1. The BIOS, having completed its test and setup functions, loads the boot code found in the master boot record and then transfers control of the system to it. At that point, the master boot record code is executed. If the boot device is a floppy disk, the process skips to step 7 below.

2. The next step in the process is the master boot code examining the master partition table. It first must determine if there is an extended DOS partition, then it must determine if there is a bootable partition specified in the partition table.

3. If the master boot code locates an extended partition on the disk, it loads the extended partition table that describes the first logical volume in the extended partition. This extended partition table is examined to see if it points to another extended partition table. If it does, this second table is examined for information about the *second* logical volume in the extended partition. Logical volumes in the extended partition have their extended partition table chained together one to the next. This process continues until all of the extended partitions have been loaded and recognized by the system.

4. Once the extended partition information (if any) has been loaded, the boot code attempts to start the primary partition that is marked active, referred to as the boot partition. If no boot partitions are marked active, then the boot process will terminate with an error. The error message is often the same as that which occurs if the BIOS could not locate a boot device, generally shown on screen as "No boot device", but also can show up as "NO ROM BASIC - SYSTEM HALTED". If there is a primary partition marked active and there is an installed operating system, the boot code will boot it. The rest of the steps presume this example is of an MS- DOS primary partition.

5. At this stage, the master or volume boot sector is loaded into memory and tested, and the boot code that it contains is given control of the remainder of the boot process. 6. The boot code examines the disk structures to ensure that everything is correct. If not, the boot process will end in an error here.

7. During the next step, the boot code searches the root directory of the device being booted for the operating system files that contain the operating system. For MS-DOS, these are the files "IO.SYS", "MSDOS.SYS" and "COMMAND.COM".

8. If no operating system files are found, the boot program will display an error

message similar to "Non-system disk or disk error - Replace and press any key when ready". Keep in mind that this message does not means that the system was never booted. It means that the BIOS examined the floppy disk for example and just rejected it because it couldn't boot an operating system. The volume boot code was indeed loaded and executed, as that is what posts the message when it can't find the operating system files.

9. In the final stages of the boot process, presuming that the operating system files are found, the boot program will load those operating system files into memory and transfer control to them. In MS-DOS, the first is IO.SYS and its code is executed.

10. SYS will then execute MSDOS.SYS. Then the more complete operating system code loads and initializes the rest of the operating system structures beginning with the command interpreter COMMAND.COM and then the execution of the CONFIG.SYS and AUTOEXEC.BAT files. At this point the operating system code itself has control of the computer.

MS-DOS Process Management

This relates to the Operating System's activity of managing the processor in the system, i.e. allocating and de-allocating of the processor to the process. The Operating System decides. Group C Page 12
the same based on the priority of the process depending if there exists any and certain predefined algorithms. Although MS-DOS is a single tasking operating system, this does not mean there can only be one program at a time in memory. However we can still load several programs into memory at one time under DOS. The only catch is that DOS only provides the ability for them to run one at a time in a very specific fashion. Unless the processes are cooperating, their execution profile follows a very strict pattern. That's why Dos exhibit Serial Multi tasking.
Users often wish to perform more than one activity at a time (load a remote file while editing a program) and uni programming does not allow this. So DOS put in things like memory resident programs that invoked asynchronously, but still have separation problems. One key problem with DOS is that there is no memory protection - one program may write the memory of another program, causing weird bugs.

Child Processes in DOS

In Dos we have one process and one thread.

When a DOS application is running, it can load and executing some other programs using the DOS EXEC function. Under normal circumstances, when an application (the parent) runs a second program (the child), the child process executes to completion and then returns to the parent. This is very much like a procedure call, except it is a little more difficult to pass parameters between the two. Group C Page 13

3.5 INPUTS/OUTPUT IN MSDOS

The core of MS-DOS is a device-independent input/output (I/O) handler, represented on a system disk by the hidden file MSDOS.SYS. It accepts requests from application programs to do high-level I/O, such as sequential or random access of named disk files, or communication with character devices such as the console. The handler processes these requests and converts them to a very low level form that can be handled by the I/O system. Because MSDOS.SYS is hardware independent, it is nearly identical in all MS-DOS versions provided by manufacturers with their equipment. The I/O system is totally device dependent and is represented on the disk by the hidden file IO.SYS. It is normally written by hardware manufacturers (who know their equipment best, anyway) with the notable exception of IBM, whose I/O system was written to IBM's specifications by Microsoft. The tasks required of the I/O system, such as outputting a single byte to a character device or reading a contiguous group of physical disk sectors into memory, are as simple as possible.

Departing From the Windows for MS-DOS I/O Model

The I/O system used in Windows for MS-DOS is based on, and limited by, capabilities of the underlying MS-DOS operating system. The device drivers at the core of this I/O system are commonly:

 Written in assembly language--they're not portable: they can only run on Intel 80x86 family processors.

 Monolithic in design, not layered--similar code for common tasks is repeated in each device driver for a given class of devices. Monolithic design also makes mixing and matching different file systems and device drivers impossible.

 Not designed for pre-emptive multitasking use--Windows has to perform many nasty tricks to allow even non-pre-emptive multitasking with non-multitasking MS-DOS and its non-multitasking device drivers.

 Incompatible with multiprocessor platforms--MS-DOS is inherently a singleprocessor OS, so MS-DOS device drivers are utterly incapable of synchronizing access to shared resources in a multiprocessor situation.
Group C Page 14

3.6 SECURITY

The security syntax set in MSDOS is masked on the command base and the password identifier which cannot be altered by someone who is not authorized. Password can be set for some programs starting from MSDOS like QBASIC.
The command – line is not prone to virus. Also it MSDOS cannot be used by someone who does not know the commands limiting the risk of harming the system. Group C Page 15

4.0 DESIGN ISSUES

MS-DOS Design Criteria

The primary design requirement of MS-DOS was CP/M-80 translation compatibility, meaning that, if an 8080 or Z80 program for CP/M were translated for the 8086 according to Intel's published rules, that program would execute properly under MS-DOS. Making CP/M-80 translation compatibility a requirement served to promote rapid development of 8086 software, which, naturally, Seattle Computer was interested in. There was partial success: those software developers who chose to translate their CP/M-80 programs found that they did indeed run under MS-DOS, often on the first try. Unfortunately, many of the software developers Seattle Computer talked to in the earlier days preferred to simply ignore MSDOS. Until the IBM Personal Computer was announced, these developers felt that CP/M-86 would be *the* operating system of 8086/8088 computers. Other concerns crucial to the design of MS-DOS were speed and efficiency. Efficiency primarily means making as much disk space as possible available for storing data by minimizing waste and overhead. The problem of speed was attacked three ways: by minimizing the number of disk transfers, making the needed disk transfers happen as quickly as possible, and reducing the DOS's "compute time," considered overhead by an application program. The entire file structure and disk interface were developed for the greatest speed and efficiency. The last design requirement was that MS-DOS be written in assembly language. While this

characteristic does help meet the need for speed and efficiency, the reason for including it is much more basic. The only 8086 software-development tools available to Seattle Computer at that time were an assembler that ran on the Z80 under CP/M and a monitor/debugger that fit into a 2K-byte EPROM (erasable programmable read-only memory). Both of these tools had been developed in house. Group C Page 16

