Operating System Components and Services

Tom Kelliher, CS 311

Feb. 6, 2012

Announcements:

From last time:

1. System architecture issues.
2. I/O programming.
3. Memory hierarchy.
4. Hardware protection.

Outline:

1. Operating system components.
2. System calls.
3. Operating system structure.
4. Virtual machines.

Next week: kernel hacking lab.

1 Operating System Components

From the virtual machine point of view (also resource management)

These components **reflect** the services made available by the O.S.

1. User interface

   - Command-line interface — Linux shell, Windows cmd.
   - Graphical user interface — "desktop" metaphor, "tiles" metaphor (MS Metro), "direct manipulation" (iOS, Android).

     Characterized by pointer — mouse, multi-touch, etc; icons to represent objects.
   - Batch system — historic. NIH Biowulf.

2. Process Management

   - Process is a program in execution — numerous processes to choose from in a *multiprogrammed* system,
   - Process creation/deletion (bookkeeping)
   - Process suspension/resumption (scheduling, system vs. user)
   - Process synchronization
   - Process communication
   - Deadlock handling

3. Memory Management

   (a) Maintain bookkeeping information

   (b) Map processes to memory locations

   (c) Allocate/deallocate memory space as requested/required

4. I/O Device Management

   (a) Disk management functions such as free space management, storage allocation, fragmentation removal, head scheduling

   (b) Consistent, convenient software to I/O device interface through buffering/caching, custom drivers for each device.

5. File System

   Built on top of disk management

   (a) File creation/deletion.

   (b) Support for hierarchical file systems

   (c) Update/retrieval operations: read, write, append, seek

   (d) Mapping of files to secondary storage

6. Protection

   Controlling access to the system

   (a) Resources — CPU cycles, memory, files, devices

   (b) Users — authentication, communication

   (c) Mechanisms, not policies

7. Network Management

   Often built on top of file system

   (a) TCP/IP, IPX, IPng
   (b) Connection/Routing strategies
   (c) "Circuit" management — circuit, message, packet switching
   (d) Communication mechanism
   (e) Data/Process migration

8. Network Services (Distributed Computing)

   Built on top of networking

   (a) Email, messaging (Exchange)
   (b) FTP
   (c) gopher, www
   (d) Distributed file systems — NFS, AFS, LAN Manager
   (e) Name service — DNS, YP, NIS
   (f) Replication — gossip, ISIS
   (g) Security — kerberos

9. User Interface

   (a) Character-Oriented shell — sh, csh, command.com (*User replaceable*)
   (b) GUI — X, Win32

2 System Calls

Some of the Linux system calls:

(Refer to *man 2*.)

1. Process Management

   Scheduling, deadlock detection is transparent.

   (a) fork, vfork, exit, exec

   (b) wait

   (c) signals, pipes, streams, sockets

2. Memory management

   For the most part, transparent to the user.

   (a) malloc, free

3. I/O Device Management

   Devices are treated as files, so I/O devices are supported by the file system.

4. File System

   (a) creat, open, close

   (b) lseek, read, write

   (c) stat, chmod, chown

   (d) link, unlink

   (e) mkdir, rmdir

(f) sync

5. Communication

   Two models:

   (a) Message passing

      i. Processes communicate by passing messages — mailbox model.

      ii. Primitives: send, receive.

      iii. Each Process has a private address space.

      iv. Perfect for inter-processor communication. No synchronization problems. Latency can be a problem. Queueing.

   (b) Shared memory

      i. Processes communicate by sharing memory — bulletin board model.

      ii. Primitives are implicit once spaces are mapped.

      iii. Good for large amounts of intra-processor communication. Synchronization problems.

   Issues: naming, security, transparency, replication, etc.

   Message passing

   (a) socket, bind, accept, read.

   (b) socket, connect, write.

   Shared memory:

   (a) mmap.

3 Operating System Structures

Definitions:

- Layered System — A system in which pieces are built on top of other pieces, with hardware as a foundation. A layer make calls exclusively to the layer beneath it.

  Advantages:

  1. Well defined structure.
  2. Modular.
  3. Information Hiding.

  Disadvantages:

  1. Poor performance on “deep” calls — latency.
  2. Difficulty in creating a “good” layered design.

- Micro Kernel — A small executive that provides only necessary functionality to support threads/processes:

  - CPU scheduling.
  - Process primitives: create, destroy, suspend, activate, change priority, etc.
  - IPC.
  - Virtual Memory.
  - Interrupt handlers.
  - Device driver interface.

Runs in kernel (supervisor) mode. Everything else runs in user mode.

Advantages:

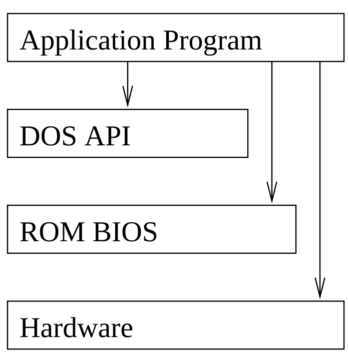
1. Only a small amount of code runs in kernel mode.
2. Easily extended.
3. System routines work with an abstract machine model — portability.

Disadvantages:

1. System calls require a context switch (slow).
2. Not easily extended (traditional kernel).

3.1 MS-DOS

Weak layering:



3.2 Unix

Traditional kernel:

Users
Programs
Kernel -- Implementing All System Services
Hardware

3.3 MS Windows NT 4.0

Micro kernel:

Users
Subsystems (Filesystem, Security, etc.), Applications
Executive Services (Thread Support, Phys. I/O)
Microkernel
Hardware

4 Virtual Machines

Lowest software level (between kernel and hardware) provides a *virtual machine* interface to multiple, independent kernels. VMware. Java VM.

1. Support for OS development alongside a production system.
2. Virtual user/kernel mode, real user/kernel mode.
3. How is a disk operation carried out?
4. Virtual Machine is/isn't a simulation/emulation:
    - Java VM: simulate one architecture with another. VMware, Xen, MS Virtual Server, not so much.
    - Efficiency?

5. How are physical devices (a disk) “virtualized?”
6. Java portability: “Write once run anywhere.”
7. Java security: the “sandbox.” Access to local resources. Security or impediment to application development?
8. Java consistency: same look, feel between platforms?