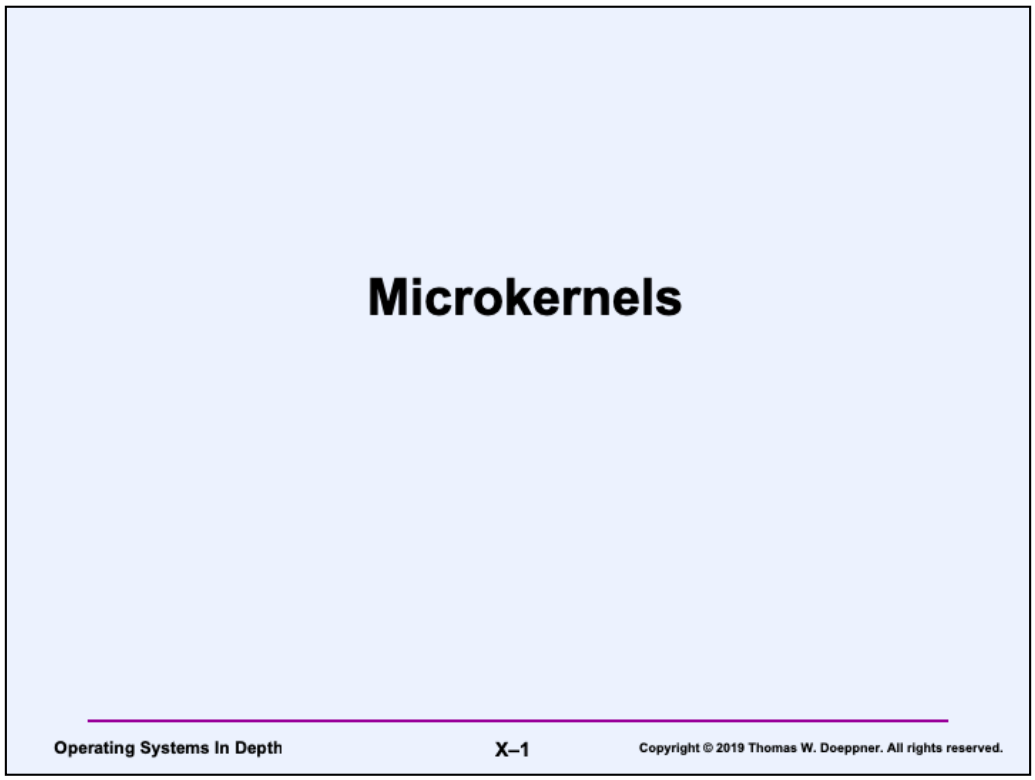
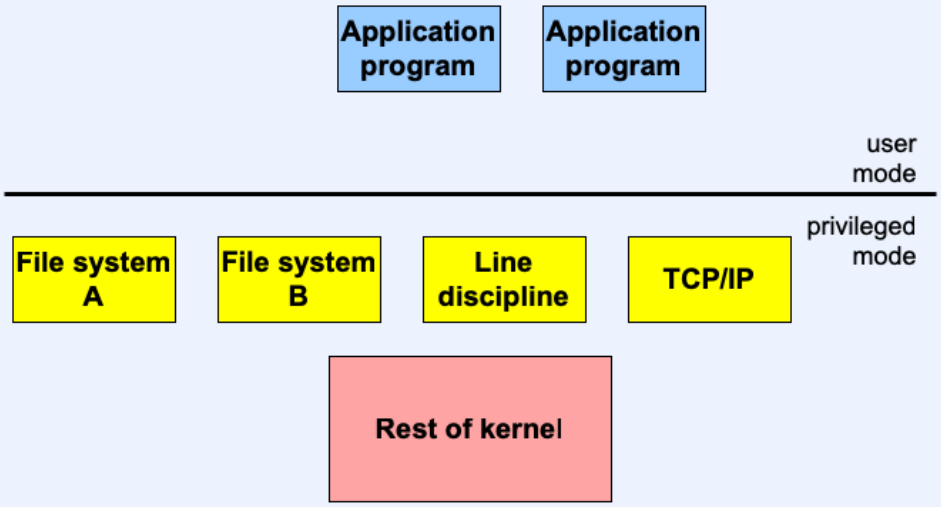


Details about Mach can be obtained from “Mach 3 Kernel Principles” by Keith Loepere, available at http://www.shakthimaan.com/downloads/hurd/kernel_principles.pdf.

