The Mach System

Some arguments

Microkernel:

- Is a layered – structure a partial microkernel idea???
- Not so:
- A microkernel is physically divided into separate modules. It may consist of 1 or more layers – but only logically.
- A layered kernel is physically divided into layers, but logically – it might consist of one or more modules.
- A microkernel may be (and often is) logically single layered because many layered kernel is again a ??

Some Issues

- How do you deal with hardware in UNIX?
  - Operating systems provide interfaces and management of hardware resources
  - E.g., interrupts and I/O devices.
- A microkernel seems to optimize operating system design
  - So, should make operating system (lower level) easier to modify
  - Layered approach– so, seems good in principle
- Is the UNIX (or other “user application” O/S) really a User Application?
  - Are users going to write additional operating systems?

Solution – Microkernel.

- Microkernel designs put a lot of OS services in separate processes to build modular operating systems.
  - Kernel's functionality is reduced and put into user servers.
- This architecture is actually a client-server model.
  - Clients call other OS services through microkernel.
- The central processes that provide the process management, file system etc are frequently called the servers.
- Microkernels are often also highly multithreaded.
  - Each thread has a different service to perform.
  - What happens with speed of IPC?

What is Mach?

- Mach
  - Transparent multiprocessing – Avoiding issues in BSD.
  - Protected message passing – Better than Unix message messaging.
  - “extensible” Microkernel
  - Multiple levels of operating system
    - Other O/S’s implemented as “applications”
  - Basis for NeXT O/S, Mac X O/S, OSF/1

Design Goals of Mach

- Full support for multiprocessing.
- Exploit other features of modern hardware architectures that were emerging at that time.
- Supports transparent distributed operation.
- Reduce the number of features in the kernel, and therefore make it less complex, giving the programmer a very small number of abstractions to work with.
- The abstractions are just general enough to allow several operating systems to be implemented on top of Mach.
- Full compatibility with UNIX BSD.
- Address the shortcomings of previous systems such as Accent.

Approach:

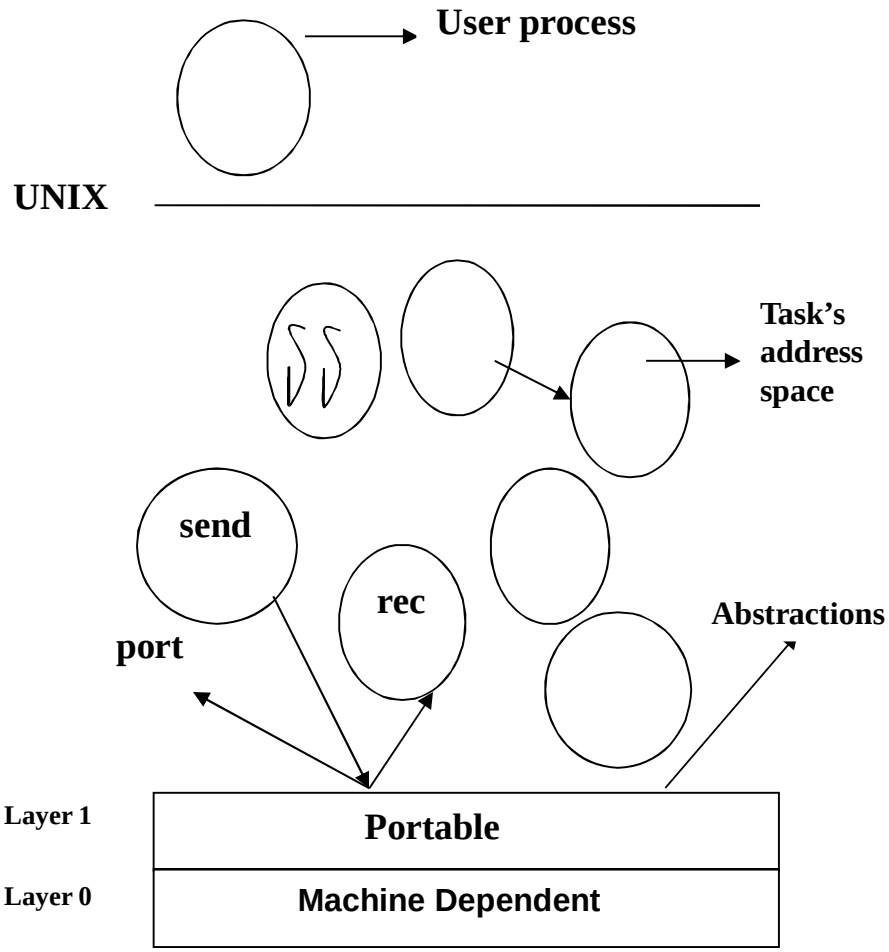
- a small, extensible system kernel which provides scheduling, virtual memory and interprocess communications
- and several, possibly parallel, operating system support environments which provide the following two items:

1) distributed file access and remote execution

2) emulation for established operating system environments such as UNIX.

Overall Mach

- IPC – RPC messages.
  - Send and receive.
- When the message queue is full the senders block; when it is empty, the receivers block.
- Indirect communication.
- Heavy weight context switching.
- Speed is compromised ; but protection is ensured.



