Virtual Machines

CIS-4020
Vermont Technical College
Peter C. Chapin
Kernel Mode Code

• Hard to debug.
  – Can't use ordinary debugger
  – Can't single step the kernel and have a usable system.
• Kernel crashes take down entire system.
• Kernel errors can corrupt entire system.
  – Result can be an unbootable mess.
• Linux helps: Protects itself against errors in modules relatively well. Also bad modules don't have to be loaded upon system reboot.
Ideal Development Platform

• Develop kernel on a separate machine from where the code is tested.
  – But... transferring code to the test system awkward.

• We will develop code on the same machine.
  – But... we will use a virtual machine to make recovery and setup easier.

• Virtual machines also allow us to do Linux and QNX kernel work on a Windows host.

• We'll use a CPU simulator for Phoenix work.
Virtual Machines Popular

- An old idea who's time has come.
  - IBM's VM operating system gave a separate virtual machine to each user. Decades old!
- Good for kernel and driver development.
- Good for security (due to isolation).
- Good for training.
- Good for administration (doing test installs).
- Saves hardware costs.
VirtualBox

- We will use VirtualBox in lab. Similar to VMware or open source products like Virtual PC.
Network Configuration

- Host OS (Windows)
- Guest OS (Linux)
- Host Network Interface
- NAT (VirtualBox)
- Private Network

Host Hardware
Different VM Types

- Hypervisors (Xen)
- Hardware simulators (VirtualPC, VMware)
- CPU simulators (Bochs)
- User Mode Linux
- Virtual CPUs (Java, .NET)
Hypervisors

- Provides a generic hardware abstraction layer.
- OS must be written to use this layer.
Hardware Simulators

• Program instructions execute natively.
  – Fast
  – Forces guest CPU to be the same as host CPU.

• Access to hardware ports cause trap to virtual machine software.
  – VM software simulates hardware response.
  – I/O operations are thus slower than normal.

• In theory this works with any OS.
  – In practice simulation usually less than perfect.
CPU Simulators

• Entire machine simulated; even the CPU.
  – **Very** slow. Each instruction in the guest requires many instructions in the host system.

• Host CPU can be different than guest CPU.
  – Run x86 software on Alpha, etc!

• Allows detailed instrumentation.
  – Guest can log CPU error conditions.
  – Guest can count instruction usage, etc.

• Bochs (http://bochs.sourceforge.net/)
User Mode Linux

- A hardware architecture that uses normal Linux as the “hardware.”
  - Part of the regular kernel distribution. Architecturally specific code in arch/um.
- Only supports Linux on Linux.
- The guest is a completely ordinary process.
  - No administrative access needed to configure and run the guest.
  - Can be debugged with gdb and similar tools!
- http://user-mode-linux.sourceforge.net/
Virtual CPUs

- No attempt to simulate an existing machine.
  - Virtual CPU can be simplified and specialized.
- Generally provides a CPU together with many OS services (file system access, security, etc).
- Code often compiled to native code “Just In Time” (JIT).
- Java VM and .NET CLR are two important examples.
In This Class...

- We will use a VirtualBox VM for Linux and QNX
  - Running on Windows
  - Full installation with many tools and applications

- We will use Bochs for Phoenix
  - Running in the HackBox VM
  - A kind of “nested virtualization!”