5.0 IMPLEMENTATION

MSDOS is a single-user operating system. MS-DOS employs a command line interface and a batch scripting facility via its command interpreter, command.com (all operations to be carried out must be written through commands); without the knowledge of those commands, the user cannot execute a job. Despite its command-line interface, it is easy to learn. The commands of MSDOS are not case sensitive. MSDOS is supported and embedded in other operating system like *Microsoft windows, MacOs;* it can be launched in windows by:

(i) Start -> All programs -> accessories -> command prompt or
(ii) Start -> run -> type **cmd** -> ok
The MSDOS contains five files (IO.SYS, MSDOS.SYS, COMMAND.COM, CONFIG.SYS and AUTOEXEC.BAT). The commands in MSDOS can be categorized in two forms:
INTERNAL and **EXTERNAL** commands.
Group C Page 17

Internal commands are executed without loading a separate program file. Command.com is responsible for the execution of internal commands and the special batch commands. Internal commands are part of the command processor always available to be used. External command exists as executable files handled by separate programs from the DOS diskette. Each program is a member of the .COM or .EXE family. An external command can only be used when the disk containing the program is in drive. They can be used for peripheral devices like printer. A file's full name is called FILESPEC. The FILESPEC for a disk file has four parts: ***drive name, directory name, file name and extension.*** Characters like. " / \ [ ] : | < > + = ; , cannot be used in naming files because they mean other things in MSDOS. When you start up DOS, the current directory is automatically the root directory. The CD will change the current directory. Other folders in the root directory are called sub – directories. A file in any directory can be accessed by typing:

cd ROOT DIRECTORY\SUB-DIRECTORY (Level 1)\ SUB-DIRECTORY (level2)...\filename.ext. CLASSIFYING MS-DOS COMMANDS

Either internal or external command will be specified for each command of MSDOS in this text. MS-DOS commands fall roughly into three categories;
Group C Page 18

1. **Environment Commands:** These report on or affect the operating system environment. Examples are CLS (clear screen), TIME, DATE, VER (display MS-DOS version number), and HELP.

 BREAK (internal): Used from the DOS prompt or in a batch file or in the CONFIG.SYS file to set (or display) whether or not DOS should check for a Ctrl + Break key combination.
BREAK =on|off

 CLS (internal) – clears screen: Clears (erases) the screen. **CLS**

 DATE AND TIME (internal): Displays and/or sets the system date. **DATE mm-dd-yy** or **DATE**

 GRAPHICS (external): Provides a way to print contents of a graphics screen display.

GRAPHICS [printer type][profile] [/B][/R][/LCD][/PB:(id)] [/C][/F][/P(port)]

 MODE (external): Sets mode of operation for devices or communications.
MODE n
MODE LPT#[:][n][,][m][,][P][retry]
MODE [n],m[,T]
MODE (displaytype,linetotal)
MODE COMn[:]baud[,][parity][,][databits][,][stopbits][,][retry]
MODE LPT#[:]=COMn [retry]
MODE CON[RATE=(number)][DELAY=(number)]
MODE (device) CODEPAGE PREPARE=(codepage) [d:][path]filename
MODE (device) CODEPAGE PREPARE=(codepage list) [d:][path]filename
MODE (device) CODEPAGE SELECT=(codepage)
MODE (device) CODEPAGE [/STATUS]
MODE (device) CODEPAGE REFRESH

 VER (internal): Displays the DOS version number. **VER**

 SELECT (external): Formats a disk and installs country-specific information and keyboard codes (starting with DOS Version 6, this command is no longer available).

SELECT [d:] [d:][path] [country code][keyboard code]

2. **Directory and File Commands:** These manipulate files. Examples are COPY, DEL (delete), TYPE (display file to screen) and, DIR (directory - or list all files in current directory). It can be further divided into three (3);

(a) **Commands for disk maintenance**

 BACKUP (external): Makes a backup copy of one or more files. (In DOS Version 6, this program is stored on the DOS supplemental disk.)

BACKUP d:[path][filename] d:[/S][/M][/A][/F:(size)] [/P][/D:date] [/T:time] [/L:[path]filename]

 RESTORE (external): Restores to standard disk storage format files previously stored using the BACKUP command.

RESTORE d: [d:][path]filename [/P][/S][/B:mm-dd-yy] [/A:mm-ddyy][/ E:hh:mm:ss] [/L:hh:mm:ss] [/M][/N][/D]

 RECOVER (external): Resolves sector problems on a file or a disk. (Beginning with DOS Version 6, RECOVER is no longer available).

RECOVER [d:][path]filename or **RECOVER d:**

 VERIFY (internal): Turns on the verify mode; the program checks all copying operations to assure that files are copied correctly.
VERIFY on|off

 FORMAT (external): Formats a disk to accept DOS files. **FORMAT d:[/1][/4][/8][/F:(size)] [/N:(sectors)] [/T:(tracks)][/B|/S][/C][/V:(label)] [/Q][/U][/V]**

 SYS (external): Transfers the operating system files to another disk.
SYS [source] d:

 CHKDSK (external): Checks a disk and provides a file and memory status report.
CHKDSK [d:][path][filename] [/F][/V]

 DISKCOPY (external): Makes an exact copy of a diskette.
DISKCOPY [d:] [d:][/1][/V][/M]

 DISKCOMP (external): Compares the contents of two diskettes.
DISKCOMP [d:] [d:][/1][/8]

 LABEL (external): Creates or changes or deletes a volume label for a disk.
LABEL [d:][volume label]

 VOL (internal): Displays a disk's volume label.
VOL [d:] (b) **Commands for directory control**

 DIR (internal): Displays directory of files and directories stored on disk.
DIR [d:][path][filename] [/A:(attributes)] [/O:(order)]
[/B][/C][/CH][/L][/S][/P][/W]

