Advanced Operating Systems
Lecture notes

Dr. Dongho Kim
Dr. Tatyana Ryutov
University of Southern California
Information Sciences Institute

CSci555: Advanced Operating Systems
Lecture 11 – November 7, 2003
Kernels
(slides by Dr. Neuman and Dr. Obraczka)

Dr. Dongho Kim
Dr. Tatyana Ryutov
University of Southern California
Information Sciences Institute

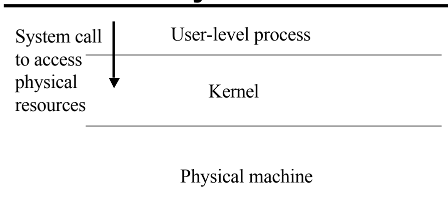
Kernels

- Executes in supervisory mode.
  - Privilege to access machine's physical resources.
- User-level process: executes in "user" mode.
  - Restricted access to resources.
  - Address space boundary restrictions.

Kernel Functions

- Memory management.
  - Address space allocation.
  - Memory protection.
- Process management.
  - Process creation, deletion.
  - Scheduling.
- Resource management.
  - Device drivers/handlers.

System Calls



System call: implemented by hardware interrupt (trap) which puts processor in supervisory mode and kernel address space; executes kernel-supplied handler routine (device driver) executing with interrupts disabled.

Kernel and Distributed Systems

- Inter-process communication: RPC, MP, DSM.
- File systems.
- Some parts may run as user-level and some as kernel processes.

Be or not to be in the kernel?

- Monolithic kernels versus microkernels.

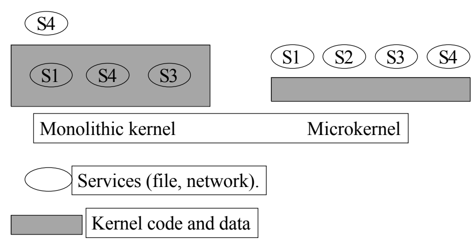
Monolithic kernels

- **Examples: Unix, Sprite.**
- **“Kernel does it all” approach.**
- **Based on argument that inside kernel, processes execute more efficiently and securely.**
- **Problems: massive, non-modular, hard to maintain and extend.**

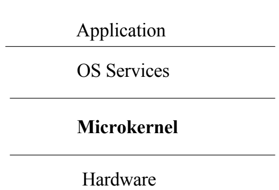
Microkernels

- **Take as much out of the kernel as possible.**
- **Minimalist approach.**
- **Modular and small.**
  - **10KBytes -> several hundred Kbytes.**
  - **Easier to port, maintain and extend.**
  - **No fixed definition of what should be in the kernel.**
  - **Typically process management, memory management, IPC.**

Micro- versus Monolithic Kernels



Microkernel



- Services dynamically loaded at appropriate servers.
- Some microkernels run service processes only @ user space; others allow them to be loaded into either kernel or user space.

The V Distributed System

- **Stanford (early 80’s) by Cheriton et al.**
- **Distributed OS designed to manage cluster of workstations connected by LAN.**
- **System structure:**
  - **_Relatively small kernel common to all machines.**
  - **_Service modules: e.g., file service.**
  - **_Run-time libraries: language support (Pascal I/O, C stdio)**
  - **_Commands and applications.**

V's Design Goals

- High performance communication.
  - Considered the most critical service.
    - _Efficient file transfer.
  - "Uniform" protocol approach for open system interconnection.
    - _Interconnect heterogeneous nodes.
  - "Protocols, not software, define the system".

The V Kernel

- Small kernel with basic protocols and services.
- Precursor to microkernel approach.
- Kernel as a "software backplane".
  - Provides "slots" into which higher-level OS services can be "plugged".

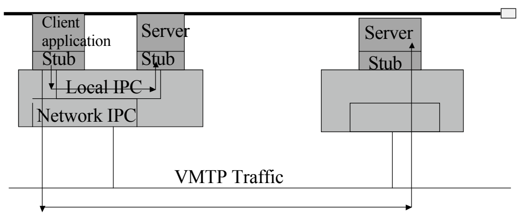
Distributed Kernel

- Separate copies of kernel executes on each node.
- They cooperate to provide "single system" abstraction.
- Services: address spaces, LWP, and IPC.