Traditional OS Organization



Quiz

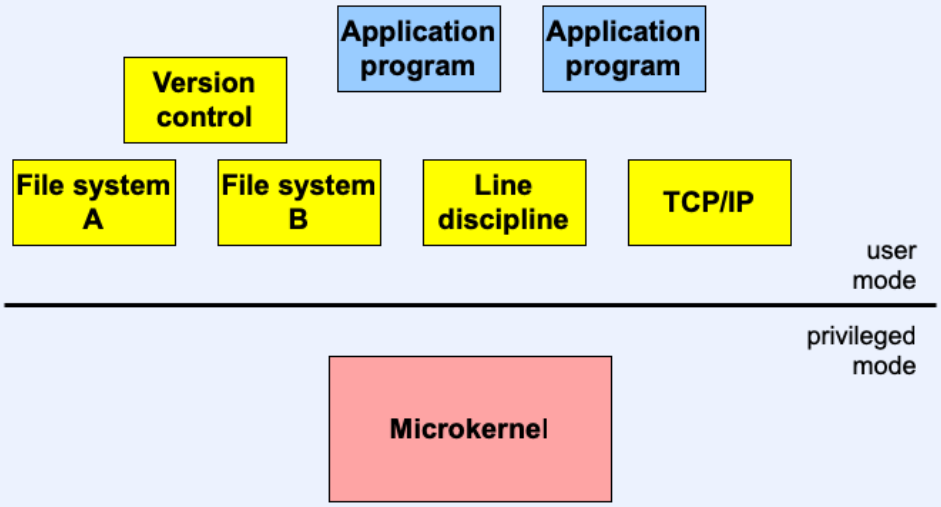
In the previous slide, assume each of the two application programs run as separate processes. What's in the slide employs:

a) three address spaces: one for each process, and one for the kernel

b) six address spaces: one for each process, one for each of the four kernel components, and one for the rest of the kernel

c) two address spaces: one for each process, with the kernel existing in a shared portion of the two process address spaces

OS Services as User Apps



Why?

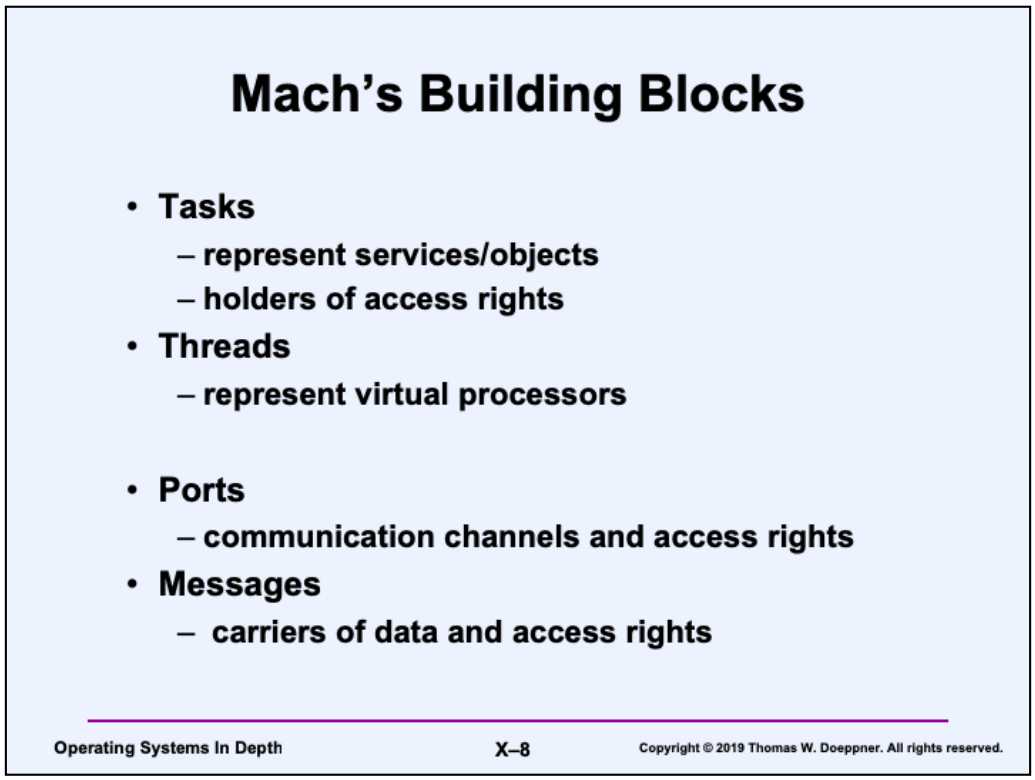
- It's cool …
- Assume that OS coders are incompetent, malicious, or both …
  - OS components run as protected user-level applications
- Extensibility
  - easier to add, modify, and extend user-level components than kernel components

