Chapter 2
Software Components

Objectives

• Describe historical milestones for three popular operating systems
• Describe the architecture of common PC operating systems
• Describe the functions of an operating system
• Describe the interaction between an operating system and its components
Objectives (continued)

- Describe utilities that are available for Windows 7 and Fedora 13
- Describe applications that are available for Windows 7 and Fedora 13

Common Operating Systems

- Disk operating system (DOS)
  - Term most often associated with MS-DOS
  - Single-tasking operating system
  - Accessible through a command-line interface (CLI)
    - User interaction based on text commands
  - Reason to learn:
    - Command line can be used to copy files, change file attributes, or execute specific utilities.
    - Enhances skill set of a game development Systems Analyst or Platform Designer
Windows 7 Operating System

- Microsoft Windows:
  - Development spurred by appearance of Macintosh
- Xerox PARC (Palo Alto Research Center):
  - Influenced the design of Macintosh and Windows
  - Example: created modern PC graphical user interface
- Windows development period: 1985 to present
  - DOS-based versions culminated in Windows Me
  - Windows NT thread started with Windows NT 3.1 and continues today with Windows 7
Fedora Operating Systems

- Linux:
  - Family of UNIX-like computer operating systems that use the Linux kernel
  - Developed by Linus Torvalds
  - Uses the X window system for graphical displays
- Fedora 13:
  - Owes its parentage to UNIX
  - Code-named Goddard
  - General-purpose OS released under the GNU General Public License
The first "Unix" system was developed by Ken Thompson in the Bell AT&T laboratories at Murray Hill in New Jersey in the United States from 1969. Ken Thompson’s aim was to develop a simple interactive operating system, called "Multics" (Multiplexed Information and Computing System) in order to run a game which he had created (space travel, a simulation of the solar system). A consortium made up of MIT (Massachusetts Institute of Technology), General Electric Co. and Bell Lab was then formed around Multics. Ken Thompson and Dennis Ritchie recreated the system in order to create a limited version of Multics called UNICS (UNiplexed Information and Computing Service), quickly shortened to Unix, that would run on a smaller machine (a DEC PDP-7, Programmed Data Processor which only had 4K of memory to make user programs run). The date of 1st January 1970 is considered as the birth date of the UNIX system.
Common Operating System Architectures

- Basic operating system services:
  - Reading and writing files
  - Allocating and managing memory
  - Making access control decisions
- DOS architecture:
  - DOS is a real-mode operating system
  - DOS user interface is a command-line interface (CLI)
  - Windows 7 provides a CLI for interacting with the OS (command prompt).

Figure 2-1 Simplified DOS architecture
Courtesy Course Technology/Cengage Learning
Windows 7 Architecture

• Basic types of user mode processes:
  – System support processes: (not service processes)
    Examples: logons (Winlogon) and session manager
  – Service processes: (started by service control manager) Examples: host Win32 services such as Task Scheduler
  – User applications: DOS, Win32, and Win64 applications
  – Environment subsystem: exposes native OS system services to user applications - emulates other OSs
  • Windows now only supports the Win32 environment

• Dynamic link library (DLL):
  – Library of executable functions or data. Is an interface between user applications and the OS

Guide to Parallel Operating Systems with Windows 7 and Linux

Figure 2-2 Simplified Windows 7 architecture
Courtesy Course Technology/Cengage Learning
Guide to Parallel Operating Systems with Windows 7 and Linux

Windows 7 Architecture (continued)

- Kernel mode components of Windows 7:
  - Executive: provides basic OS services
  - Kernel: provides low-level OS functions
  - Device drivers: translate user I/O requests into specific hardware I/O requests
  - Hardware Abstraction Layer (HAL): insulates kernels/drivers from intricacies of the hardware
  - Windowing and graphics: implement the windowing interface or graphical user interface (GUI)
Windows Architecture

- Windows uses a modified microkernel, or macrokernel OS.
- No commercial OS is based on a pure microkernel design because it is too slow. (context switching)
- Windows isolates functions in user mode when it makes sense to do so.
- However, for performance reasons, functions are moved to kernel mode. A good example is the windowing and graphics servers that have a high degree of interaction with the graphics and print drivers. The move reduces the number of context switches required to render screen images or prints. (Versions of Windows prior to NT 4.0 ran windowing and graphics servers in user mode, resulting in very poor graphics performance.)