 ASSIGN (external): Redirects disk drive requests to a different drive.
ASSIGN A:=B: [...] /sta

 MKDIR (internal): Creates a new subdirectory.
MKDIR (MD) [d:]path

 CHDIR (internal): Displays working (current) directory and/or changes to a different directory.
CHDIR (CD) [d:]path or **CHDIR (CD)[..]**

 RMDIR (internal): Removes a subdirectory.
RMDIR (RD) [d:]path

 TREE (external): Displays directory paths and (optionally) files in each subdirectory.
TREE **[d:][path] [/A][/F]**

 PATH (external): Sets or displays directories that will be searched for programs not in the current directory.
PATH; or **PATH [d:]path[;][d:]path[...]**

 JOIN (external): Allows access to the directory structure and files of a drive through a directory on a different drive.
JOIN d: [d:path] or **JOIN d: [/D]**

 SUBST (external): Substitutes a virtual drive letter for a path designation.
SUBST d: d:path or **SUBST d: /D**
(c) **Commands for file control:**

 COPY (internal): copies source file to target file and appends file.
COPY [d:][path]source [d:][path][target] [/V]
Or COPY [d:][path]filename+[d:][path]filename[...][d:][path][filename] [/V]

 COMP (external): Compares two groups of files to find information that does not match. **COMP [d:][path][filename] [d:][path][filename] [/A][/C][/D][/L][/N:(number)]**

 RNAME (internal): Changes the filename under which a file is stored.
RENAME (REN) [d:][path]filename [d:][path]filename

 ERASE/DELETE (internal): Deletes (erases) files from disk.
DEL (ERASE) [d:][path]filename [/P]

 TYPE (internal): Displays the contents of a file.
[d:][path]filename

 PRINT (external): Queues and prints data files.
PRINT [/B:(buffersize)] [/D:(device)] [/M:(maxtick)] [/Q:(value] [/S:(timeslice)] [/U:(busytick)] [/C][/P][/T] [d:][path][filename] [...]

 ATTRIB (external) : Sets or displays the read-only, archive, system, and hidden attributes of a file or directory.
ATTRIB [d:][path]filename [/S]
ATTRIB [+R|-R] [+A|-A] [+S|-S] [+H|-H] [d:][path]filename [/S]

3. **Utilities:** These perform some useful function. Examples are FORMAT (format a diskette) and EDIT (invoke MS-DOS text editor).
Group C Page 22

6.0 STRENGTH AND WEAKNESS

Strengths of Microsoft Disk Operating System

(a) **It has a good User interface**
MS-DOS employs a command line interface and a batch scripting facility via its command interpreter, command.com. MS-DOS was designed so users could easily substitute a different command line interpreter

(b) **MS-DOS compatibility with other Microsoft operating systems** users also desired a graphical user interface. Many programs running under MS-DOS tried to fill the void by creating their own graphical interface, such as Microsoft Word for DOS, XTree, and the Norton Shell. However, this required duplication of effort and did not provide much consistency in interface design (even between product lines). Non-Microsoft efforts to provide a consistent interface.

(c) **It has command line interpreter which is the most efficient way to manage files and run a computer program**. It can be used to run program like JAVA and C++, MYSQL.

(d) **Software Base**
There is a huge software base for developing software in DOS, which is another major strength.

Weakness of MS-DOS

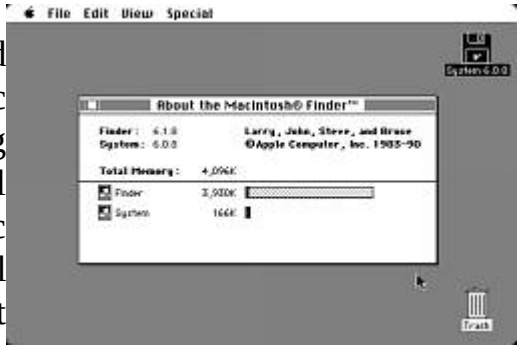
(a) **MSDOS** is a single user operating system.
(b) It does not support features like multi-tasking and multi-processing.
(c) DOS doesn't have built-in capability for scheduling or multithreading.
(d) You must also install interrupt handlers directly into the software application, and API calls tend to be through software interrupts rather than some other more direct procedural method instead.
(e) Equipment vendors supporting DOS tend to follow an approach of either providing raw spec sheets for their equipment or writing a pre-compiled binary object library that has to be linked into your software using a specific compiler.
Group C Page 23

CONCLUSION

MSDOS is a powerful operating system. Though it is an old operating system yet not outdated and versions of it are being released. There is a huge software base for developing software in DOS, which is another major strength**.** DOS controls the computer's hardware and provides an environment for programs to run. Everything you can do with a GUI can be done at the DOS prompt. It can be used to execute specific programs directly from the command prompt like sql queries, C++, C# and Java programs because it interface between with computer hardware and software effectively.

CHAPTER 5: MACINTOSH OPERATING SYSTEM
INTRODUCTION

Mac OS is a Graphical User Interface based operating system designed for Apple's Macintosh Computer Mac OS was named by the company Apple as "Mac System Software" in the beginning, a specially designed operating system only for 68000 first Motorola processors. With own Macintosh hardware, Mac OS takes up a special role in the world of desktop systems.



The first version was "System 1" and appeared bundled with the Mac in 1984. The classic desktop is designed as a single user operating system and almost completely hides the full path to files and directories. The graphic representation is reduced to the essence. Overall the interface is very easy to use and does not need the right mouse button for user interaction.

Starting with System 3.0, the used filesystem, Hierachical File System was used officially, which does not different between uppercase and lowercase letters.
System 5.0 was the first release to run several programs with the integrated MultiFinder at the same time. In 1988, system 6.0 came onto the market. It requires 1 MB RAM and can address up to 8 MB. The file system can organize hard disks up to 2 GByte with 65,536 files. Optionally applications run with the multi Finder in cooperative multitasking. For word processing are programs such as WriteNow, MacWrite II, and Microsoft Word 4.0 available In May 1991, system 7 came into existence. The new operating system needed 2 MB RAM, optionally it can be switched to 32-bit depending from the used hardware. New is the direct support of networks with file exchange, AppleScript as scripting language and display of colors. Balloons provide help for the user to use the interface. The TrueType fonts are scalable to any size.