Implementation Issues

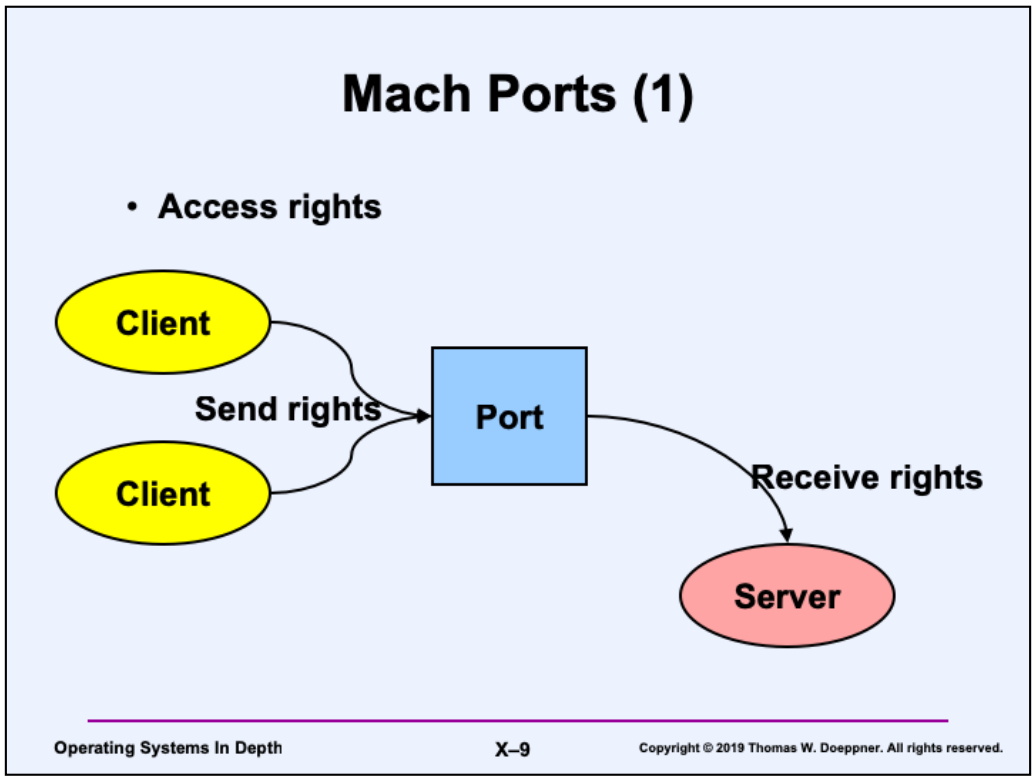
- What are the building blocks?
- What is run in privileged mode?

