Chapter 2: Operating-System Structures
Administrivia

- Read Chapter 3.
- Kernel assignment to complete by start of class Friday:
  - Clean kernel source:
    ```bash
cd linux-2.6.27.1; make mrproper
    ```
  - Import kernel source into your own repository:
    ```bash
    svn import https://merlin.goucher.edu/svn/kelliher/linux-2.6.27.1
    ```
    (Replace my username with YOURS)
  - Remove unversioned kernel source:
    ```bash
    cd ..; rm -rf linux-2.6.27.1
    ```
Administrivia, contd.

- Checkout versioned kernel source:
  
  `svn co https://merlin.goucher.edu/svn/kelliher/linux-2.6.27.1`

- Confirm that all is ok:
  
  `cd linux-2.6.27.1; svn status`
  
  should execute quietly.

- Read Programming Project 2.8 starting on pg. 93.

- Read [http://www.linuxchix.org/content/courses/kernel_hacking/lesson5](http://www.linuxchix.org/content/courses/kernel_hacking/lesson5). What kernel file should actually be modified?
Outline

- Virtual machine examples.
- Adding a syscall to Linux.
Virtual Machines

- A virtual machine takes the layered approach to its logical conclusion. It treats hardware and the operating system kernel as though they were all hardware.
- A virtual machine provides an interface identical to the underlying bare hardware.
- The operating system host creates the illusion that a process has its own processor and (virtual memory).
- Each guest provided with a (virtual) copy of underlying computer.
Virtual Machines History and Benefits

- First appeared commercially in IBM mainframes in 1972
- Fundamentally, multiple execution environments (different operating systems) can share the same hardware
- Protect from each other
- Some sharing of file can be permitted, controlled
- Commutate with each other, other physical systems via networking
- Useful for development, testing
- **Consolidation** of many low-resource use systems onto fewer busier systems
- “Open Virtual Machine Format”, standard format of virtual machines, allows a VM to run within many different virtual machine (host) platforms
Virtual Machines (Cont)

(a) Nonvirtual machine (b) virtual machine
Para-virtualization

- Presents guest with system similar but not identical to hardware
- Guest must be modified to run on paravirtualized hardware
- Guest can be an OS, or in the case of Solaris 10 applications running in containers
Solaris 10 with Two Containers

<table>
<thead>
<tr>
<th>global zone</th>
<th>user programs</th>
<th>user programs</th>
<th>user programs</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>system programs</td>
<td>system programs</td>
<td>system programs</td>
</tr>
<tr>
<td></td>
<td>CPU resources</td>
<td>network addresses</td>
<td>network addresses</td>
</tr>
<tr>
<td></td>
<td>memory resources</td>
<td>device access</td>
<td>device access</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CPU resources</td>
<td>CPU resources</td>
</tr>
<tr>
<td></td>
<td></td>
<td>memory resources</td>
<td>memory resources</td>
</tr>
</tbody>
</table>

| virtual platform | device management |

| zone management |

Solaris kernel

network addresses

device

... device
VMware Architecture

- Application
  - Guest operating system (free BSD)
  - Virtual CPU
  - Virtual memory
  - Virtual devices

- Virtualization layer

- Host operating system (Linux)

- Hardware
  - CPU
  - Memory
  - I/O devices
The Java Virtual Machine

Java program
.class files

class loader

Java interpreter

host system
(Windows, Linux, etc.)

Java API
.class files
Operating-System Debugging

- **Debugging** is finding and fixing errors, or bugs
- OSes generate *log files* containing error information
- Failure of an application can generate *core dump* file capturing memory of the process
- Operating system failure can generate *crash dump* file containing kernel memory
- `printk()` :
Adding a Syscall to Linux

Three main elements:

- Assigning a number to the syscall, making it accessible from user space
- Adding it to the kernel syscall table, allowing the kernel to call the syscall
- Writing the syscall and adding it to the kernel.
Adding a Syscall to Linux

- Syscall path:
Syscall Parameter Passing

- Parameters passed in registers:
  ```c
  _syscall1( long, diffjiffies, long, ujiffies );
  printf( "difference is %lx\n", diffjiffies(jifs) );
  ```

- Parameters passed by reference:
  ```c
  int access_ok( type, address, size );

  unsigned long copy_from_user( void *to, const void __user *from, unsigned long n );

  unsigned long copy_to_user( void *to, const void __user *from, unsigned long n );
  ```
Additional Background:

End of Chapter 2