Fedora 13 Architecture

- Subsystems of the Fedora 13 kernel:
  - Process scheduler (SCHED): controls process access to the processor
  - Memory manager (MM): Permits multiple processes to securely share the main memory system
  - Virtual file system (VFS): abstracts the details of the various hardware devices
  - Network interface (NET): provides access to several networking standards
  - Interprocess communication (IPC): supports several mechanisms for process-to-process communication
Figure 2-3 Simplified Fedora 13 architecture
Courtesy Course Technology/Cengage Learning

Guide to Parallel Operating Systems with Windows 7 and Linux
Functions of an Operating System

- Managing system resources
  - Processor management
  - Memory management
  - Device management
  - Storage management
  - User interface

Figure 2-4 Functions of an operating system
Courtesy Course Technology/Cengage Learning
Processor Management

- Single-tasking OS runs one program at a time
- Multitasking OS runs multiple programs at once:
  - Individual programs share one or more processors
  - Example: run both spreadsheet and word processor
  - Windows and Linux are multitasking
- Processes are composed of one or more threads:
  - Thread: sequence of instructions
- Multithreading:
  - Executing threads in parallel
  - Threads for one process share one address space
    - This can be good and bad; it facilitates easy communications
      between threads, but a poorly written program may cause one
      thread to overwrite data used by another thread.
- Simultaneous execution is actually rapid switching
Processor Management (continued)

- Two methods for sharing CPU time:
  - Cooperative multitasking: tasks yield control by choice
  - Preemptive multitasking: task preemption allowed
    - Tasks allocated to processor for fixed time (time slice)
- Multiprocessing: runs multiple concurrent threads
- Symmetric multiprocessing (SMP):
  - Identical processors share computational load
  - Examples: Windows and Linux
- Multiuser operating systems:
  - Windows 7 has a Fast User Switching feature
  - Fedora 13 supports more than one user at a time (Ctrl-Alt-F1 through Ctrl-Alt-F6)

Memory Management

- Protected mode: an improvement over real mode
  - Addresses memory > 1024 KB, 32-64 bit data path
  - Allows for virtual memory and multitasking
  - Windows and Linux operate in protected mode
- Virtual memory gives appearance of more memory
  - Virtual address: logical address in a program
  - Physical address: identifies actual storage location
- Paging process
  - Programs divided into pages with virtual addresses
  - OS maps page from disk to primary memory
  - Unneeded pages are returned to disk and replaced
Memory Management

- Allocation of memory based on program need
- Virtual addresses:
  - Memory locations mapped to physical memory
- Physical addresses:
  - Numbers that identify an actual storage location
- Paging process:
  - OS moves pages of physical memory to and from a paging file onto a disk
  - Page maps of the affected processes are updated

Abstract Model of Virtual to Physical Address Mapping

VPFN – Virtual Page Frame Number
PFN – Page Frame Number

Guide to Parallel Operating Systems with Windows 7 and Linux
Virtual Memory

- This and the following 7 slides cover details of how virtual memory functions and are provided to give some insight into the benefits of virtual memory.
- As the processor executes a program it reads an instruction from memory and decodes it. In decoding the instruction, the processor may need to fetch or store the contents of a location in memory. The processor then executes the instruction and moves on to the next instruction in the program. In this way the processor is always accessing memory either to fetch instructions or to fetch and store data.
- In a virtual memory system all of these addresses are virtual addresses and not physical addresses. These virtual addresses are converted into physical addresses by the processor based on information held in a set of tables maintained by the operating system.