Mach

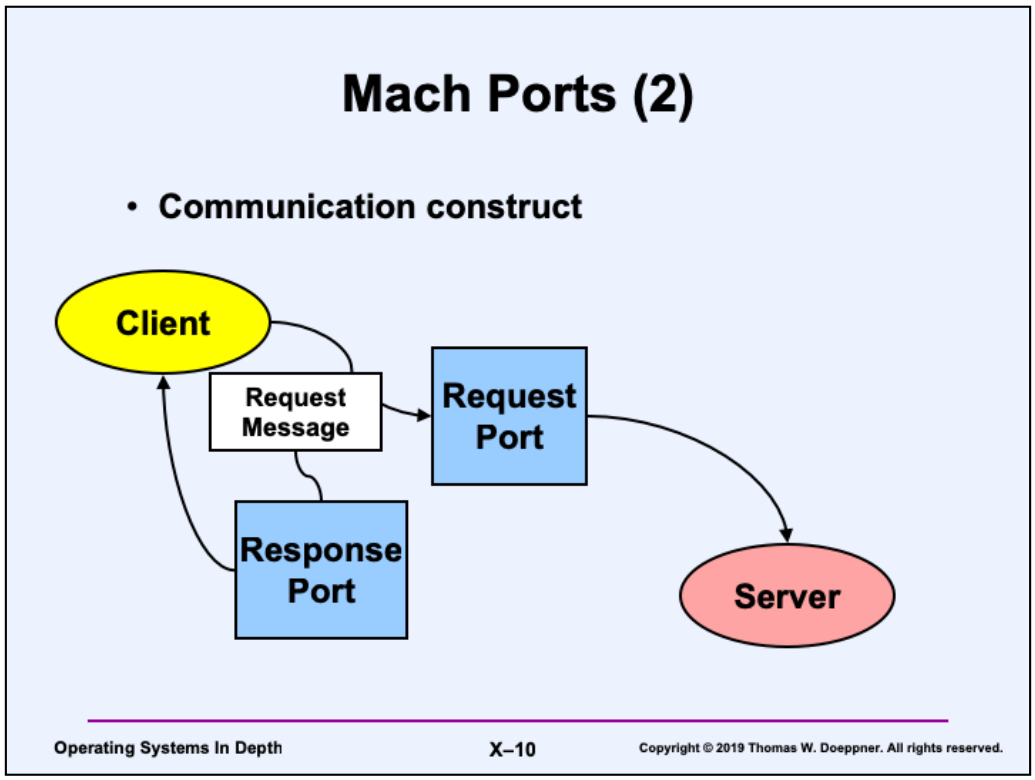
- Developed at CMU, then Utah
- Early versions shared kernel with Unix
  - basis of NeXT OS
    - basis of Apple OS X
- Later versions still shared kernel with Unix
  - basis of OSF/1
- Even later versions actually functioned as working microkernel
  - basis of GNU/HURD project
    - HURD: HIRD of Unix-replacing daemons
    - HIRD: HURD of interfaces representing depth



