The parent program shown in Figure 8.2 illustrates how *execve* and wait can be used. In this complete example, the parent process creates a child using the *fork* command and then simulates execution with the *printf* and sleep function calls.

```
A Simple Parent Program
#include <sys/wait.h>
#define NULL 0
int main (void) {
if (fork() = = 0) {  /* This is the child process */
      execve ("child", NULL, NULL);
       exit (0); /* Should never get here, terminate */
}
/* Parent code here */
       printf ("Process [%d]: Parent in execution…./n",
getpid ());
sleep (2);
    if (Wait (NULL) > 0) /* Child terminating */
    printf ("Process [%d]: Parent detects terminating child /n",
        getpid ());
    printf ("Process [%d]: Parent terminating…/n", getpid());
}
```

Figure 8.2: Sample Parent Program

In a more practical application, the parent would execute from its own address space in place of the *printf* and *sleep* calls.

You have just learnt how to use UNIX system calls. Following are the rest of UNIX system calls related to process management:

- acct : enable/disable process accounting
- alarm : set a process alarm clock
- exit : terminate a process
- fork : create a new process
- getpid : get process, process group and parent process ID
- getuid : get real user, effective user real group and effective group ID
- kill : send a signal to a process or group of processes
- msgctl : message control operation
- msgop : message operation
- nice : change priority of a process
- pause : suspend until
- pipe : create an inter-process channel

- profil : execution time profile
- ptrace : process trace
- semctl : semaphor control operations
- semget : get set of semaphor
- semop : semaphor operations
- setpgrp : set process group ID
- setuid : set group and user ID
- signal : specify what to do when a signal is received
- stime : set time
- sync : update super block
- time : get time
- times : get process and child process times
- ulimit : get user upper limits
- uname : get the name of the current operating system
- ulink : remove directory entry
- Wait : wait for the child process to stop or terminate

All these function calls are used more or less the same way. Some of the most important of them will be explained through this unit.

8.4.2 Memory Management Functions

Like any currently executing process, the kernel also resides in the main memory so long as computer is operational. When a program is compiled, a set of addresses are generated in the program by the compiler. These represent addresses of variables or addresses of instructions such as functions. The addresses generated by the compiler are for a virtual machine. The addresses, therefore, are relative to each other not absolute in terms of memory addresses where they will be loaded eventually. It assumes that no other program will be However, when you run the program, some space is allocated to it in the main memory by the kernel. But the virtual addresses generated by the compiler might not resemble the physical addresses occupied in the machine. Then the kernel tries to coordinate with the machine hardware in order to map the compiler generated address with the physical machine addresses.

UNIX divides the available memory into system memory and application memory. It loads itself into system memory and creates data structures it will use in its operation into this area of memory.

The application memory area is where creates processes and loads user's programs in them, and allocates memory for the same. The application memory area is divided into global stack, local stack and heap. The global and static type of variables, functions etc., are assigned space in these memory areas. UNIX provides system calls to affect loading and unloading of programs into and out of the processes.

