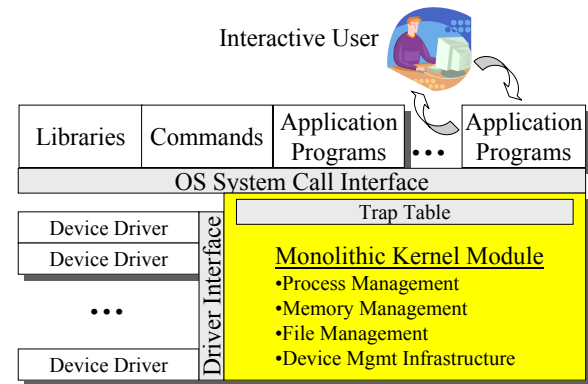


Micro-Kernel OS

CS 256/456

Dept. of Computer Science, University of Rochester

Monolithic System Structure



Most modern OSes fall into this category!

Microkernel System Structure

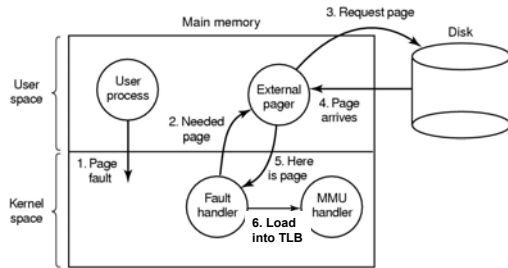
- Microkernel structure:
 - Moves bunch of functionalities from the kernel into "user" space.
 - Tend to have more frequent kernel/user crossings.
- What must be in the kernel and what can be in user space?
 - Protection **mechanisms** (protecting hardware; protecting user processes from each other).
 - Resource management **policies**.
 - Examples in memory management, file system, networking, ...
- Benefits:
 - Modular design?
 - More reliable/secure (less code is running in kernel mode).

Microkernel OS: Mach

- Mach
 - developed at CMU in late 80s
 - bits/pieces leading to NeXT, the foundation of MacOS 10
- Micro-kernel design
 - OS functionalities are pushed to user-level servers (e.g., user-level memory manager)
 - user-level servers are trusted (often run as root)
 - protection mechanisms stay in kernel while resource management policies go to the user-level servers

User-level Memory Management

- User-level memory management
 - trusted/protected by the kernel
 - kernel provides the basic protection mechanism
 - user-level memory manager handles page loading; decides replacement policy



4/12/2006

CSC 256/456 - Spring 2006

5

Inter-domain Communications

- Inter-domain communications in micro-kernel systems
 - between user-level applications and the kernel
 - between user-level applications and servers
 - between user-level servers and the kernel
 - down calls (normal system calls) and upcalls (calling from the kernel to user-level routines)
- Tend to have many inter-domain communications
 - resulting in large overhead

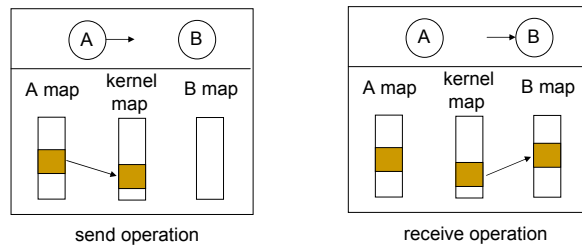
4/12/2006

CSC 256/456 - Spring 2006

6

Virtual Message Passing

- Virtual message passing
 - mess around with memory map tables (page tables) to speed up message passing



4/12/2006

CSC 256/456 - Spring 2006

7

Microkernel OS: Exokernel

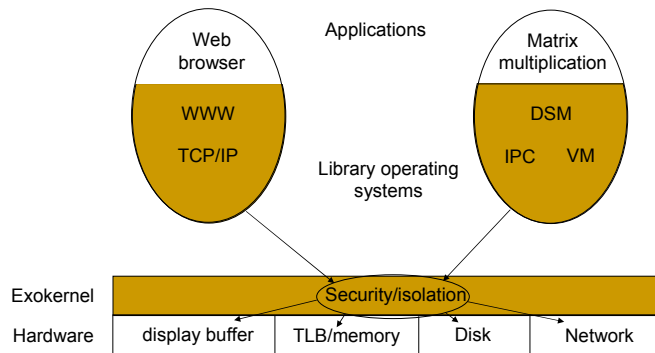
- Exokernel
 - developed at MIT in early 90s
- Micro-kernel design
 - OS functionalities are pushed to library OSes linked with individual user-level processes
 - communication takes place between user/kernel modules using message passing
- Benefits:
 - more reliable/secure (less code is running in kernel mode).
 - more flexibility (different user program can use different VM page replacement policies) ⇒ better performance.
- Problem it introduces?
 - security and isolation

4/12/2006

CSC 256/456 - Spring 2006

8

Architecture of Exokernel



4/12/2006

CSC 256/456 - Spring 2006

9

Library OS

- The kernel does not trust the library OS
- The library OS trusts the user program; so the library OS can be implemented without the concern of protection
- The library OS and the user program are linked together
 - low cost interaction between them
 - particularly helpful when applications and the OS interact frequently (application-assisted VM page replacement)
- Applications can link with customized library OS
 - flexibility
 - e.g., one process can use LRU page replacement and another can use MRU

4/12/2006

CSC 256/456 - Spring 2006

10

The Kernel

- The kernel does not trust the library OS or user processes
 - must decide allocation/binding of resources to different library OSes and user processes
 - must enforce allocation/binding of resources to library OSes and user processes

4/12/2006

CSC 256/456 - Spring 2006

11

Memory Management

- Two-level allocation
 - The kernel allocates memory among library OSes
 - Each library OS manages memory pages allocated to it
- The kernel enforces allocation among library OSes
- how does it work (for software-loaded TLBs)?
 - only the kernel can access the TLB, the kernel checks every TLB load to make sure a library OS (or a user process) doesn't access a page that it is not supposed to
 - who maintains the page table?
- how does it work (for hardware-loaded TLBs)?

4/12/2006

CSC 256/456 - Spring 2006

12

Summary

- Microkernel structure:
 - Moves functionalities from the kernel into "user" space.
- Benefits:
 - More reliable/secure (less code is running in kernel mode)
- Disadvantage on performance:
 - Tend to have more frequent domain crossings.
- Two types of micro-kernels:
 - Running user-level OS in a trusted server - Mach
 - Running user-level OS within untrusted user processes - Exokernel
 - more flexibility, but problems on security and isolation
- Why aren't they taking over the world?