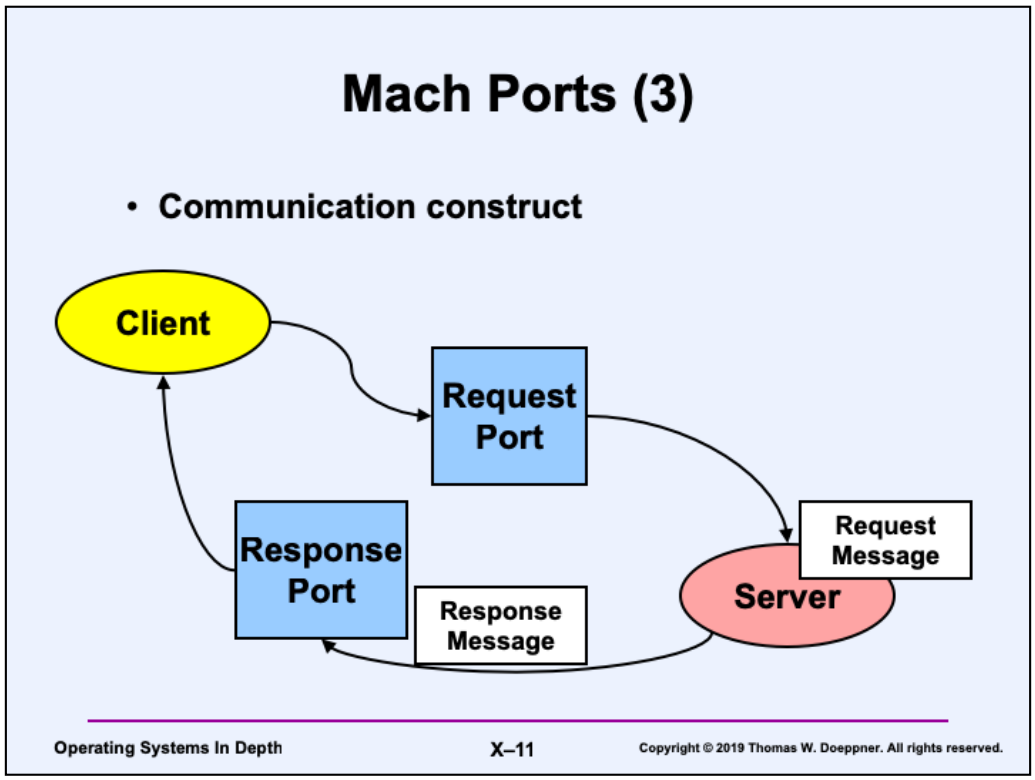
Tasks may be used to construct objects. A task itself is an object, as is a thread. There are additional objects implemented by the kernel, including device objects



For any particular port, exactly one task has receive rights, but any number may hold send rights. Thus a port represents a one-way communication channel from the holders of send rights to the holder of receive rights.



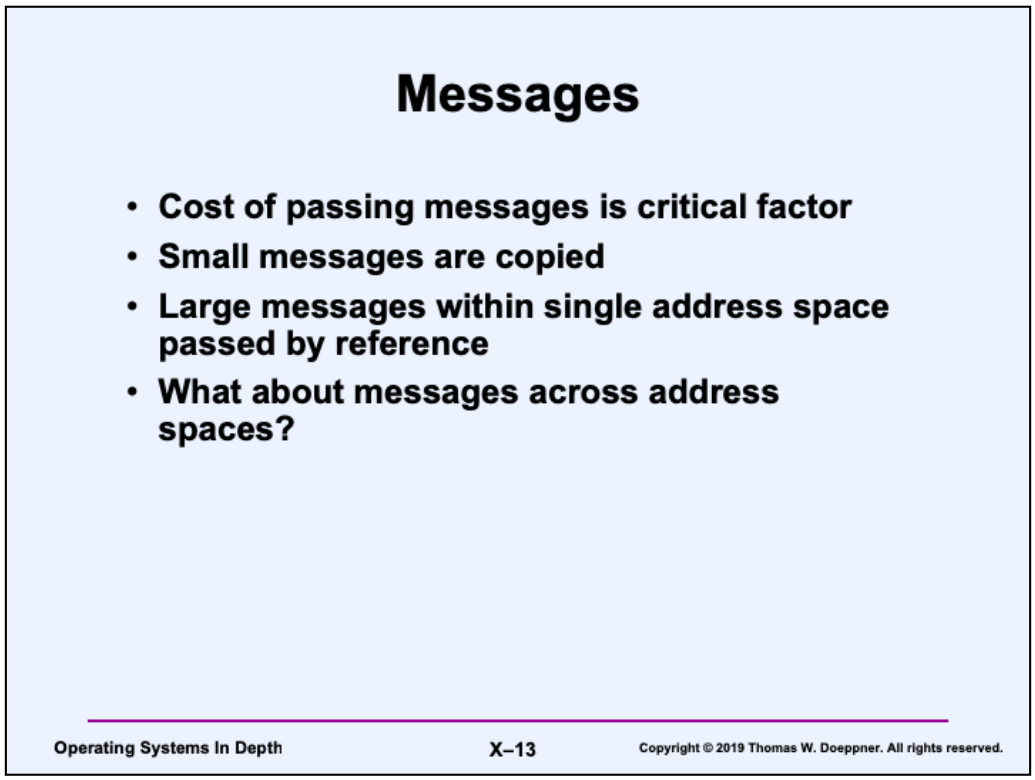
Messages, possibly containing send rights to some other port, may be sent by a holder of send rights to the port to the holder of receive rights.



