CS533 Concepts of Operating Systems

Jonathan Walpole
The Mach System

"Operating Systems Concepts, Sixth Edition"
by Abraham Silberschatz, Peter Baer Galvin, and Greg Gagne
Mach is a Microkernel OS
History

Developed at CMU
Based on the Accent Operating System
BSD Unix support
Mach influenced other OS designs
  - GNU Hurd
  - OSF
  - Mac OS X
  - etc
Goals

Support a diverse range of architectures
  - UMA, NUMA, NORMA
  - homogeneous and heterogeneous systems
Integrate memory management and IPC
Minimal kernel
Ease of use (Unix compatibility)
Ease of porting
Key Building Blocks

- Task
- Thread
- Port
- Port set
- Message
- Memory object

Diagram:
- Text region
- Threads
- Port
- Port set
- Memory object
- Secondary storage
Process Management

Tasks and threads

Synchronization primitives:

- Message exchange using Mach IPC:
  - Semaphore wait/signal operations can be implemented using IPC
- Suspend and resume calls defined for thread synchronization
User-Level Threads

C Thread library built on top of Mach primitives
Influenced POSIX P 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
Condition Variables (wait, signal)
CPU Scheduling

Only threads are scheduled
  – dynamic thread priority based on CPU usage

Global run queues + per processor local run queues

Thread time quantum varies inversely with total number of threads, but constant over the entire system
Interprocess Communication

Location independent IPC

Sender/Receiver must have *rights*
- port name + send or receive capability

Ports:
- Protected bounded queue in the kernel
- System Calls:
  - Allocate new port in task, give the task all access rights
  - Deallocate task’s access rights to a port
  - Get port status
  - Create backup port
- Port sets
Interprocess Communication

Messages:
  – Header + typed data objects

Header: destination port name, reply port name, message length

In-line data: simple types, port rights

Out-of-line data: pointers
  – Via virtual-memory remapping
  – Copy-on-write
Interprocess Communication

NetMsgServer:
- user-level capability-based networking daemon
- used when receiver port is not on the kernel’s computer
- Forward messages between hosts
- Provides primitive network-wide name service
Memory Management

Memory Object
- secondary storage, or data mapped into virtual memory
- served by user-level memory managers

User-level Memory Managers:
- Memory can be paged by user-written memory managers
- Mach has no knowledge of memory object contents
- New system calls to support external memory managers
- Mach default memory manager
Memory Management and IPC

Memory Management using IPC:
- Memory objects represented by port(s)
- IPC messages to those ports to request operation on the object
- Memory objects can be remote → kernel caches the contents

IPC using memory-management techniques:
- Pass message by moving pointers to shared memory objects
- Virtual-memory remapping to transfer large contents using virtual copy / copy-on-write techniques
Programmer Interface

System-call level
- Implemented via emulation libraries and servers
- Micro-kernel upcalls to libraries in task address space, or server

C Threads package
- C language interface to Mach threads primitives
- Not suitable for NORMA systems

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

Minimal kernel
A few simple abstractions
Communication is foundational
Higher level OS functionality built in user level servers