6.033 Spring 2018

Lecture #6

- Monolithic kernels vs. Microkernels
- Virtual Machines
operating systems enforce modularity on a single machine using virtualization in order to enforce modularity + build an effective operating system

1. programs shouldn’t be able to refer to (and corrupt) each others’ memory

2. programs should be able to communicate

3. programs should be able to share a CPU without one program halting the progress of the others

**today:** running multiple OSes at once (and dealing with kernel bugs)
Virtual Machines

problem: how to (safely) share access to physical hardware?
Virtual Machines

VMM runs in kernel-mode on hardware

- virtual machine running guest OS
- virtual machine running guest OS
- virtual machine monitor (VMM)
- physical hardware
guest OS

virtual hardware
- U/K
- PTR
- page table

virtual machine monitor (VMM)

physical hardware
- U/K, PTR, page table, ...

VMM’s goal: virtualize hardware
A diagram illustrating the relationship between guest virtual address, guest physical address, host physical address, and the virtual machine monitor (VMM). The diagram shows the guest OS, virtual hardware, and physical hardware, with U/K, PTR, and page table as key components. The VMM manages the translation between virtual and physical addresses.
guest OS

virtual hardware

guest OS page table

load PTR

intercept!

VMM

guest OS page table + VMM page table = host page table

guest virtual -> guest physical

VMM page table

guest physical -> host physical

host page table

guest virtual -> host physical

physical hardware

 PTR -> host page table
In modern hardware, the physical hardware is aware of both page tables, and performs the translation from guest virtual to host physical itself.
guest OS

virtual hardware
  U/K
  PTR
  page table

virtual machine monitor (VMM)

physical hardware
  U/K, PTR, page table, ...

VMM’s goal: virtualize hardware
source: bugzilla.kernel.org, count of all bugs currently marked NEW, ASSIGNED, REOPENED, RESOLVED, VERIFIED, or CLOSED, by creation date

© source unknown. All rights reserved. This content is excluded from our Creative Commons license. For more information, see https://ocw.mit.edu/help/faq-fair-use.
monolithic kernels: no enforced modularity within the kernel itself

Basic interprocess communication, virtual memory, scheduling, file server, device drivers, network, …
microkernels: enforce modularity by putting subsystems in user programs
• **Virtual Machines** allow us to run multiple isolated OSes on a single physical machine, similar to how we used an OS to run multiple programs on a single CPU. VMs must handle the challenges of virtualizing the hardware (examples: virtualizing memory, the U/K bit).

• **Monolithic kernels** provide no enforced modularity within the kernel. **Microkernels** do, but redesigning monolithic kernels as microkernels is challenging.
Virtual Machines

(in the host OS model, there was actually a host OS)