Mach's abstractions

- A task is an execution environment and is the basic unit of resource allocation.
  - Includes a paged virtual address space (potentially sparse)
  - protected access to system resources (such as processors, port capabilities and virtual memory).
- A thread is the basic unit of execution. A thread executes in the context of a single task. A UNIX Process = Task + thread.
- A port is a simplex communication channel -- implemented as a message queue managed and protected by the kernel.
  - Basic object reference mechanism in MACH.
  - Ports are used to refer to objects; operations on objects are requested by sending messages to the ports which represent them.

Contd..

- A port set is a group of ports, implemented as a queue combining the message queues of the constituent ports.
  - A thread may use a port set to receive a message sent to any of several ports.
- A message is a typed collection of data objects used in communication between threads.
  - Can by of any size and may contain inline data, pointers to data, and capabilities for ports.
- A memory object is a secondary storage object that is mapped into a task's virtual memory.
  - memory object is treated like any other object.

Differences

- Ports are a protected entity that can only be addressed by the Mach microkernel,
- Port rights are attached to a given task and describe the operations that they can provide on a port,
- Port names are the identifiers that tasks must use to request some operations on this ports.
- This looks similar to - Files, files access rights and file descriptors in a traditional UNIX system.

Process Management

- Page fault- Performs better than Unix processes.
  - Each thread runs on a processor.
- Mach IPC is used for thread synchronization.
- Cthreads package.
- CPU scheduler - Global run queues with appropriate locks and local run queues. – Heavy weight context switching.
- Fixed Time quantum – What if the threads are lesser than processors?? Mach uses a variable time quantum inversely proportional to the no. of threads.
- Exception Handling – RPC message passing for handling.

IPC

- IPC -> ports and messages.
- Memory management is used in its implementation.
- Conversely, IPC is used in memory management.
- **Ports:**

  - Enable a thread to send data to another thread in a controlled manner.
  - Send & receive *rights – Port name + capability.*
  - Only one task with receive rights
  - Can be multiple with send rights
  - Sending receive rights to another task causes ownership of receive rights to change.

IPC Contd..

- Ports are location independent.
  - Ensures portability through this one communication mechanism.

Messages – Fixed-length header + variable number of typed data objects.

- Header - Destination port + reply port +length of the message.
- Data (inline data – versions vary).
- Port rights (only way to send port rights is in messages)
- Pointers to “out of line” data (Large messages).
  - Two cases: receiver on same vs. different.
- Used to implement remote procedure calls (RPC).

IPC Contd..

- Receiver on same computer
  - No need to necessarily copy message from sender to receiver
  - Takes time to copy messages.
  - Instead, when message contents unchanged, use virtual memory-based technique for improving efficiency
    - A kind of shared memory solution.
    - “copy-on-write”

IPC contd…

- Receiver on different computers.

  - In comparison with UNIX which uses low-level network protocols.

  - Mach provides an optimized solution.

  - Provided by NetMsgServer.

"User-Level" Message Server: NetMsgServer

- Enables location-transparent naming of ports
  - Does not matter which computer a port is on.
  - NetMsgServer dynamically resolves the addresses.
- Services:
  - Data Independence.
  - Network wide name service
  - Allows ports to be registered
  - Distributed database of port rights
  - Forwarding of messages by acting as proxy ports.
  - Data conversions (different computer architectures).

Memory Management

- A memory object represents a secondary object that is mapped into the address space of a task.
  - Treated just like any other object.
- Unlike traditional UNIX, which implied a contiguous virtual memory space Mach allowed for *sparse* address spaces, where regions of memory could be allocated from anywhere in the address space.
- No regular page table.
- User-level memory managers can be used instead for memory to be paged.

Contd..