V's IPC Support

- Fast and efficient transport-level service.
  - Support for RPC and file transfer.
- V's IPC is RPC-like.
  - Send primitive: send + receive.
    - _ Client sends request and blocks waiting for reply.
    - _ Server: processes request serially or concurrently.
    - _ Server response is both ACK and flow control.
      - It authorizes new request.
      - Simplifies transport protocol.

V's IPC



Support for short, fixed size messages of 32 bytes with optional data segment of up to 16 Kbytes; simplifies buffering, transmission, and processing.

VMTP (1)

- Transport protocol implemented in V.
- Optimized for request-response interactions.
  - No connection setup/teardown.
  - Response ACKs request.
  - Server maintains state about clients.
    - _Duplicate suppression, caching of client information (e.g., authentication information).

VMTP (2)

- Support for group communication.
  - Multicast.
  - Process groups (e.g., group of file servers).
    - _Identified by group id.
    - _Operations: send to group, receive multiple responses to a request.

VMTP Optimizations

- Template of VMTP header + some fields initialized in process descriptor.
  - Less overhead when sending message.
- Short, fixed-size messages carried in the VMTP header: efficiency.

V Kernel: Other Functions

- Time, process, memory, and device management.
- Each implemented by separate kernel module (or server) replicated in each node.
  - Communicate via IPC.
  - Examples: kernel process server creates processes, kernel disk server reads disk blocks.

Time

- Kernel keeps current time of day (GMT).
- Processes can get(time), set(time), delay(time), wake up.
- Time synchronization among nodes: outside V kernel using IPC.

Process Management

- Create, destroy, schedule, migrate processes.
- Process management optimization.
  - Process initiation separated from address space allocation.
    - _Process initiation = allocating/initializing new process descriptor.
  - Simplifies process termination (fewer kernel-level resources to reclaim).
  - Simplifies process scheduling: simple priority based scheduler; 2nd. level outside kernel.

Memory Management 1

- Protect kernel and other processes from corruption and unauthorized access.
- Address space: ranges of addresses (regions).
  - Bound to an open file (UIO like file descriptor).
  - Page fault references a portion of a region that is not in memory.
  - Kernel performs binding, caching, and consistency services.

Memory Management 2

- Virtual memory management: demand paging.
  - Pages are brought in from disk as needed.
  - Update kernel page tables.
- Consistency:
  - Same block may be stored in multiple caches simultaneously.
  - Make sure they are kept consistent.

Device Management

- Supports access to devices: disk, network interface, mouse, keyboard, serial line.
- Uniform I/O interface (UIO).
  - Devices are UIO objects (like file descriptors).
  - Example: mouse appears as an open file containing x & y coordinates & button positions.
  - Kernel mouse driver performs polling and interrupt handling.
  - But events associated with mouse changes (moving cursor) performed outside kernel.

More on V…

- Paper talks about other V functions implemented using kernel services.
  - File server.
  - Printer, window, pipe.
- Paper also talks about classes of applications that V targets with examples.

The X-Kernel

- UofArizona, 1990.
- Like V, communication services are critical.
- Machines communicating through internet.
  - Heterogeneity!
  - The more protocols on user's machine, the more resources are accessible.
- The x-kernel philosophy: provide infrastructure to facilitate protocol implementation.

Virtual Protocols

- The x-kernel provide library of protocols.
  - Combined differently to access different resources.
  - Example:
    - _If communication between processes on the same machine, no need for any networking code.
    - _If on the same LAN, IP layer skipped.

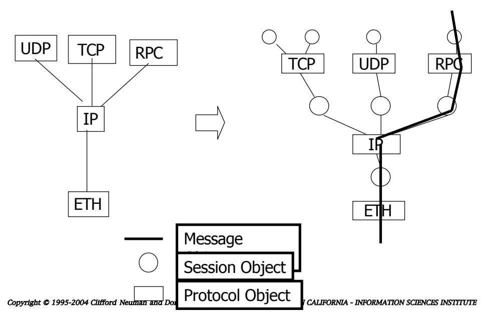
The X-Kernel : Process and Memory

- ability to pass control and data efficiently between the kernel and user programs
  - user data is accessible because kernel process executes in same address space
- kernel process -> user process
  - sets up user stack
  - pushes arguments
  - use user-stack
  - access only user data
- kernel -> user (245 usec), user -> kernel 20 usec on SUN 3/75

Communication Manager

- Object-oriented infrastructure for implementing and composing protocols.
- Common protocol interface.
- 2 abstract communication objects:
  - Protocols and sessions.
  - Example: TCP protocol object.
    - TCP open operation: creates a TCP session.
    - TCP protocol object: switches each incoming message to one of the TCP session objects.
    - Operations: demux, push, pop.

