The Mach System

by Abraham Silberschatz, Peter Baer Galvin, and Greg Gagne

Presented by Karl Matthias
Portland State University
CS 533 Concepts of Operating Systems

Much credit to Abdelhalim Ragab
Some photos from Life Magazine
Monolithic Kernel Structure

Monolithic Kernel based Operating System

System User mode

Operating system Kernel mode

Application

VFS, System call
IPC, File System
Scheduler, Virtual Memory
Device Drivers, Dispatcher, ...

Hardware

Source: http://upload.wikimedia.org/wikipedia/commons/d/d0/OS-structure2.svg
Microkernel Structure

Microkernel based Operating System

System

Operating system

User mode

Kernel mode

Application

Application IPC

UNIX Server

Device Driver

File Server

Basic IPC, Virtual Memory, Scheduling

Hardware

Source: http://upload.wikimedia.org/wikipedia/commons/d/d0/OS-structure2.svg
Mach History

- Derived from Carnegie-Mellon's Accent OS
  - Communications system
  - Small is good philosophy
- Mach = BSD Unix + Accent concepts + More
  - Originally constructed inside 4.2BSD kernel
  - Replaced one piece at a time
  - Not just research: commercially accepted
Design Principles

Object-oriented design: everything in Mach is an object.

Retain BSD Compatibility...

- Simple programmer interface
- Easy portability
- Extensive library of apps
- Combine utilities via pipes

But Add

- Distributed operation
- Varying network speed
- Simplified kernel
- Integrated memory management and IPC
- Heterogeneous systems
- Object-orientation
Mach’s overriding principle is to be a “simple, extensible kernel, concentrating on communications facilities.”

Few Primitive Abstractions

- Task
- Thread
- Port
- Port set
- Message
- Memory object
Memory Management and IPC

“Mach connects memory management and communication (IPC) by allowing each to be used in the implementation of the other.”

- Memory Management using IPC
  - Based on memory objects
  - Memory object represented by port or ports
  - IPC messages are sent to those ports to request operation on the object
  - Memory objects can be remote—the kernel caches the contents

- IPC using memory-management techniques
  - Messages are passed by moving pointers to shared memory objects
  - Virtual-memory remapping to transfer large contents and prevent copying whenever possible
    (virtual copy or copy-on-write)
Basic Process Management

- Synchronization Primitives
  - Using Mach IPC
    - Send/receive on a port (more on this later)
  - Thread level synchronization
    - Two states: running, suspended
    - Threads can be started and stopped at user level (co-operative)
Process Management:

C Threads Package

- User-level thread library built on top of Mach primitives
- Major influence on POSIX Threads standard
- Thread-control
  - Create/Destroy a thread
  - Wait for a specific thread to terminate then continue the calling thread
  - Yield
- Mutual exclusion using spinlocks only
- Condition Variables (wait, signal)
Process Management:

CPU Scheduler

- Only threads are scheduled, tasks are ignored
- Dynamic thread priority number (0 – 127)
  - The lowest priority thread is the one with the most recent large CPU usage
  - Priority sorts thread into one of 32 global run queues
- Per-processor queues used for locality specific threads (e.g. device drivers)
- There is NO central dispatcher
- Processors consult run queues to select next thread: the local queue first, then the global queue
- Optimization: thread time quantum varies inversely with total number of threads, but constant over the entire system
Process Management:

Exception Handling

- Implements a global cross-task exception handling system
- Works on distributed systems because it’s implemented via RPC messages
- Used to emulate BSD style signals
  - Supports execution of BSD programs
  - Signals are not suitable for multi-threaded environment
Interprocess Communication

Mach’s overriding principle is to be a “simple, extensible kernel, concentrating on communications facilities.”

- Universal communications mechanism between all objects in the system
- Location-independence, automatic addressing
- Isolation between objects because all messages pass through the kernel
- Secured communications via port rights
  - A capability to communicate on a particular port (many senders, 1 receiver)
  - A transferable right
  - Rights must be transferred via IPC so the kernel can track the transfer
Interprocess Communication:

Ports and Port Sets

- **Ports**
  - Implemented as a protected bounded queue in the kernel
  - Has a unique global name
  - **System Calls**
    - Allocate a new port in task, give the task all access rights
    - Deallocate task’s access rights to a port
    - Get port status
    - Create backup port to inherit the receive right when the existing port is deallocated

- **Port Sets**
  - A grouping of ports in the same task
  - Used for using a single thread as an incoming queue processor (e.g. Unix select or poll system calls)
  - A port may be a member of one set at a time
Interprocess Communication:

Messages

- Header + one or more typed data objects
- Header contains destination port name, reply port name, message length
- In-line message data contains simple types, port rights
- Out-of-line data: pointers
  - Via virtual-memory management
  - Uses copy-on-write
- Sparse virtual memory
Interprocess Communication:

Location Independence and Transparency

- **NetMsgServer**
  - User-level daemon that forwards messages between hosts
  - Used when receiver port is not on the kernel’s computer
  - Provides primitive network-wide name service
  - Network protocol independent interface allows many implementations

- **Mach 3.0 IPC for NORMA multiprocessor systems**
  - Directly in the kernel rather than in user space
  - Supports the formation of one single system across smaller systems
Interprocess Communication:

Synchronization Using IPC

- IPC-based Synchronization
  - Port used as synchronization variable since
    - Receive message = wait
    - Send message = signal
  - Only works natively for threads within a single task because one receiver task is allowed on a single port
  - Or via a daemon process that sends/receives messages between tasks
Memory Management

- **Memory Object**
  - Mach's basic abstraction of physical memory, an object just like everything else
  - Can represent mapped files, pipes, or other abstractions

- **User-level Memory Managers**
  - Memory can be paged by user-written memory managers
  - When needing to swap, kernel upcalls to support user-written memory manager
  - Respond to page faults from program code!

- **Mach default memory manager**
  - Fall back to kernel memory manager if no local manager
Memory Management:

Shared Memory

- Shared memory provides reduced complexity and enhanced performance
  - Used to implement fast IPC
  - Reduced overhead in file management
- Mach does not provide facilities to maintain memory consistency on different machines
  - User-level managers can implement when needed
Programmer Interface

- **System-call level**
  - BSD emulation in kernel in 2.5 (not really a microkernel), externalized in 3.0
  - Emulation libraries and servers in user space
  - Upcalls made to libraries in task address space, or server

- **C Threads package**
  - C language interface to Mach threads primitives

- **Interface/Stub generator (called MIG) for RPC calls**
Summary

- Few simple abstractions
- Focus on communication facilities
- System calls only for
  - IPC
  - Tasks, Threads, and Ports
  - Virtual memory
- Real world microkernels
  - Wide adoption in operating systems for servers, embedded devices, networking equipment
  - Mach pioneered many concepts
Resources

Original Mach research papers:

Apple Darwin Kernel Programming Guide:

OSF Mach kernel interfaces reference manual:

Linus vs. Tanenbaum famous debate (1992)—microkernels vs monolithic:
- [http://www.dina.kvl.dk/~abraham/Linus_vs_Tanenbaum.html](http://www.dina.kvl.dk/~abraham/Linus_vs_Tanenbaum.html)
- [http://groups.google.com/group/comp.os.minix/browse_thread/thread/c25870d7a41696d2](http://groups.google.com/group/comp.os.minix/browse_thread/thread/c25870d7a41696d2)

Again in 2006: