### A View of Operating System Services

<table>
<thead>
<tr>
<th>user and other system programs</th>
</tr>
</thead>
<tbody>
<tr>
<td>GUI</td>
</tr>
<tr>
<td>batch</td>
</tr>
<tr>
<td>command line</td>
</tr>
<tr>
<td>user interfaces</td>
</tr>
</tbody>
</table>

### system calls

<table>
<thead>
<tr>
<th>services</th>
</tr>
</thead>
<tbody>
<tr>
<td>program execution</td>
</tr>
<tr>
<td>I/O operations</td>
</tr>
<tr>
<td>file systems</td>
</tr>
<tr>
<td>communication</td>
</tr>
<tr>
<td>resource allocation</td>
</tr>
<tr>
<td>accounting</td>
</tr>
<tr>
<td>error detection</td>
</tr>
<tr>
<td>protection and security</td>
</tr>
</tbody>
</table>

### operating system

### hardware
Operating System Design and Implementation

- Affected by choice of hardware, type of system
- *User* goals and *System* goals
  - *User goals* – operating system should be convenient to use, easy to learn, reliable, safe, secure, and fast
  - *System goals* – operating system should be easy to design, implement, and maintain, as well as flexible, reliable, error-free, secure, and efficient
- Important principle is separation
  - **Policy**: What will be done?
  - **Mechanism**: How to do it?
  - The separation of policy from mechanism is a very important principle, it allows maximum *flexibility* if policy decisions are to be changed later
Operating Systems Structures

- Structure/Organization/Layout of OSs:
  1. Monolithic (one unstructured program)
  2. Layered
  3. Microkernel
  4. Virtual Machines

- The role of Virtualization
Monolithic Operating System
Monolithic OS – Basic Structure

- Application programs that invoke the requested system services.
- A set of system services that carry out the operating system procedures/calls.
- A set of utility procedures that help the system services.
MS-DOS System Structure

- MS-DOS – written to provide functionality in the least space:
  - not divided into modules (monolithic).
  - Although MS-DOS has some structure, its interfaces and levels of functionality are not well separated.
MS-DOS Layer Structure

- Application Program
- Resident System Program
- MS-DOS Device Drivers
- ROM BIOS Device Drivers
UNIX System Structure

- UNIX – limited by hardware functionality, the original UNIX operating system had limited structuring. The UNIX OS consists of two separable parts
  - The kernel
    - Consists of everything below the system-call interface and above the physical hardware
    - Provides the file system, CPU scheduling, memory management, and other operating-system functions; a large number of functions for one level
  - Systems programs
## Traditional UNIX System Structure

<table>
<thead>
<tr>
<th>(the users)</th>
</tr>
</thead>
<tbody>
<tr>
<td>shells and commands</td>
</tr>
<tr>
<td>compilers and interpreters</td>
</tr>
<tr>
<td>system libraries</td>
</tr>
</tbody>
</table>

### System-call interface to the kernel

<table>
<thead>
<tr>
<th>signals terminal handling</th>
</tr>
</thead>
<tbody>
<tr>
<td>character I/O system</td>
</tr>
<tr>
<td>terminal drivers</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>file system</th>
</tr>
</thead>
<tbody>
<tr>
<td>swapping block I/O system</td>
</tr>
<tr>
<td>disk and tape drivers</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CPU scheduling</th>
</tr>
</thead>
<tbody>
<tr>
<td>page replacement</td>
</tr>
<tr>
<td>demand paging</td>
</tr>
<tr>
<td>virtual memory</td>
</tr>
</tbody>
</table>

### Kernel interface to the hardware

<table>
<thead>
<tr>
<th>terminal controllers</th>
</tr>
</thead>
<tbody>
<tr>
<td>terminals</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>device controllers</th>
</tr>
</thead>
<tbody>
<tr>
<td>disks and tapes</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>memory controllers</th>
</tr>
</thead>
<tbody>
<tr>
<td>physical memory</td>
</tr>
</tbody>
</table>
Traditional UNIX Kernel [Bach86]
Layered Approach

- The operating system is divided into a number of layers (levels), each built on top of lower layers.
- The bottom layer (layer 0) is the hardware; the highest (layer N) is the user interface.
- With modularity, layers are selected such that each uses functions (operations) and services of only lower-level layers.
Layered Operating System
Operating System Layers
Older Windows System Layers
Microkernel System Structure

- Move as much functionality as possible from the kernel into "user" space.
- Only a few essential functions in the kernel:
  - primitive memory management (address space)
  - I/O and interrupt management
  - Inter-Process Communication (IPC)
  - basic scheduling
- Other OS services are provided by processes running in user mode (vertical servers):
  - device drivers, file system, virtual memory...
Layered vs. Microkernel Architecture

(a) Layered kernel

(b) Microkernel
Microkernel System Structure

Communication takes place between user modules using message passing.

Benefits:
- Easier to extend a microkernel
- Easier to port the operating system to new architectures
- More reliable (less code is running in kernel mode)
- More secure

Detriments:
- Performance overhead of user space to kernel space communication
Microkernel Operating System
Benefits of a Microkernel Organization

- Extensibility/Reliability
  - modular design
  - easier to extend a microkernel
  - more reliable (less code is running in kernel mode)
  - more secure (less code to be validated in kernel)
  - small microkernel can be rigorously tested.

- Portability
  - changes needed to port the system to a new processor is done in the microkernel, not in the other services.
Mach 3 Microkernel Structure

Figure A.1  Mach 3 structure.
Mac OS X Structure

application environments and common services

kernel environment

BSD

Mach
Windows NT Client-Server Structure
Windows NT 4.0 Architecture
Structure of the MINIX 3 System

Microkernel handles interrupts, processes, scheduling, IPC

Clock  Sys

Drivers

Servers

User progs.

Process

User mode

Shell  Make  ...  Other

FS  Proc.  Reinc.  ...  Other

Disk  TTY  Netw  Print  ...  Other
Kernel Modules

- Most modern operating systems implement kernel modules
  - Uses object-oriented approach
  - Each core component is separate
  - Each talks to the others over known interfaces
  - Each is loadable as needed within the kernel

- Overall, similar to layers but more flexible
Solaris Modular Approach
Virtual Machines

- A virtual machine takes the layered approach to its logical next step. 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 (Cont.)

(a) Non-virtual machine

(b) virtual machine
Testing a new Operating System

- Test operating system
  - Test kernel
  - Virtual devices

- Other processes
  - $P_2$
  - $P_3$
  - $P_4$

- Virtualizing kernel $V$

- Physical devices
Integrating two Operating Systems

The diagram illustrates the integration of a batch operating system and an interactive operating system through a virtualizing kernel. The batch operating system is represented by $P_1$ and the interactive operating system by $P_2$. The virtualizing kernel is denoted as $V$.
The Role of Virtualization

(a) General organization between a program, interface, and system.

(b) General organization of virtualizing system A on top of system B.
Java Virtual Machine

- Compiled Java programs are platform-neutral bytecodes executed by a Java Virtual Machine (JVM).

- JVM consists of:
  - class loader
  - class verifier
  - runtime interpreter

- Just-In-Time (JIT) compilers increase performance.
The Java Virtual Machine

- Java program .class files
- class loader
- Java API .class files
- Java interpreter
- host system (Windows, Linux, etc.)
Hypervisor/VMM
Types of Hypervisors

(a) A type 1 hypervisor. (b) A type 2 hypervisor
Para- vs. Full-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
- Full-virtualization: unmodified guest OSes