Mach

- History
- Design Principles
- System Components
- Process Management
- Interprocess Communication
- Memory management

History

- 1976: University of Rochester
  - OS for Data General computer
- 1986: Carnegie-Mellon University
  - OS for PERQ computer
- Retargeted for Multiprocessor machines
  - named: MACH
- 1990: Mach became basis for OSF/1
- Versions:
  - up to Mach 2.5
    - monolithic kernel, share code with BSD Unix
  - Mach 3.0: microkernel architecture
    - BSD code in user space
Design Principles

- Unix compatibility
- support diverse architectures
- simplified kernel structure
- distributed operation

System Components

- task
- thread
- port
- port set
- message
- memory object

small set of basic abstractions
Process Management (1/4)

- Task:
  - execution environment
  - provides resource allocation
  - has virtual address space
  - allows protected access to system resources via ports
  - contains: Thread(s)
- Thread:
  - basic unit of execution
  - runs within task
  - shares resources of task
- task + thread ~ traditional process

Process Management (2/4)

- thread control routines
  - create a new thread within a task, return id
  - destroy calling thread, return a value to creating thread
  - wait for thread to terminate
  - yield the processor
- traditional fork:
  - duplicate task
  - create thread which is started at point of fork call in parent
Process Management (3/4)

- Synchronization
  - Mutex operations:
    - lock
    - trylock
    - unlock
  - Condition variable:
    - wait
    - signal
    - broadcast

Process Management (4/4)

- CPU scheduling
  - CPU belongs to processor set
  - each thread has priority
    - based on its average CPU use
  - run queue for run-able threads:
    - global run queues for each processor set
    - local run queue for CPU specific thread
      - i.e. device driver
  - CPU executes from local run queue first, then global
  - thread runs for a fixed time amount
    - time amount is same for all CPUs
    - can be readjusted based on CPU/thread ratios
Interprocess Communication (1/3)

- **port**
  - reference mechanism
  - has port rights
    - receive, send, send-once
- **message**
  - data structure: typed collection of objects
  - contains port rights
- **communication is done by sending message to port**
  - messages are queued
- **port set**
  - group of ports (with receive right) that share a queue
  - thread can serve a port set

Interprocess Communication (2/3)

- **port functionality:**
  - allocate a new port with port rights
    - only one process has receive right
  - de-allocate a port
    - includes notification of port rights holders
  - get port status
  - create backup port
    - to receive messages if port is de-allocated
- **message content:**
  - destination port
  - reply port
  - message length
  - port rights
  - pointers to parameters
- **message is delivered via memory mapping**
Interprocess Communication (3/3)

- ports are location independent
  - “Network Message Server” handles non-local port communication
  - runs in user space
  - uses naming service to locate remote port
  - creates local proxy port and manages connection to network drivers
- performance considered inadequate
  - future versions have networking code in kernel

Memory Management (1/2)

- memory is organized into memory objects
- object may represent
  - raw data
  - file, whole or partial
  - pipe
- object is accessible via port
- object are managed
  - via user-space memory manager
  - or default kernel provided memory manager
- task’s memory is set of memory objects
Memory Management (2/2)

- shared memory
  - automatic among threads
  - between tasks user-space memory manager
- new task
  - can share objects of parent
  - can get copy of object
    - with copy-on-write semantics

Status

- Apple MacOS X
  - Darwin
    - Mach 3.0 + 4.4BSD Unix
- GNU HURD
  - uses Linux drivers
- OSF/1 AD
  - Paragon OS for massively-parallel computers