One may provide a “send-once right” to a port, allowing the recipient to send just one message. This is useful for response ports.

Method Invocation

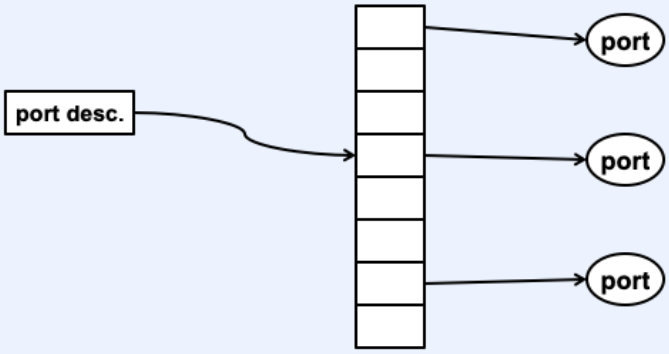
- ***Tasks*** **implement objects**
- **Access rights to *ports* are secure object references**
- ***Messages*** **are method invocations and responses**



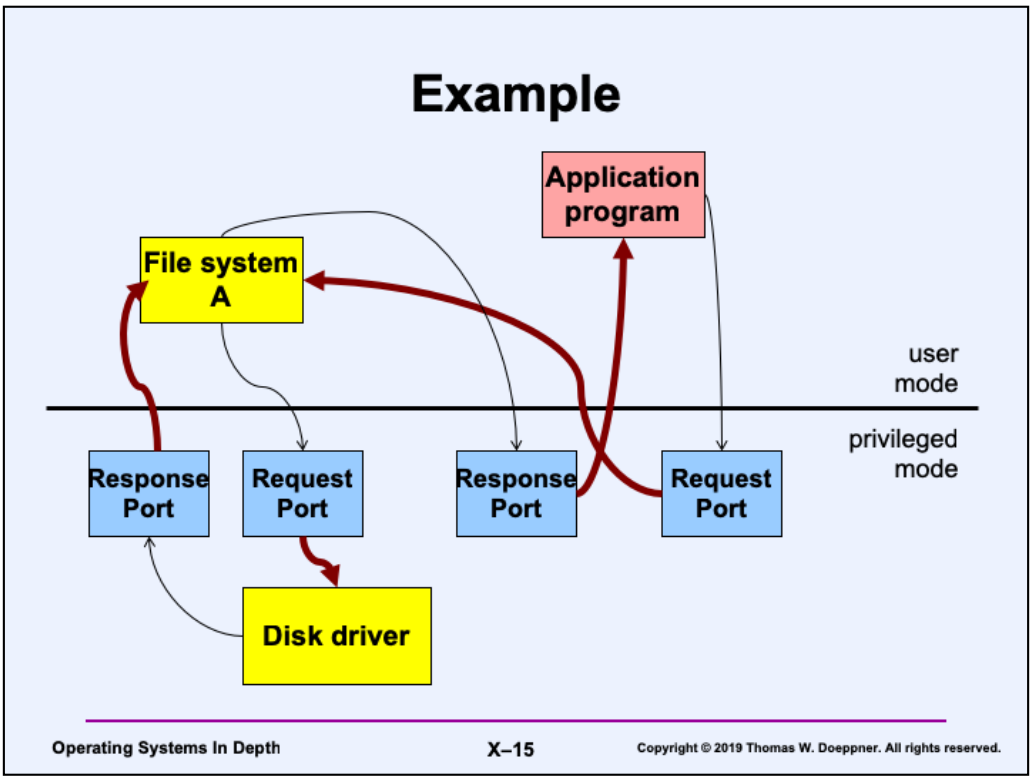
We'll take up the topic of optimizing message transfer across address-space boundaries when we discuss virtual memory.

