The art is gone*



---

---

---

---

---

---

---

---

## Systems Software Research is Irrelevant(6)

- In the 80s much system work revolved around new architectures (RISC, Lisp Machines, etc.) . No more
- Portability was also an issue. Now hardware is the same.
- To be a viable computer system, one must honor a huge list of large, and often changing standards. Little space for novelty
- Graduating PhDs use Unix, X, Emacs and Tex. Long time ago © students were exposed to a wide variety of OS.
- Narrowness of experience – narrowness of imagination.
- Change of scale



---

---

---

---

---

---

---

---

---

---

## Systems Software Research is Irrelevant(7)

- What to do?
  - Go and *build* systems.
  - Be courageous. Experiment. Try to give a cool demo.
  - Make industry *want* your work
- What to build?
  - GUIs (but not Windows like)
  - Distributed systems
  - Languages and Compilers design
- **YOU** can make the difference.



---

---

---

---

---

---

---

---

---

---

## CS134a: Modern OS

- ✓ An interesting point of view
- Two rounds of research
- 1980's: modularity and micro-kernels
  - Mach (CMU; Rashid, ...)
  - Amoeba (Vrije; Tannenbaum, ...)
  - X-Kernel (U. Arizona; Peterson, ...)
  - V (Stanford; Cheriton, ...)
- 1990's: minimalism
  - ExoKernel (MIT; Kaashok, ...)
  - Vino (Harvard; Seltzer, ...)
  - SPIN (U. Washington; Bershad, ...)



---

---

---

---

---

---

---

---

---

---

## 1980's: microkernels

- Simplify the kernel
- Add support for multi-processor, distributed computing
- Unix/Windows are *monolithic* kernels



---

---

---

---

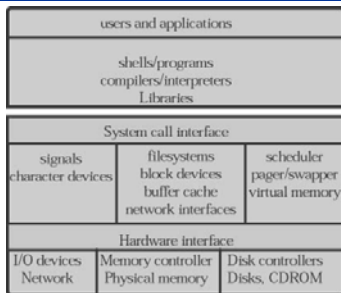
---

---

---

---

## Unix architecture



---

---

---

---

---

---

---

---

## Mach (CMU: Rashid, Bershad, ...)

- Evolved out of BSD4.2
- Provides compatibility with
  - 4,3 BSD
  - Mac
  - OS/2
  - ...



---

---

---

---

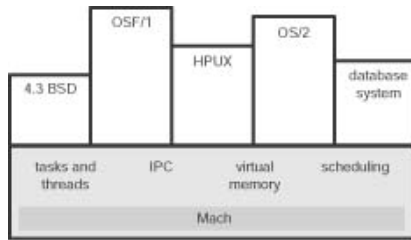
---

---

---

---

## Mach architecture



Mach 3 structure.



---

---

---

---

---

---

---

---

---

---

## Mach design goals

- Support for diverse architectures, including multiprocessors
  - UMA: uniform memory access (like x86 SMP)
  - NUMA: non-uniform memory access (large numbers of processors)
  - NORMA: no remote memory access (Beowulf)
- Transparent access to network resources
- Exploit parallelism in both system and applications
- Simplified kernel; small number of abstractions
- Provide a base for building other OS (e.g. UNIX)



---

---

---

---

---

---

---

---

---

---

## Mach design goals

- Distributed operation
  - Heterogeneous (multiple machine types)
- Includes memory management and communication



---

---

---

---

---

---

---

---

---

---

## Mach abstractions

- A *task* is an execution environment
- A *thread* is the basic execution unit that runs within a task
  - All threads in a task share the same memory space and resources
- A *port* is a generic reference to an object
  - Implemented as a buffered communication channel
- A *port set* is a group of ports sharing a message queue
- A *message* is the unit of communication between threads
- A *memory object* is a source of memory: a memory manager, or a file on a file server, ...



---

---

---

---

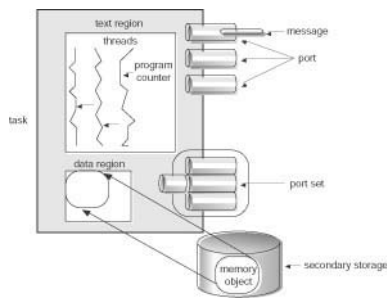
---

---

---

---

## Mach abstractions



---

---

---

---

---

---

---

---

## Mach processes

- Threads are either Running or Suspended
- Multiprocessor scheduler
  - Quantum varies with number of threads in the system
- *Spinlocks* are used for mutual exclusion
- Some threads are exception handlers



---

---

---

---

---

---

---

---

## Mach memory management

- There is a *pageout* kernel thread: FIFO with second-chance
- Memory is managed by a *memory server*
  - A page fault generates a request to the server
  - A pageout copies the page to the server, which probably writes it to a file
  - Memory managers are *user programs*



---

---

---

---

---

---

---

---

## Mach interface

- The systems primitives (like message-passing) are very primitive
- Mach includes a compiler: the MIG interface generator
- Kernel uses run-time code generation



---

---

---

---

---

---

---

---

## CS134a: Modern OS

- ✓ An interesting point of view
- ✓ Two rounds of research
- ✓ 1980's: modularity and micro-kernels
  - ✓ Mach (CMU; Rashid, ...)
    - Amoeba (Vrije; Tannenbaum, ...)
    - X-Kernel (U. Arizona; Peterson, ...)
    - V (Stanford; Cheriton, ...)
- 1990's: minimalism
  - ExoKernel (MIT; Kaashok, ...)
  - Vino (Harvard; Seltzer, ...)
  - SPIN (U. Washington; Bershad, ...)