To make this translation easier, virtual and physical memory are divided into handy sized chunks called pages. These pages are all the same size. They need not be but if they were not, the system would be very hard to administer. Linux on Intel x86 systems it uses 4 Kbyte pages (1024, 4-byte values=4096 bytes). Each of these pages is given a unique number: the page frame number (PFN).
- In this paged model, a virtual address is composed of two parts: an offset and a virtual page frame number. If the page size is 4 Kbytes, bits 1-10 of the virtual address contain the offset \(2^{10}=1024\) and bits 12 and above are the virtual page frame number. The processor extracts the virtual page frame number and offset from a virtual address every time it encounters one. Then it matches the virtual page frame number to a physical page and uses the offset to specify how far to go into the page.
- The processor uses page tables to match the virtual page frame number to the physical page.
Details of how page tables are used to obtain a physical address

- Each entry in the page table contains the following information:
  - Valid flag. This indicates if this page table entry (PTE) is valid
  - The physical page frame number that this entry describes
  - Access control information. This describes how the page may be used. Can it be written to? Does it contain executable code?
- The page table is accessed using the virtual page frame number as an offset. Virtual page frame 5 would be the 6th element of the table (0 is the first element).
- To translate a virtual address within process Y into a physical one, the processor must first work out the virtual address page frame number and the offset within that virtual page. By making the page size a power of 2 this can be easily done by masking and shifting. Looking again at the figures and assuming a page size of 0x2000 bytes (which is decimal 8192) and an address of 0x2194 in process Y's virtual address space then the processor would translate that address into offset 0x194 into virtual page frame number 1 (0x2194 – 0x2000 = 0x194).

The processor uses the virtual page frame number as an index into the process' page table to retrieve its page table entry. If the page table entry at that offset is valid, the processor takes the physical page frame number from this entry.

- If the entry is invalid, the process has accessed a non-existent area of its virtual memory. In this case, the processor cannot resolve the address and must pass control to the operating system so that it can fix things up.
- Just how the processor notifies the operating system that the correct process has attempted to access a virtual address for which there is no valid translation is specific to the processor. However the processor delivers it, this is known as a page fault and the operating system is notified of the faulting virtual address and the reason for the page fault.
- For a valid page table entry, the processor takes that physical page frame number and multiplies it by the page size to get the address of the base of the page in physical memory. Finally, the processor adds in the offset to the instruction or data that it needs.
• Using the above example again, process Y's virtual page frame number 1 is mapped to physical page frame number 4 which starts at 0x8000 (4 x 0x2000). Adding in the 0x194 byte offset gives us a final physical address of 0x8194.

• By mapping virtual to physical addresses this way, the virtual memory can be mapped into the system's physical pages in any order.

• In the figure above, process X's virtual page frame number 0 is mapped to physical page frame number 1, whereas virtual page frame number 7 is mapped to physical page frame number 0 although it is higher in virtual memory than virtual page frame number 0.

• This demonstrates an interesting byproduct of virtual memory; the pages of virtual memory do not have to be present in physical memory in any particular order.

---

Shared Virtual Memory

• Virtual memory makes it easy for several processes to share memory. All memory access are made via page tables and each process has its own separate page table. For two processes sharing a physical page of memory, its physical page frame number must appear in a page table entry in both of their page tables.

• The figure above shows two processes that each share physical page frame number 4. For process X this is virtual page frame number 3 whereas for process Y this is virtual page frame number 1.

• This illustrates an interesting point about sharing pages: the shared physical page does not have to exist at the same place in virtual memory for any or all of the processes sharing it.
Physical and Virtual Addressing Modes

- It does not make much sense for the operating system itself to run in virtual memory.
- This would be a nightmare situation where the operating system must maintain page tables for itself. Most multi-purpose processors support the notion of a physical address mode as well as a virtual address mode.
- Physical addressing mode requires no page tables and the processor does not attempt to perform any address translations in this mode.
- The Linux kernel is linked to run in physical address space.
**Virtual Memory Access Control**

- The page table entries also contain access control information. As the processor is already using the page table entry to map a process’ virtual address to a physical one, it can easily use the access control information to check that the process is not accessing memory in a way that it should not.
- There are many reasons why you would want to restrict access to areas of memory. Some memory, such as that containing executable code, is naturally read only memory; the operating system should not allow a process to write data over its executable code. By contrast, pages containing data can be written to, but attempts to execute that memory as instructions should fail. Most processors have at least two modes of execution: *kernel* and *user*. This adds a level of security to your operating system. Because it is the core of the operating system, and therefore, can do most anything, kernel code is only run when the CPU is in kernel mode. You would not want kernel code executed by a user or kernel data structures to be accessible except when the processor is running in kernel mode.