Implementing Port Rights

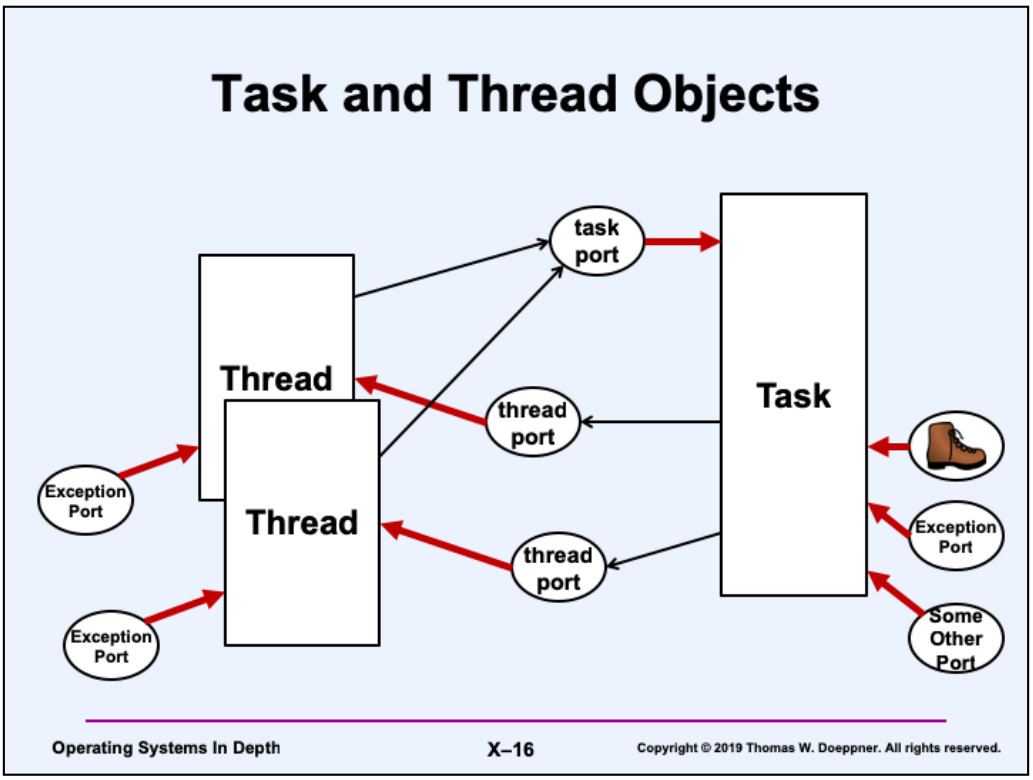
- References to ports must be secure and unforgeable
  - how are they done?



A port reference (or port descriptor) is implemented very much like a file descriptor.



The fat lines represent receive rights, the skinny lines represent send rights.



Note that all threads of a task have access to all the port rights owned by the task. Operations on threads via the thread port include suspending and resuming the thread, setting its register context, getting its register context, and terminating it. Creating a thread is an operation on the task port. Operations on task ports include creating a new task and terminating a task. A thread can set up a method to handle messages on its exception port, which are sent in response to exceptions by the kernel. If a thread hasn't set up such a method, or what it has set up doesn't handle a particular exception, the message is sent to the task's exception port.

A task's bootstrap port is used to obtain other port rights from some other object (the one that holds its receive rights).

Virtual Memory

- Memory cache objects
  - implemented in kernel
  - represent what's in real memory
- Memory objects
  - implemented in kernel or as user tasks
  - represent what's mapped into real memory

Devices

- Device master port exported by kernel
- Tasks holding send rights may request access to any device
  - send rights given for device port

Successful Microkernel Systems

-
-
- …

Attempts

- Windows NT 3.1
  - graphics subsystem ran as user-level process
  - moved to kernel in 4.0 for performance reasons
- Apple OS X
  - based on Mach
  - all services in kernel for performance reasons
- HURD
  - based on Mach
  - services implemented as user processes
  - no one uses it, for performance reasons …