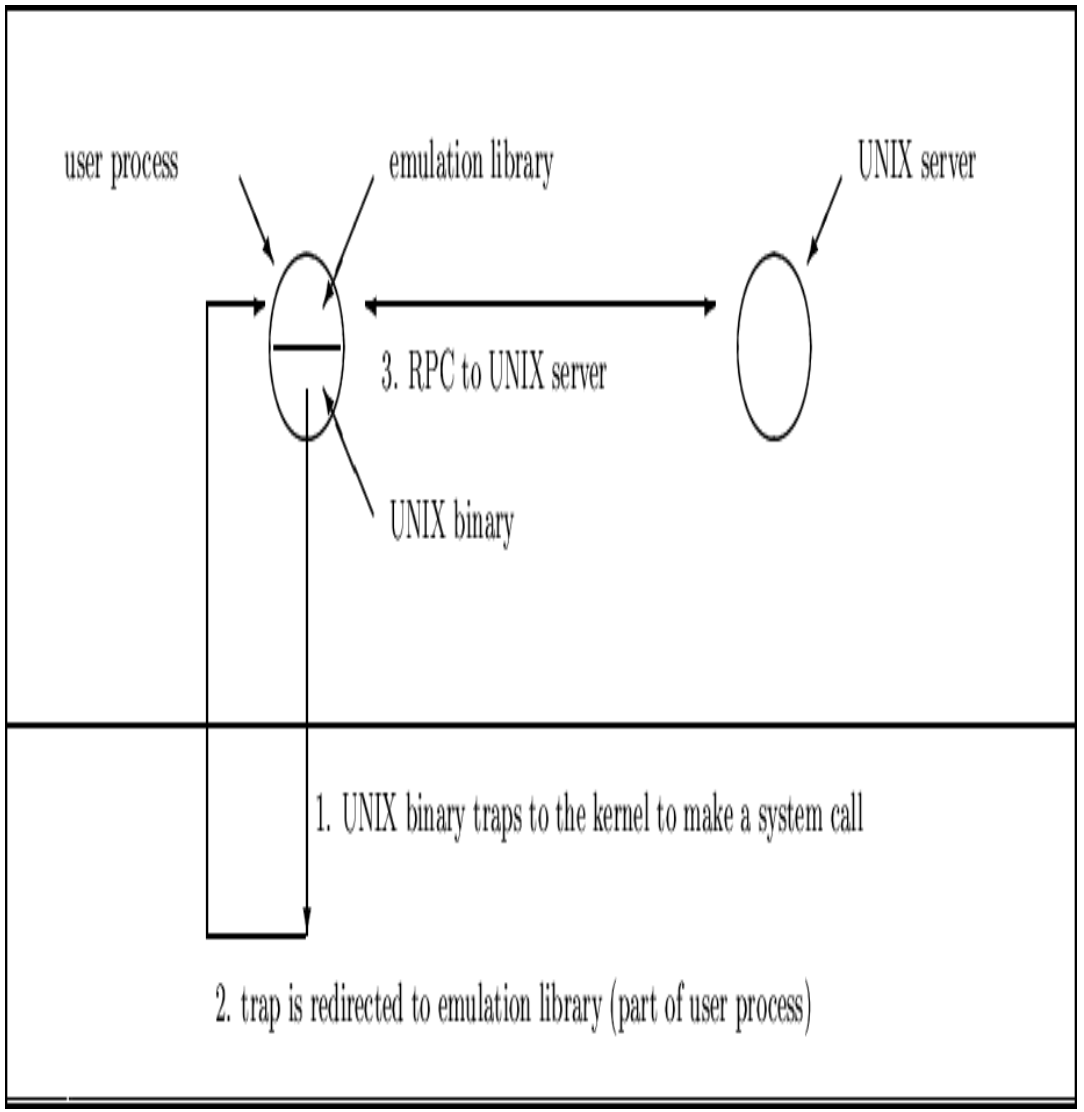
- Mach takes care of basics only
  - Acts as interface between hardware and user-level
    - e.g. receives page faults from hardware
    - Notifies relevant task (via port) of page fault
  - Implements pageout policy (FIFO).
  - Supplies default Memory Manager in some cases where user level fails.
- User-Level Memory Managers
  - Handle majority of memory management - can page memory.
  - System calls used to communicate with kernel for memory mapping / page-in / page-out / provide page-level locking
  - Responsible for consistency of the contents of a memory object mapped by tasks on different machines.

Shared Memory

- Mach approaches the shared memory in a different way.
- Consistent shared memory is supported only for shared processors.
- Tasks running on processors sharing memory
  - Standard FORK system call , Parent declares regions to be inherited by the child task.
  - Doesn't use copy-write strategy. But readable-writable technique.
  - shared page is readable: may be replicated on multiple machines.
  - shared page is writable: only one copy is changed.
- External memory manager – NetMemServer – Handles shared read-write from different machines willing to share memory.

System Calls

- Traps to the kernel.
- Upcalls into emulation library(USER LEVEL).
- Switch to any thread waiting on a port for operations like disk writes.
- Returns to emulation library.
- Returns from trap.
- System call is slow compared to traditional systems.



Summary

- Unix code – evicted from the kernel – can be replaced by another code at the user level.
- Successful in implementing multiprocessing and distributed computing.
- Extensibility at the expense of speed.
- Integrated memory management and IPC.
- IPC (message passing, system calls are very SLOW.

BSD

- **Berkeley Software Distribution** (**BSD**, sometimes called **Berkeley Unix**) is a Unix operating system derivative developed and distributed by the Computer Systems Research Group (CSRG) of the University of California, Berkeley, from 1977 to 1995. Today the term "BSD" is often used non-specifically to refer to any of the BSD descendants which together form a branch of the family of Unix-like operating systems. Operating systems derived from the original BSD code remain actively developed and widely used.
- Historically, BSD has been considered a branch of UNIX—"BSD UNIX", because it shared the initial codebase and design with the original AT&T UNIX operating system. In the 1980s, BSD was widely adopted by vendors of workstation-class systems in the form of proprietary UNIX variants such as DEC ULTRIX and Sun Microsystems SunOS. This can be attributed to the ease with which it could be licensed, and the familiarity it found among the founders of many technology companies of this era.

Application

- MACH is used in invoking kernel operations.
- Used in Task and Thread Management

Scope of Research

- Process Management in MACH
- Unix emulation in MACH
- Binary level operating system emulation.