**Memory Management (continued)**

- Differences between Windows 7 and Fedora 13:
  - You can make more detailed decisions in Fedora 13 because the OS software is open source
  - Fedora 13 allows you to:
    - Look at source code
    - Know which algorithms are being used
    - Use source code to expand your knowledge of memory management, etc.
- Modern operating systems:
  - Available in 32-bit and 64-bit versions
Device Management

- Device drivers connect the OS to hardware - They translate data streams between the operating system and the hardware subsystems.
- Drivers take data streams that the operating system has defined as a file and translate them into a series of pixels on a display or into streams of bits placed in specific locations on a storage device.
- One reason that drivers are separate from the operating system is so that new functions can be added to the driver – and thus to the hardware subsystem – without requiring the operating system itself to be modified, recompiled, and redistributed.
- The next slide shows how Lotus under DOS, for example, provided printer software and drivers that controlled printer fonts, new page creation, etc. Lotus had to provide a driver for each printer on the market; these drivers were hard to write, and Lotus, like most developers, was struggling to keep up with new printers as they came to market.
- Any application development including games were faced with the same problems. It was an untenable approach.
Device Management

- In Windows 7, the OS provides a device driver for the printer. With this approach applications no longer have to deal with printer drivers themselves, but only have to provide an interface to the Windows component called the graphical device interface (GDI).

- Windows implementation of device drivers
  - Application provides graphical device interface (GDI)
  - Driver connects GDI to a device, such as a printer
  - Request flow: application->GDI->driver->device
  - See the next slide

- Linux implementation of device drivers is similar to Windows
  - Application produces PostScript (page description language developed and marketed by Adobe Systems)
  - The postscript output goes to Ghostscript, a suite of free software based on an interpreter of the Adobe Postscript and Portable Document Format (PDF) page description language. Ghostscript converts PostScript to device commands.
  - Flow is then passes to the device driver.
  - The OS provides the printer driver for every printer that it supports.
Device Management

- The basis of a printing system in Linux and Windows is the spooler.
- Spooler: manages queues of print jobs
  - Spool (acronym for simultaneous peripheral operations online)
- A queue is usually associated with a single printer, and jobs submitted by users are processed on a first-come, first-served basis.
- For Windows 7, the application generates output that is passed to a module, such as the GDI. The module then calls the printer driver to render the output. The output is then passed to the printer monitor, which controls the transfer to the printer.
- The printing is done in the background while the user interacts with other applications in the foreground.

Figure 2-8 Printing system
Courtesy Course Technology/Cengage Learning
Storage Management

- Setting up a partition in DOS:
  - `fdisk` command creates a partition
  - `format` command prepares partition for use
- Setting up partition in Windows 7:
  - Use Disk Management utility
- Setting up partition in Fedora 13:
  - `fdisk` command creates a partition
  - `mke2fs` command creates ext2 and ext3 file systems
- Three other storage-related tasks performed by OS:
  - File maintenance, security, quota management

User Interface

- Command-line interface (CLI): text-based interaction
- Graphical user interface (GUI):
  - Provides image- and text-based interface
  - Example: Windows 7 and Linux
- GUI interfaces provided by Fedora 13:
  - GNOME: Windows-like desktop system
  - KDE: user friendly desktop
- Fedora 13:
  - Configures six different types of consoles for use
    - Switch consoles by pressing Ctrl-Alt and F keys 1-6
    - Allows you to work on several tasks at once, or have several different user IDs logged in.
OS Interaction

• Major components of a PC:
  – Hardware
  – Applications

• OS interaction with these major components. How does the OS accomplish the following things?
  – How does it control the hardware?
  – What valuable services does it provide to applications?

Interaction with Hardware

• Windows and Linux take different views on the interaction with hardware in a PC.

• Windows Driver Model (WDM):
  – Creates drivers that are source-code compatible
  – Channels work of device driver into portions of code that are integrated into the operating system
  – Device driver becomes more streamlined with less code and work at greater efficiency
  – Implemented in all versions of Windows after ‘95

• Fedora 13 divides devices into major categories:
  – Character devices
  – Block devices
  – Network devices
**Windows Display Driver Model Architecture (WDDM)**

- The display driver model architecture for Windows Vista and later (WDDM) is composed of user-mode and kernel-mode parts. The following figure shows the architecture required to support the Windows Vista and later display driver model.