The System Software 7.5 appeared in 1994 and requires at least 4 MB RAM. It was running both on 68000-Macs and Power Macintosh. In September 1996, the update System 7.5.5 includes all available bug fixes, Open Transport 1.1.2, current Ethernet driver and support for storage drive volumes up to 4 GB. With release 7.6 the company Apple changed the name from Mac Software System to Mac Operating System in 1997. Mac OS 8 by Apple appeared in July 1997. As minimum requirements are specified a 68040 or PowerPC processor, 32 MB RAM and 120 MB of free disk space. The CTRL key is used to display a specific context

menu for different actions. It makes it easier to copy files. **Mac OS 8.1**, Informations are stored more efficiently on the file system. The file system can handle up to 2 billion files with a current file size of up to 2 GB.

Mac OS 8.5 further optimized the stability and speed of the operating system, AppleScript is now up to 5 times faster than the previous version. The graphical display is accelerated by new QuickDraw routines. Copying files has become faster and increase the disk throughput. A tool for system maintaining detects and fixes errors on the file system automated. Following applications are included in current version: Finder 8.5 QuickTime Pro 3, Open Transport 2, Internet Explorer 4.01, Outlook Express 4.01, e.t.c.

The operating system Mac OS 9 has been developed under the name Sonata and released to the 23. October 1999. The installation requires 32 MB RAM with virtual memory. The free disk storage should be 150 up to 400 MB depending on the installation type. 50 new features are added in comparison to the previous version. This includes support for multiple users with password and access management for files and settings. The login is available through authentication by voice. MAC OS X

The operating system core Darwin is open source, Mac OS X works with preemptive multi-tasking and includes beside the new Graphical user interface (GUI) Aqua the classic GUI from Mac OS 9.

Mac OS X 10.0 came out in March 2001. To install are 128 MB RAM (256 MB RAM starting from Mac OS X 10.3.9) and 1.5 GB hard disk space (3.0 GByte starting from Mac OS X 10.2) provided. Mac OS X 10.5 requires at least 512 MB RAM and 9 GByte of free disk space.

- 32-bit or 64-bit processing

Field	**of**	**Application**
-	digital	photography
- 2-D	and	3-D animations
-	video	processing
-	audio	processing

Structure **Information**

- supports QuickTime
- graphical user interaction with the finder
- graphical representation by Quickdraw

Considerable performance and comfort improvements were carried out in version **Mac OS X 10.1**. The surface reacts quicker at user interaction, the system start was accelerated and the OpenGL performance increased noticeable.

Mac OS X 10.3 has now a GUI in metallic scheme and the optimized Finder. The use and access in heterogeneous networks was further simplified. 12 million MacOS X user were counted in October 2004.

According to Apple **Mac OS X 10.4** brings more than 200 new features. Features are the fast, system-wide and index-based search function named Spotlight, the Dashboard for easy access to small programms (Widgets), the Automator for the simplified composition of Applescripts for the automation of tasks. The Web browser Safari in version 2.0 now contained. Further novelty is the delivery at a DVD medium, an installation of CD-ROM is no longer possible.

First since the 10th January 2006 is MacOS X 10.4.4 next to the PowerPC version available for Intel based Macs. On the 6. June 2005 Steve jobs announced the switch to Intel processors. As further details became known that Apple had developed Mac OS X since 2000 internally also for the Intel platform.

Apple released the successor **MacOS X 10.5**, Leopard at the 26 october,2007. With more than 300 innovations MacOS offers the user an enhanced user interface with virtual desktops, a fast file preview and Dock with 3D effect.ose snapshot. The security of the operating system and applications is improved by 11 enhancements. The first update with bug fixes was released with Mac OS X 10.5.1 by Apple on November 15th, 2007. It contains general bug fixes for the operating system to improve stability, better compatibility and safety.
Mac OS X 10.5.2 cames with 125 bug fixes and smaller optimizations on January 24th, 2008.

Mac OS X 10.6 is a Mac computer with Intel Core 2 Duo processor with at least 1 GB memory and 5 GB free space ahead. This operating system no longer exists as PowerPC execution. Apple placed the focus development on performance and stability. It supports up to 16 TByte memory, it is optimized for multi core

processors, and is a pure 64-bit operating system. With the technology OpenCL graphics processor can speed up in specific applications calculation

PROCESS MANAGEMENT

Mac OS implemented non-preemptive multitasking. Although the scheduling algorithm was simple in the absence of preemption, it supported process priorities. A process could only be created by another process, except the initial process, which was created by the operating system as the "shell" process upon booting. The shell process ran the *Desktop Manager* application by default. The system's process management API included calls for creating, terminating, suspending, and resuming.

Technical History of Apple's Operating Systems processes.

Terminating a process also resulted in the termination of all its descendants. Examples of MAC process-management system calls included the following.

- make_process
- kill_process
- activate_process
- suspend_process
- info_process
- setpriority_process
- yield_process
- sched_class

System-level exceptions resulted in the termination of a process—a side effect of the execution of default exception handlers. Processes could install custom exception handlers, which were invoked with detailed exception context. Examples of Lisa exception-management system calls included the following.

- enable_excep
- disable_excep
- declare_excep_hdl
- signal_excep

Interprocess Communication

By default, a process was not allowed to access the logical address space of another process. Interprocess communication was possible through multiple mechanisms such as events, shared files, and shared memory. *Events* were structured messages consisting of a system-attached header and a sender-provided data block, transmitted between processes over named *channels*. A process could listen on a channel, waiting for messages to arrive. Alternatively, a process could register an exception handler and arrange for an exception to be generated upon message arrival.

Mac OS X Internals

Examples of mac event-channel management system calls included the following.

- make_event_chn
- kill_event_chn
- open_event_chn
- close_event_chn
- wait_event_chn

CHAPTER SIX: WINDOWS OPERATING SYSTEM

INTRODUCTION

Microsoft Windows is a series of software operating systems and graphical user interfaces produced by Microsoft. Microsoft first introduced an operating environment named Windows in November 1985 as an add-on to MS-DOS in response to the growing interest in graphical user interfaces (GUIs). Microsoft Windows came to dominate the world's personal computer market, overtaking Mac OS, which had been introduced previously. At the 2004 IDC Directions conference, it was stated that Windows had approximately 90% of the client operating system market. The most recent client version of Windows is Windows Vista; the most recent server version is Windows Server 2008. Vista's successor, Windows 7 (currently in public beta) is slated to be released between July 1, 2009 and June 30, 2010.
The term **Windows** collectively describes any or all of several generations of Microsoft operating system products. These products are generally categorized as follows:

- ✓ **History of Microsoft Windows**

Microsoft has taken two parallel routes in its operating systems. One route has been for the home user and the other has been for the professional IT user. The dual routes have generally led to home versions having greater multimedia support and less functionality in networking and security, and professional versions having inferior multimedia support and better networking and security.

The first version of Microsoft Windows, version 1.0, released in November 1985, lacked a degree of functionality and achieved little popularity, and was to compete with Apple's own operating system. Windows 1.0 is not a complete operating system; rather, it extends MS-DOS. Microsoft Windows version 2.0 was released in November, 1987 and was slightly more popular than its predecessor. Windows 2.03 (release date January 1988) had changed the OS from tiled windows to overlapping windows. The result of this change led to Apple Computer filing a suit against Microsoft alleging infringement on Apple's copyrights.

A Windows for Workgroups 3.11 desktop

Microsoft Windows version 3.0, released in 1990, was the first Microsoft Windows version to achieve broad commercial success, selling 2 million copies in the first six months. It featured improvements to the user interface and to

multitasking capabilities. It received a facelift in Windows 3.1, made generally available on March 1, 1992. Windows 3.1 support ended on December 31, 2001.
In July 1993, Microsoft released Windows NT based on a new kernel. NT was considered to be the professional OS and was the first Windows version to utilize preemptive multitasking.[citation needed]. Windows NT would later be retooled to also function as a home operating system, with Windows XP.
On August 24, 1995, Microsoft released Windows 95, a new, and major, consumer version that made further changes to the user interface, and also used preemptive multitasking. Windows 95 was designed to replace not only Windows 3.1, but also Windows for Workgroups, and MS-DOS. It was also the first Windows operating system to use Plug and Play capabilities. The changes Windows 95 brought to the desktop were revolutionary, as opposed to evolutionary, such as those in Windows 98 and Windows Me. Mainstream support for Windows 95 ended on December 31, 2000 and extended support for Windows 95 ended on December 31, 2001.

The next in the consumer line was Microsoft Windows 98 released on June 25, 1998. It was substantially criticized for its slowness and for its unreliability compared with Windows 95, but many of its basic problems were later rectified with the release of Windows 98 Second Edition in 1999. Mainstream support for Windows 98 ended on June 30, 2002 and extended support for Windows 98 ended on July 11, 2006.

As part of its "professional" line, Microsoft released Windows 2000 in February 2000. The consumer version following Windows 98 was Windows Me (Windows Millennium Edition). Released in September 2000, Windows Me implemented a number of new technologies for Microsoft: most notably publicized was "Universal Plug and Play."

In October 2001, Microsoft released Windows XP, a version built on the Windows NT kernel that also retained the consumer-oriented usability of Windows 95 and its successors. This new version was widely praised in computer magazines. It shipped in two distinct editions, "Home" and "Professional", the former lacking many of the superior security and networking features of the Professional edition. Additionally, the first "Media Center" edition was released in 2002,[20] with an emphasis on support for DVD and TV functionality including program recording and a remote control. Mainstream support for Windows XP ended on April 14, 2009. Extended support will continue until April 8, 2014.

In April 2003, Windows Server 2003 was introduced, replacing the Windows 2000 line of server products with a number of new features and a strong focus on security; this was followed in December 2005 by Windows Server 2003 R2.
On January 30, 2007 Microsoft released Windows Vista. It contains a number of new features, from a redesigned shell and user interface to significant technical changes, with a particular focus on security features. It is available in a number of different editions, and has been subject to some criticism.

- ✓ **Early versions**

Windows 1.0, Windows 2.0, and Windows 2.1x

The history of Windows dates back to September 1981, when the project named "Interface Manager" was started. It was announced in November 1983 (after the Apple Lisa, but before the Macintosh) under the name "Windows", but Windows 1.0 was not released until November 1985. The shell of Windows 1.0 was a program known as the MS-DOS Executive. Other supplied programs are Calculator, Calendar, Cardfile, Clipboard viewer, Clock, Control Panel, Notepad, Paint, Reversi, Terminal, and Write. Windows 1.0 does not allow overlapping windows, due to Apple Computer owning this feature. Instead all windows are tiled. Only dialog boxes can appear over other windows.

Windows 2.0 was released in October 1987 and featured several improvements to the user interface and memory management. Windows 2.0 allowed application windows to overlap each other and also introduced more sophisticated keyboard-shortcuts. It could also make use of expanded memory.

Windows 2.1 was released in two different flavors: Windows/386 employed the 386 virtual 8086 mode to multitask several DOS programs, and the paged memory model to emulate expanded memory using available extended memory. Windows/286 (which, despite its name, would run on the 8086) still ran in real mode, but could make use of the high memory area.

The early versions of Windows were often thought of as simply graphical user interfaces, mostly because they ran on top of MS-DOS and used it for file system services. However, even the earliest 16-bit Windows versions already assumed many typical operating system functions; notably, having their own executable file format and providing their own device drivers (timer, graphics, printer, mouse, keyboard and sound) for applications. Unlike MS-DOS, Windows allowed users to

execute multiple graphical applications at the same time, through cooperative multitasking.

Windows implemented an elaborate, segment-based, software virtual memory scheme, which allowed it to run applications larger than available memory: code segments and resources were swapped in and thrown away when memory became scarce, and data segments moved in memory when a given application had relinquished processor control, typically waiting for user input.

Windows 3.0 and Windows 3.1x

Windows 3.0 (1990) and Windows 3.1 (1992) improved the design, mostly because of virtual memory and loadable virtual device drivers (VxDs) which allowed them to share arbitrary devices between multitasked DOS windows. Also, Windows applications could now run in protected mode (when Windows was running in Standard or 386 Enhanced Mode), which gave them access to several megabytes of memory and removed the obligation to participate in the software virtual memory scheme. They still ran inside the same address space, where the segmented memory provided a degree of protection, and multi-tasked cooperatively. For Windows 3.0, Microsoft also rewrote critical operations from C into assembly, making this release faster and less memory-hungry than its predecessors. With the introduction of the Windows for Workgroups 3.11, Windows was able to bypass DOS for file management operations using 32-bit file access.

Windows 95, Windows 98, and Windows Me

Windows 95 featured a new user interface, supported long file names, could automatically detect and configure installed hardware (plug and play), natively ran 32-bit applications, and featured several technological improvements that increased its stability over Windows 3.1. Windows 95 uses pre-emptive multitasking and runs each 32-bit application in a separate address space. This makes it harder for a single buggy application to crash the whole system. It was still not a secure multi-user operating system like Windows NT as a strict separation between applications was not enforced by the kernel. The API was a subset of the Win32 API supported by Windows NT, notably lacking support for Unicode and functions related to security. Windows 95 was now bundled together with MS-DOS 7.0, however its role was mostly delegated to that of a boot loader.
There were several releases of Windows 95; the first in 1995, with Service Pack 1 following in December which included Internet Explorer 2.0. Subsequent versions

were only available with the purchase of a new computer and were called OEM Service Releases. OSR1 was equivalent to Windows 95 with SP1. OSR2 (also called Windows 95 B) included support for FAT32 and UDMA and shipped with Internet Explorer 3.0. OSR 2.1 included basic support for USB and OSR 2.5 (also called Windows 95 C) shipped with Internet Explorer 4.0.
Microsoft's next OS was Windows 98, which had two versions; the first in 1998 and the second, named Windows 98 Second Edition, in 1999.
In 2000, Microsoft released Windows Me (Me standing for Millennium Edition), which used the same core as Windows 98 but adopted some aspects of Windows 2000 and removed the "boot in DOS mode" option. It also added a new feature called System Restore, allowing the user to set the computer's settings back to an earlier date. Me is also the last DOS-based Windows release which does not include Microsoft Product Activation.

Windows NT family

The NT family of Windows systems was fashioned and marketed for higher reliability business use, and was unencumbered by any Microsoft DOS patrimony. The first release was MS Windows NT 3.1 (1993, numbered "3.1" to match the consumer Windows version, which was followed by NT 3.5 (1994), NT 3.51 (1995), NT 4.0 (1996), and Windows 2000 (2000). 2000 is the last NT-based Windows release which does not include Microsoft Product Activation. NT 4.0 was the first in this line to implement the "Windows 95" user interface (and the first to include Windows 95's built-in 32-bit runtimes). Microsoft then moved to combine their consumer and business operating systems with Windows XP, coming in both home and professional versions (and later niche market versions for tablet PCs and media centers); they also diverged release schedules for server operating systems. Windows Server 2003, released a year and a half after Windows XP, brought Windows Server up to date with MS Windows XP. After a lengthy development process, Windows Vista was released toward the end of 2006, and its server counterpart, Windows Server 2008 was released in early 2008. In 2009, Windows 7 and Windows Server 2008 R2 entered beta. Microsoft plans to release Windows 7 in late 2009 or early 2010.
Windows CE, Microsoft's offering in the mobile and embedded markets, is also a true 32-bit operating system that offers various services for all sub-operating workstations.

Windows CE

Windows CE (officially known as Windows Embedded), is an edition of Windows that runs on minimalistic computers, like satellite navigation systems, and uncommonly mobile phones. Windows Embedded runs as CE, rather than NT, which is why it should not be mistaken for Windows XP Embedded, which is NT. Windows CE was used in the Sega Dreamcast along with Sega's own proprietary OS for the console.

Windows Lifecycle Policy

Microsoft has stopped releasing updates and hotfixes for many old Windows operating systems, including all versions of Windows 9x, and earlier versions of Windows NT. Windows versions prior to XP are no longer supported, with the exception of Windows 2000, which is currently in the Extended Support Period, that will end on July 13, 2010. No new updates are created for unsupported versions of Windows.

DESIGN ISSUES

WINDOWS RESOURCE MANAGEMENT

This has do with the management of all the resources in the component of the Operating System e. g the Memory, the Central Processing Unit, I/O devices and other components.

✓ PROCESS MANAGEMENT

Processor

A circuit designed to automatically perform lists of logical and arithmetic operations.
Unlike microprocessors, processors may be designed from discrete components rather than be a monolithic integrated circuit.

A process is running program containing one or more threads. A process encapsulates the protected memory and environment for its threads.

Processes and Threads

In addition to being a preemptive multitasking operating system, Windows NT is also multithreaded, meaning that more than one thread of execution (or ***thread***) can execute in a single task at once.

A process comprises:

- A private memory address space in which the process's code and data are stored.
- An access token against which Windows NT makes security checks.
- System resources such as files and windows (represented as object handles).
- At least one thread to execute the code.

A thread comprises:

- A processor state including the current instruction pointer.
- A stack for use when running in user mode.
- A stack for use when running in kernel mode.

Since processes (not threads) own the access token, system resource handles, and address space, threads do NOT have their own address spaces nor do they have their own access token or system resource handles. Therefore, all of the threads in a process SHARE the same memory, access token, and system resources (including quota limits) on a "per-process" rather than a "per-thread" basis. In a multithreaded program, the programmer is responsible for making sure that the different threads don't interfere with each other by using these shared resources in a way that conflicts with another thread's use of the same resource. (As you might suspect, this can get a little tricky.)

Why Use Multithreading?

Multithreading provides a way to have more than one thread executing in the same process while allowing every thread access to the same memory address space. This allows very fast communication among threads. Threads are also easier to create than processes since they don't require a separate address space.

Inside Windows NT, processes and threads are represented as objects that are created, maintained, and destroyed by the Process Manager. These Process Manager process and thread objects contain simpler kernel process and thread objects.

Some typical examples of the use of multiple threads are using a background thread to print a document in a word processor and to recalculate a spreadsheet. When a new thread is created to do these tasks, the main thread can continue responding to user input. A single-threaded application can't respond to user input until it's done printing or recalculating or whatever.

On a uniprocessor platform, the use of multiple threads allows a user to continue using a program even while another thread is doing some lengthy procedure. But only one thread executes at a time.

On a multiprocessor platform, more than one processor may be running different threads in the same process. This has the potential for very significantly speeding up the execution of your program.

Sharing A Single Address Space--Synchronizing Access To Data

Running each process in its own address space had the advantage of reliability since no process can modify another process's memory. However, **all of a process's threads run in the same address space and have unrestricted access to all of the same resources, including memory.** While this makes it easy to share data among threads, it also makes it easy for threads to step on each other. As mentioned before, multithreaded programs must be specially programmed to ensure that threads don't step on each other.

A section of code in Windows operating system that modifies data structures shared by multiple threads is called a ***critical section***. It is important than when a critical section is running in one thread that no other thread be able to access that data structure. Synchronization is necessary to ensure that only one thread can execute in a critical section at a time. This synchronization is accomplished through the use of some type of Windows synchronization object. Programs use Windows synchronization objects rather than writing their own synchronization both to save coding effort and for efficiency: when you wait on a Windows synchronization object, you do NOT use any CPU time testing the object to see when it's ready.