---

---

---

---

---

---

---

---

## Amoeba (Tannenbaum)

- Processors are cheap
- A distributed system should look like a single machine
- Users should not be aware of:
  - How many machines they are using
  - The locations of the machines
  - Where their files are stored
- The OS should provide the illusion of a single system, while efficiently managing resources



---

---

---

---

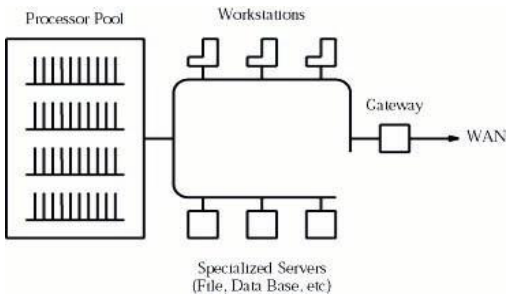
---

---

---

---

## Amoeba system model



---

---

---

---

---

---

---

---

## Amoeba system model

- Pool processors
  - Not owned by any user
  - Processes are dynamically assigned to processors
  - Individual processors are time-shared
  - No shared memory
- Workstations are essentially X-terminals



---

---

---

---

---

---

---

---

## Amoeba micro-kernel

- Amoeba software contains
  - a micro-kernel for each processor
  - a set of servers
- The micro-kernel has four functions
  - Manage processes, and threads within processes
  - Provide low-level memory management
  - Support transparent communication between any threads
  - Handle I/O



---

---

---

---

---

---

---

---

## Amoeba processes and threads

- Processes have the traditional model (memory, stack, PC, ...)
- Threads are processes that share the same address space
  - Managed by the kernel
  - Allows threads to block
- Memory management: segmentation



---

---

---

---

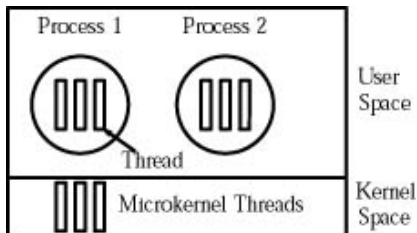
---

---

---

---

## Amoeba processes and threads



---

---

---

---

---

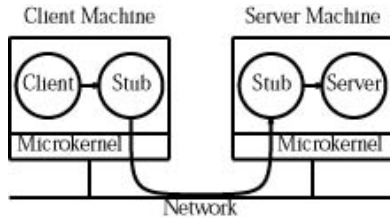
---

---

---

## Amoeba communication

- RPC: remote procedure call
- Includes all thread-thread calls (user to user, user to kernel, ...)



---

---

---

---

---

---

---

---

---

---

## Amoeba

- A server allocates a *port*
- RPC request broadcasts a LOCATE message, and a server replies
- RPC primitives
  - do\_remote\_op: user -> server request
  - get\_request: server listens on a port
  - do\_reply: server's response to a request



---

---

---

---

---

---

---

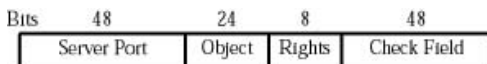
---

---

---

## Amoeba servers

- An object is a piece of abstract data on which operations can be performed (it is an abstract data type)
- Objects are *named* with *capabilities*; there are servers that give out capabilities



---

---

---

---

---

---

---

---

---

---

## Amoeba capabilities

- When a client wants to access an object
  - It traps to the kernel
  - Kernel locates the server port (it maintains a cache)
  - Kernel passes the capability to the server
- Rights determine which operations can be performed
- Check field prevents the capability from being modified



---

---

---

---

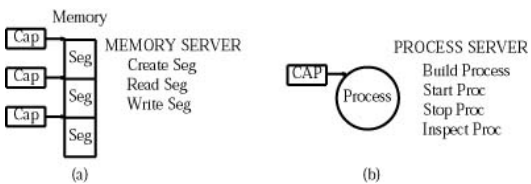
---

---

---

---

## Amoeba capabilities



---

---

---

---

---

---

---

---

## Amoeba filesystems

- Filesystems are user-defined servers
- Any number of filesystems can be defined and used
- *Bullet* fileserver
  - Files are *immutable!*
  - Two operations: CREATE (returns a new capability), and READ
  - Contiguous allocation (fast)



---

---

---

---

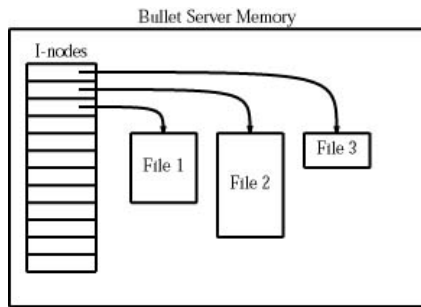
---

---

---

---

## Amoeba Bullet fileserver



---

---

---

---

---

---

---

---

## Amoeba directory server

- Maps names to capabilities
- Directory contains (name, capability-set) entries
  - Not immutable
- Columns indicate recipient rights (user, group, other)



---

---

---

---

---

---

---

---

## Amoeba directory server

	Column 1	Column 2	Column 3
File6	<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>	<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>	<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>
File5	<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>	<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>	<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>
File4	<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>	<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>	<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>
File3	<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>	<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>	<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>
File2	<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>	<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>	<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>
File1	<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>	<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>	<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>

Capabilities for replicated files



---

---

---

---

---

---

---

---

## Amoeba programming support

- Coda language
- Native multi-threading support
  - `fork int loop() { ... }`
- Synchronization:
  - distributed monitors
  - Conditional regions
    - \* `guard P -> { code... }`
- Atomic multicast



---

---

---

---

---

---

---

---