A graphics hardware vendor must supply the user-mode display driver and the display miniport driver. The user-mode display driver is a dynamic-link library (DLL) that is loaded by the Microsoft Direct3D runtime. The display miniport driver communicates with the Microsoft DirectX graphics kernel subsystem.

**Interaction with Hardware (continued)**

- The Windows view of hardware
  - Bus drivers:
    - Drive an I/O bus
    - Provide per-slot functionality that is device independent
  - Function drivers:
    - Drive an individual device
  - Filter drivers:
    - Filter I/O requests for a device, a class of devices, or a bus
### The Windows View of Hardware

- **Three kinds of WDM drivers:** bus, function, filter
  - **Bus drivers drive an I/O bus**
    - Provide per-slot functionality that is device independent
    - WDM support is provided for a number of buses
      - Examples: PCI Express, SCSI, Plug and Play serial I/O (USB, IEEE1394), Network Driver Interface Specification (NDIS)
  - **Function drivers drive an individual device**
    - WDM support is provided for USB device classes:
      - Examples: cameras/scanners, USB and IEEE 1394 storage devices, video capture over IEEE 1394, audio
  - **Filter drivers filter I/O requests for a device, a class of devices, or a bus**
    - Example: used by antivirus agent to intercept requests targeted at a file system that may contain viruses.

### The Fedora 13 View of Hardware

- **Device categories:** character, block, networking
  - **Character devices**
    - Can be accessed as streams of characters
    - Example: standard input device (keyboard), or standard output device (screen, or monitor), others such as serial ports.
  - **Block devices**
    - Handle I/O operations by block (usually 512 byte blocks)
      - Block may exceed 512 bytes by power of 2
    - Examples: disk drives, CD drives, loop devices (see next slide)
  - **Networking devices**
    - Handled differently by the kernel -- block or character devices use read and write calls
    - Require the kernel to manage packets that are transferred over a network where packets are transferred.
Linux Loop Device

- A loop device is a pseudo-device that makes a file accessible as a block device.
- Before use, a loop device must be connected to an existing file in the file system. The association provides the user with an API that allows the file to be used in place of a block special file (device file system). Thus, if the file contains an entire file system, the file may then be mounted as if it were a disk device.
- Files of this kind are often used for CD ISO images and floppy disc images. Mounting a file containing a file system via such a loop mount makes the files within that file system accessible. They appear in the mount point directory.

Interaction with Applications

- Operating systems provide a software platform
- Application programs run on software platform
- Application support provided by OS
  - Installing applications
  - Running applications
  - Managing disks and files
  - Connecting to a network
  - Printing documents
Installing Applications

• Typical software installation steps
  – Request made to OS to install application
  – OS locates program files on media, such as a CD
  – OS copies program files to hard drive
  – Program entry created on start or application menu
  – Application configured for use

Interaction with Applications

• Installing applications
  – Software is installed from a CD-ROM or is downloaded
  – Microsoft Windows Installer: installation and configuration service
  – Windows registry: maintains system configuration
  – Accessed by application installers or Registry Editor
  – YUM (Yellowdog Updater, Modified):
    • Automatic updater and package installer/remover for RPM packages
  – RPM (Red Hat Package Manager):
    • Determines how to safely install, remove, and update RPM packages
Running Applications

- OS tasks triggered by request to run an application:
  - Application user’s level of authorization is checked
  - Address space for the program is allocated
  - Program is loaded into the address space
  - Processes are scheduled for thread execution
  - Execution of thread or processes is monitored
- OS tasks performed during program execution:
  - Processor resources are shared by multitasking
  - I/O operations queued for execution
  - Data read from or written to requested device
  - Program terminated and memory released on request

Managing Disks and Files

- Data files arranged in system specific hierarchy:
  - Windows 7 implements a folder structure
  - Fedora 13 uses directories
- Windows 7:
  - View folder structure and file attributes with Explorer
  - Home directory stores data
  - Supports the use of libraries
  - Monitors folders that contain your files
  - Lets you access and arrange the files in different ways
Managing Disks and Files

• Data files arranged in system specific hierarchy
  – Windows implements a folder structure
  – Linux uses a directory similar to Windows folders

• Basic features of folder structure in Windows
  – View folder structure and file attributes with Explorer
  – Home folder is called My Documents
    • Subfolder located under Documents and Settings

• Basic features of directory structure in Linux
  – View directory structure and file names with `ls` and `tree`
  – Use the `/home` directory to store data

Managing Disks and Files

• **The root directory** (sometimes simply referred to as "root"), is the lowest-level entity since it can contain files or directories but cannot itself be contained in a directory!

  – It is designated by "\" (in Windows environment) or by "/" (in UNIX/Linux environment). There is only one root in the UNIX/Linux systems, but there is one for each partition in Microsoft Windows.

  – A directory containing another directory is known as a "parent directory". When moving from a directory to a parent directory, the latter is designated by ".:" on most systems (type "cd .." in DOS or UNIX to access a parent directory).
Managing Disks and Files (continued)

![My Documents folder](image)

**Figure 2-15** My Documents folder

Connecting to a Network

- File server: central repository used for file sharing
- Directories or folders may be shared on network:
  - Grant access to desired users only
- Protocols used to support file sharing over network:
  - **Server Message Block (SMB)**: used by Windows
  - **Samba**: used by Fedora 13
- Windows 7:
  - Provides the Network folder in Windows Explorer
- Fedora 13:
  - Uses Network command to connect to remote share
• Tools used to access files over network
  – Windows: Add Network Place Wizard
  – Linux: *mount* command
Printing Documents

• Printed page: primary output for most documents
• OS provides access to local and network printers
• Making printer available to applications:
  – Windows 7: uses the Add Printer Wizard
  – Fedora 13: uses the Add Printer program
• Using printer after installation and configuration:
  – Follow instructions provided by application

System Utilities

• Perform various tasks on your PC
• Key utilities:
  – Text editors
  – Configuration editors
  – System information monitors
Text Editors

- Used to enter, modify, and delete data in a text file
- Three text editors will be covered latter.

Why are text files important?

- Text files are used for many purposes:
  - Examples: writing scripts, storing source code
- System utilities interact best with text files
- Listing the contents of a text file:
  - DOS: uses the `type` command
  - Example: enter `type boot.ini` at c-prompt
  - Fedora 13: uses the `cat` command

Text Editors (continued)

Figure 2-14 Text file listed with DOS type command

Courtesy Course Technology/Cengage Learning
Operating System Configuration

- Registry:
  - Hierarchical database that contains configuration entries
- Windows 7 Configuration
  - Regedit: allows system administrators to edit Registry entries
- `visudo` in Fedora 13
  - The file `sudoers`, in the `/etc` directory:
    - Requires the use of a configuration editor
    - Edit this file with the `visudo command`

visudo in Linux

- System administrator is granted most permissions
- Users divided into groups based on function
  - Each group often has a group administrator
- `/etc/sudoers` file: holds configuration information that allows group administrators to execute the privileged instructions necessary to do their jobs.
  - Accessible to group administrators
  - Edit this file with the `visudo command`
    - E.g. Using `ubuntu`: `sudo visudo`
visudo in Linux (continued)

![Figure 2-20 Editing with visudo](image)

**sudoers file**

# User privilege specification root

root ALL = (ALL) ALL

As you can see there is basically one line

root ALL = (ALL) ALL

- This line means that the user root can execute from ALL terminals, acting as ALL (any) users, and run ALL (any) command.
- So the first part is the user, the second is the terminal from where the user can use sudo, the third is as which user he may act, and the last one, is which commands he may run.
- Let's see another example.

operator ALL = /sbin/poweroff

- This makes that users operator can from any terminal, run the command poweroff.
System Information

- **Windows 7 System Information**
  - System Information tool provides summary and configuration information

- **Fedora 13 System Profiler**
  - Allows you to see the type of CPU your computer uses
  - Click on Applications > System Tools > System Profiler and Benchmark
  - To review system summary, click Summary within System Profiler

Figure 2-18 System Information tool
*Courtesy Course Technology/Cengage Learning*
System Monitors

- PCs should be monitored to identify problems
- Tools used to show system performance:
  - Task Manager: available in Windows 7
  - System Monitor: available in Fedora 13
- A question to ask before using a monitor:
  - “What do I need to do?”