X-kernel Configuration



Message Manager

- Defines single abstract data type: message.
  - Manipulation of headers, data, and trailers that compose network transmission units.
  - Well-defined set of operations:
    - Add headers and trailers, strip headers and trailers, fragment/reassemble.
  - Efficient implementation using directed acyclic graphs of buffers to represent messages + stack data structure to avoid data copying.

Mach

- CMU (mid 80's).
- Mach is a microkernel, not a complete OS.
- Design goals:
  - As little as possible in the kernel.
  - Portability: most kernl code is machine independent.
  - Extensibility: new features can be implemented/tested alongside existing versions.
  - Security: minimal kernel specified and implemented in more secure way.

Mach Features

- OSs as Mach applications.
- Mach functionality:
  - Task and thread management.
  - IPC.
  - Memory management.
  - Device management.

Mach IPC

- Threads communicate using ports.
- Resources are identified with ports.
- To access resource, message is sent to corresponding port.
  - Ports not directly accessible to programmer.
  - Need handles to "port rights", or capabilities (right to send/receive message to/from ports).
- Servers: manage several resources, or ports.

Mach: ports

- *process port* is used to communicate with the kernel.
- *bootstrap port* is used for initialization when a process starts up.
- *exception port* is used to report exceptions caused by the process.
- *registered ports* used to provide a way for the process to communicate with standard system servers.

Protection

- **Protecting resources against illegal access:**
  - **Protecting port against illegal sends.**
- **Protection through *capabilities*.**
  - **Kernel controls port capability acquisition.**
  - **Different from Amoeba.**

Copyright © 1995-2004 Clifford Neuman and Dongho Kim - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE

Capabilities 1

- **Capability to a port has field specifying port access rights for the task that holds the capability.**
  - **Send rights: threads belonging to task possessing capability can send message to port.**
  - **Send-once rights: allows at most 1 message to be sent; after that, right is revoked by kernel.**
  - **Receive rights: allows task to receive message from port's queue.**
    - **At most 1 task, may have receive rights at any time.**
    - **More than 1 task may have sned/send-once rights.**

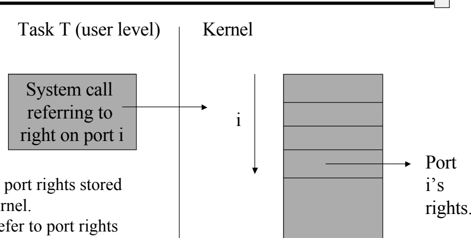
Copyright © 1995-2004 Clifford Neuman and Dongho Kim - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE

Capabilities 2

- **At task creation:**
  - **Task given bootstrap port right: send right to obtain services of other tasks.**
  - **Task threads acquire further port rights either by creating ports or receiving port rights.**

Copyright © 1995-2004 Clifford Neuman and Dongho Kim - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE

Port Name Space



. Mach's port rights stored inside kernel.
. Tasks refer to port rights using local id's valid in the task's local port name space.

. Problem: kernel gets involved whenever ports are referenced.

Copyright © 1995-2004 Clifford Neuman and Dongho Kim - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE

Communication Model

- **Message passing.**
- **Messages: fixed-size headers + variable-length list of data items.**

Header	T	Port rights	T	In-line data	T	Pointer to out-of line data

Header: destination port, reply port, type of operation.
T: type of information.
Port rights: send rights: receiver acquires send rights to port.
Receive rights: automatically revoked in sending task.

Copyright © 1995-2004 Clifford Neuman and Dongho Kim - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE

Ports

- Mach port has message queue.
  - Task with receive rights can set port's queue size dynamically: flow control.
  - If port's queue is full, sending thread is blocked; send-once sender never blocks.
- System calls:
  - Send message to kernel port.
  - Assigned at task creation time.

Task and Thread Management

- Task: execution environment (address space).
- Threads within task perform action.
- Task resources: address space, threads, port rights.
- PAPER:
  - How Mach microkernel can be used to implement other OSs.
  - Performace numbers comparing 4.3 BSD on top of Mach and Unix kernels.