Windows provides a variety of different types of synchronization objects that programs can use to coordinate threads' access to shared data structures. Synchronization objects remember their states and can be set and tested in one uninterruptable step. They also cause the thread to be suspended while waiting on an object and to automatically restart when the other thread signals that it's done.

During the initialization of a program, the program creates a synchronization object for each data structure or object that will be shared among threads.

EVERY critical section will have the following structure:

1. Wait on the synchronization object before accessing the data structure. The Windows waiting API insures that your thread is suspended until the synchronization object becomes unlocked. As soon as the synchronization object becomes unlocked, Windows sets the synchronization object to "locked" and restarts your thread.
2. Access the data structure. (This is the critical section.)
3. Unlock the synchronization object so that the data can be accessed by other threads.

The first step is critical because if it's omitted then any thread can access the data structure while you're accessing. The last step is also critical--it it's omitted, then no thread will be able to access the data even after you're done.

Using this technique on every critical section insures that only one thread can access the data at a time.

The Life Cycle Of A Thread

Each thread has a ***dispatcher state*** that changes throughout its lifetime.

The most important dispatcher states are:

- Running: only one thread per processor can be running at any time.
- Ready: threads that are in the Ready state may be scheduled for execution the next time the kernel dispatches a thread. Which Ready thread executes is determined by their priorities.
- Waiting: threads that are waiting for some event to occur before they become Ready are said to be waiting. Examples of events include waiting for I/O, waiting for a message, and waiting for a synchronization object to become unlocked.

SCHEDULLING

The Kernel's Dispatcher

The kernel's dispatcher performs scheduling and context switching.

Thread scheduling is the act of determining which thread runs on each processor at a given time.

Context switching is the act of saving one thread's volatile state (CPU register contents) and restoring another thread's state so it can continue running where it previously left off.

How Thread Priorities Affect Scheduling

The kernel's dispatcher schedules threads to run based a 32-level priority scheme. Windows guarantees that the threads that are ready that have the highest priority will be running at any given time. (That's one thread on a single-processor system.) Threads with a priority of 31 will be run before any others, while threads with a priority of 0 will run only if no other threads are ready. The range of priorities is divided in half with the upper 16 reserved for real-time threads and the lower 16 reserved for variable priority threads.

Real-time threads run at the same priority for their entire lifetime. They are commonly used to monitor or control systems that require action to be taken at very precise intervals. These threads run at higher priorities than all variable priority threads, which means that they must be used sparingly.

Variable priority threads are assigned a base priority when they are created. (A thread's base priority is determined by the process to which the thread belongs.) The priority of such threads can be adjusted dynamically by the kernel's dispatcher. A thread's dynamic priority can vary up to two priority levels above or below its base priority.

The dispatcher maintains a ***priority queue*** of ready tasks. When prompted to reschedule, it changes the state of the highest priority task to Standby. When the conditions are right, a context switch is performed to begin the thread's execution and the thread goes into the Ready state.

Lower priority threads will always be preempted when a higher priory thread enters the ready state. This is true even if the lower priority thread has time remaining in its quantum, or if the lower priority thread is running on a different processor.

Performance Tuning

In order to get the computer system to perform as users expect, Windows changes the priorities of threads over time.

Each process has a base priority. Threads in a process can alter their base priority by up to two levels up or down.

Depending on the type of work the thread is doing, Windows may also adjust the thread's dynamic priority upwards from its base priority. For instance:

- Threads that are waiting for input get a priority boost, as do threads in the foreground process. This makes the system responsive to the user.
- Threads get a priority boost after completing a voluntary wait.
- All threads periodically get a priority boost to prevent lower priority threads from holding locks on shared resources that are needed by higher priority threads.
- Compute-bound threads get their priorities lowered.

Scheduling On Multiprocessor Systems

A ***multiprocessing*** operating system is one that can run on computer systems that contain more than one processor. Windows is a ***symmetric multiprocessing (SMP)*** system, meaning that it assumes that all of the processors are equal and that they all have access to the same physical memory. Therefore, Windows can run any thread on any available processor regardless of what process, user or Executive, owns the thread.

There are also asymmetric multiprocessing (ASMP) systems in which processors are different from each other--they may address different physical memory spaces, or they may have other differences. These operating systems only run certain processes on certain processors--for instance, the kernel might always execute on a particular processor.

The design of Windows supports ***processor affinity***, whereby a process or thread can specify that it is to run on a particular set of processors, but this facility isn't supported in the first release.

Windows uses the same rules for scheduling on a multiprocessor system as it does on a single processor system, so at any given time the threads that are ready and have the highest priorities are actually running.

Using Task Manager

The *Task Manager* utility shows the applications and processes that are currently running on your computer, as well as CPU and memory usage information. To access Task Manager, press Ctrl□Alt□Delete and click the Task Manager button. Alternatively, right-click an empty area in the Taskbar and select Task Manager from the pop-up menu.

The Task Manager dialog box has four main tabs: Applications, Processes, Performance, and Networking. These options are covered in the following subsections.

Managing Application Tasks

The Applications tab of the Task Manager dialog box, shown in **Figure iii**, lists all of the applications that are currently running on the computer. For each task, you will see the name of the task and the current status (running, not responding, or stopped).

To close an application, select it and click the End Task button at the bottom of the dialog box. To make the application window active, select it and click the Switch To button. If you want to start an application that isn't running, click the New Task button and specify the location and name of the program you wish to start.

- Processor Scheduling, which allows you to optimize the processor time for running programs
  or background services
- Memory Usage, which allows you to optimize memory for programs or system cache
- Virtual Memory, which is used to configure the paging file

Stopping Processes

Process Description

System Idle Process A process that runs when the processor is not executing any other threads

smss.exe Session Manager subsystem

csrss.exe Client-server runtime server service

mmc.exe Microsoft Management Console program (used to track resources used by MMC snap-ins such as System Monitor)
explorer.exe Windows Explorer interface
Ntvdm.exe MS-DOS and Windows 16-bit application support

Managing Process Priority

You can manage process priority through the Task Manager utility or through the start command-line utility. To change the priority of a process that is already running, use the Processes tab of Task Manager. Right-click the process you want to manage and select Set Priority from the pop-up menu. You can select from RealTime, High, AboveNormal, Normal, BelowNormal, and Low priorities.