Windows Task Manager

- Three ways to start the Task Manager:
  - Press Ctrl+Alt+Delete
  - Press Ctrl+Shift+Esc
  - Right-click an empty area of the taskbar
    - Click Task Manager from the menu
- Applications tab
  - Determine if a program is hung or not responding
  - End program by selecting it and clicking End Task
  - Switch to a listed program
  - Start a new program
Windows Task Manager (continued)

• Processes tab
  – View the names of processes
  – Display CPU and memory usage
  – Stop selected program by clicking End Process
  – Sort any column of data by clicking the column name
• Performance tab
  – Review CPU usage and page file (PF) usage
  – View a variety of counts for other items
• Networking tab
  – View graph of network activity related to NIC card

Windows Resource Monitor

• Resource Monitor button on the Performance tab:
  – Monitors usage of four critical resources: CPU, hard disk, network, and memory
• Automatically opens to the Overview tab
Fedora 13 System Monitor

- Reviews process activity in real time
- Lists the most CPU-intensive tasks on the PC
- Contains more tabs than just the Processes tab
- Resources tab allows you to review real-time information about:
  - CPU History
  - Memory and swap usage
  - Network activity

Linux *top* Command

- Used to review processor activity in real time
- Information returned by *top* command
  - A listing of the most CPU-intensive tasks on the PC
  - A variety of details about the processor state
- Display is updated every five seconds
- Display is divided into upper and lower sections
  - Upper section provides data related to system status
    - Rows: top, Tasks, Cpu(s), Mem, Swap
  - Lower section identifies process information
    - Some columns: PID, USER, PR, VIRT, %CPU, TIME+
Linux *top* Command (continued)

![Fedora Core 4 top command](image)

**Applications**

- Directly provide services that interest the user
- Frequently bundles smaller programs
- Microsoft Office is a suite of programs:
  - Microsoft Word: industry standard word processor
  - Microsoft Excel: de facto standard for spreadsheets
  - Microsoft PowerPoint: used to create slide shows
- OpenOffice.org: alternative to Microsoft Office:
  - Writer: supports word processing and PDF creation
  - Calc: similar to Excel, includes other functions
  - Impress: like PowerPoint, includes other functions
Freeware for Windows 7

- Freeware: free software available for personal use
- Freeware is frequently distributed on the Web
- Freeware cannot be altered or sold:
  - The author of the software retains the copyright

Freeware for Windows 7 (continued)

<table>
<thead>
<tr>
<th>Category</th>
<th>Windows 7</th>
<th>Fedora 13</th>
</tr>
</thead>
<tbody>
<tr>
<td>Web browser</td>
<td>Microsoft Internet Explorer</td>
<td>Firefox, Netscape, Opera, Google Chrome</td>
</tr>
<tr>
<td>E-mail client</td>
<td>Outlook Express</td>
<td>Evolution, Thunderbird</td>
</tr>
<tr>
<td>Project management</td>
<td>Microsoft Project</td>
<td>Planner</td>
</tr>
<tr>
<td>Diagramming</td>
<td>Microsoft Visio</td>
<td>Dia</td>
</tr>
<tr>
<td>Image editing</td>
<td>Adobe Photoshop</td>
<td>GIMP</td>
</tr>
<tr>
<td>PDF reader</td>
<td>Adobe Acrobat</td>
<td>Adobe Acrobat</td>
</tr>
<tr>
<td>Database</td>
<td>Microsoft Access</td>
<td>Rekall RAD DBMS</td>
</tr>
<tr>
<td>FTP client</td>
<td>GlobalSCAPE CuteFTP</td>
<td>gFTP</td>
</tr>
</tbody>
</table>

Table 2-2 Business applications
Summary

• Software components of a PC are used to perform a variety of tasks
• DOS: first OS for the PC
• The operating system:
  – Manages resources by controlling the processor, memory, devices, storage, and the user interface
  – Interacts with both hardware and applications
• Common operating systems:
  – DOS, Windows, and Fedora 13

Summary (continued)

• The operating system:
  – Works with different system utilities to perform various tasks on your PC
  – Use different terminology and utilities to perform these tasks
• Commonly used applications on your PC:
  – Word processors
  – Spreadsheets
  – Web browser