Options for the start Command-Line Utility

Option Description

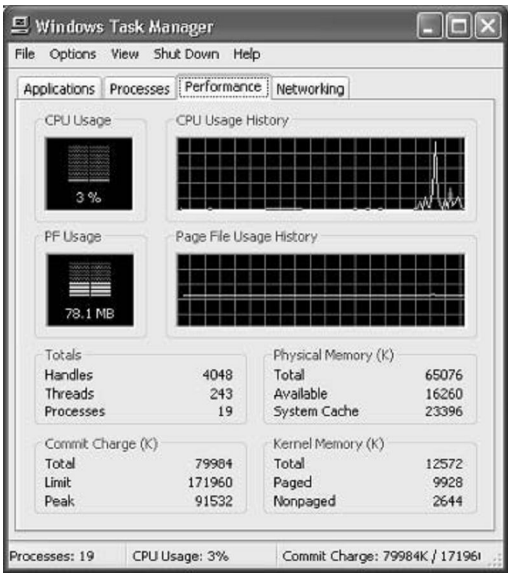
- low Starts an application in the Idle priority class.
- normal Starts an application in the Normal priority class.
- high Starts an application in the High priority class.
- realtime Starts an application in the RealTime priority class.
- abovenormal Starts an application in the AboveNormal priority class.
- belownormal Starts an application in the BelowNormal priority class.
- min Starts the application in a minimized window.
- max Starts the application in a maximized window.
- separate Starts a Windows 16-bit application in a separate memory space. By default Windows 16-bit applications run in a shared memory space, NTVDM, or NT Virtual DOS Machine.
- shared Starts a DOS or Windows 16-bit application in a shared memory space.

Managing Performance Tasks

The Performance tab shows the following information:

- CPU Usage, in real time and in a history graph
- Page File Usage, in real time and in a history graph
- Totals for handles, threads, and processes
- Physical Memory statistics
- Commit Charge memory statistics

- Kernel Memory statistics



A task manager windows

Scheduling Tasks

Windows includes a *Task Scheduler* utility that allows you to schedule tasks to occur at specified intervals. You can set any of your Windows programs to run automatically at a specific time and at a set interval, such as daily, weekly, or monthly. For example, you might schedule your Windows Backup program to run daily at 2:00 a.m.

Tuning and Upgrading the Processor

If you suspect that you have a processor bottleneck, you can try the following solutions:

- Use applications that are less processor-intensive.
- Upgrade your processor.
- If your computer supports multiple processors, add one. Windows can support up to two processors, which will help if you use multithreaded applications. You can also use processor affinity to help manage processor-intensive applications.

Monitoring and Optimizing the Processor

Processor bottlenecks can develop when the threads of a process require more processing cycles than are currently available. In this case, the process will wait in a processor queue and system responsiveness will be slower than if process requests could be immediately served.
The most common causes of processor bottlenecks are processor-intensive applications and other subsystem components that generate excessive processor interrupts (for example, disk or network subsystems).
In a workstation environment, processors are usually not the source of bottlenecks. You should still monitor this subsystem to make sure that processor utilization is at an efficient level.

MEMORY MANAGEMENT

The memory manager implements virtual memory, provides a core set of services such as memory mapped files, copy-on-write memory, large memory support, and underlying support for the cache manager.

Memory management in Microsoft Windows operating systems has evolved into a rich and sophisticated architecture, capable of scaling from the tiny embedded platforms (where Windows executes from ROM) all the way up to the multi-terabyte NUMA configurations, taking full advantage of all capabilities of existing and future hardware designs.

With each release of Windows, memory management supports many new features and capabilities. Advances in algorithms and techniques yield a rich and sophisticated code base, which is maintained as a single code base for all platforms and SKUs.

Memory management improvements in Windows Vista focused on areas such as dynamic system address space, enhanced NUMA and large system/page support, advanced video model support, I/O and section access, and robustness and diagnosability.

Among other things, a multiprogramming operating system kernel must be responsible for managing all system memory which is currently in use by programs. This ensures that a program does not interfere with memory already used by another program. Since programs time share, each program must have independent access to memory.

Cooperative memory management, used by many early operating systems assumes that all programs make voluntary use of the kernel's memory manager, and do not exceed their allocated memory.

Memory protection enables the kernel to limit a process' access to the computer's memory. Various methods of memory protection exist, including memory segmentation and paging. All methods require some level of hardware support (such as the 80286 MMU) which doesn't exist in all computers.

Virtual memory

The use of virtual memory addressing (such as paging or segmentation) means that the kernel can choose what memory each program may use at any given time, allowing the operating system to use the same memory locations for multiple tasks.

If a program tries to access memory that isn't in its current range of accessible memory, but nonetheless has been allocated to it, the kernel will be interrupted in the same way as it would if the program were to exceed its allocated memory. When the kernel detects a page fault it will generally adjust the virtual memory range of the program which triggered it, granting it access to the memory requested. This gives the kernel discretionary power over where a particular application's memory is stored, or even whether or not it has actually been allocated yet.

In modern operating systems, application memory which is accessed less frequently can be temporarily stored on disk or other media to make that space available for use by other programs. This is called swapping, as an area of memory can be used by multiple programs, and what that memory area contains can be swapped or exchanged on demand.

- ✓ **ERROR HANDLING**

Existing operating systems include error handling tools that identify and log errors that occur during the operation of the operating systems and provide a user with the ability to view the logged errors. Although most applications running in the

operating system's environment provide error messages when errors occur, these error messages are often technical in nature and may not be easily understood by the user. Therefore, the error handling tools not only identify and log these errors but, in addition, they describe the errors to the user in an easily comprehensible format. For example, the Windows NT® operating system by Microsoft Corporation includes an event log that collects error messages from applications, device drivers and the operating system itself to a common location that the user can access. The Windows NT® operating system also includes an event viewer that permits the user to view the error messages in the order of their occurrences.

For existing error handling tools, the client application must have advance knowledge of the possible types of service failures. Otherwise, the client application would not be able to provide meaningful information regarding the nature of the error. This poses problems in a client-server environment because the client application and applications of the service providers are loosely coupled and developed independently, and it is not practical to change an application as its underlying service provider evolves. It is, therefore, difficult for a client application to obtain meaningful information for an error message within a client/server environment.

The present invention resolves the above problems by providing a mechanism for a client application that identifies the source of a service error and obtains detailed error information at the time the error occurs. In particular, the mechanism accesses the error message information specific to the service that encounters the error and, then, reports that information as part of its error handling operation. The present invention also functions in a plurality of nested subsystems in which a subsystem that detects a particular problem is identified among a group of subsystems that are called in a nested manner. Accordingly, the present invention automatically handles the operation of reporting events whose source subsystem is different from that of the reporting subsystem without requiring the reporting subsystem to have advance knowledge of the possible errors that could be encountered.

Using Event Viewer