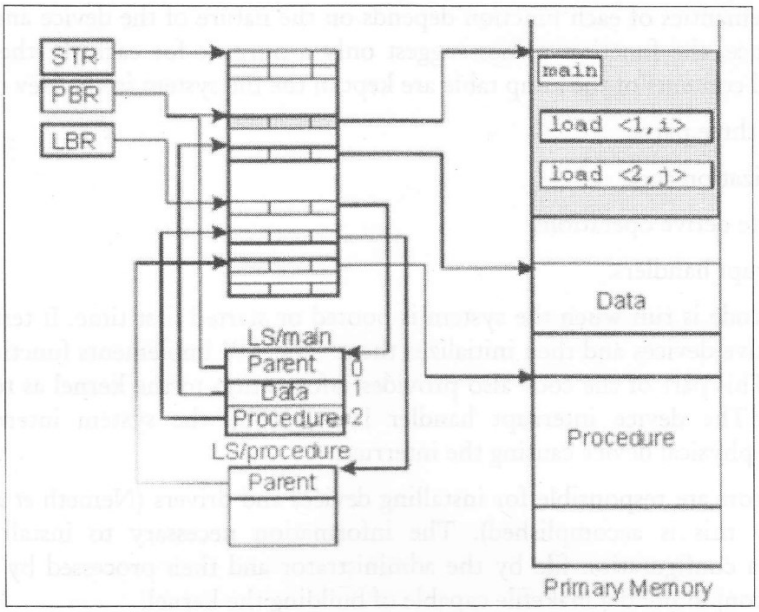

Figure 8.3: Memory Segmentation in UNIX

Internally, UNIX uses paging with segmentation methods (Figure 8.3) to manage memory. In addition to these primitive operations, UNIX provides library functions like *malloc*, for allocating memory to a process and objects dynamically. Other memory related system calls are:

- brk : change data segment space allocation
- shmop : shared memory operations
- shmctl : shared memory control operation
- shmget : get shared memory segment
- plock : lock process, text or data in memory
- msgget : get message queue

8.5 IMPLEMENTATION OF UNIX

UNIX treats every device the same way, just as it treats a file. For each device, however, it has device drivers for low level communication specific to the device.

8.5.1 UNIX Device Drivers

UNIX explicitly attempts to simplify the application programming model for files and devices by making files and devices as similar as possible. The UNIX device driver API (Application Program Interface) uses operation names similar to the file interface, although they apply to physical devices rather than to storage abstraction. Device drivers are intended to be accessed by user space code. If an application accesses a driver, it uses one of two standardized interfaces: The block device interface or the character device interface. Both interfaces provide a fixed set of functions to the user programs.

When a user program calls the driver, it performs a system call. The kernel searches the entry point for the device in the block or character in direct reference table (the jump table) and then calls the entry

point. The exact semantics of each function depends on the nature of the device and the intent of the driver design. Hence, the function names suggest only a purpose for each. In the UNIX family of systems, the logical contents of the jump table are kept in the file system in the/dev directory.

A Unix driver has three parts:

- System initialization code
- Code to initiate derive operations
- Devices interrupt handlers.

The initialization code is run when the system is booted or started first time. It tests for the physical presence of respective devices and then initializes them. The API implements functions for a subset of the entry points. This part of the code also provides information to the kernel as to which functions are implemented. The device interrupt handler is called by the system interrupt handler that corresponds to the physical device causing the interrupt.

System administrators are responsible for installing devices and drivers (Nemeth *et al* [1995, ch. 6] for details about how this is accomplished). The information necessary to install a driver can be incorporated into a configuration file by the administrator and then processed by the configuration builder tool, /etc/conj in build a makefile capable of building the kernel.

Apart from the system calls introduced so far, UNIX has a host of system calls to affect I/O manipulation. Some of them alongwith their purposes are:

- write : write on a file
- utime : set file access and modification times
- ustat : get file system statistics
- ulink : remove directory entry
- umount : unmount a file system
- umask : set and get file creation mask
- stat : get file status
- read : read from a file
- open : open for reading or writing
- mount : mount a file system
- mknod : make a directory or special or ordinary file
- lseek : move read/write pointer
- link : link to a file
- fcntl : file control
- exec : execute a file
- dup : duplicate an open file descriptor
- creat : create a new file or rewrite an existing one
- close : close a file descriptor
- chroot : change to root

- chown : change owner or group of file
- chmod : change mode of file
- chdir : change directory
- access : determine accessibility of a file

These are basic services made available to the user by UNIX. These services are very primitive and therefore, are not very convenient for the programmer. We will learn how to create library functions that are more powerful and flexible than system calls.

8.5.2 UNIX Kernel

The services provided by the operating system are infact provided by the kernel. The kernel performs various operations and acts as a user interface. The services provided by the kernel are given below:

1. It controls the fate and state of various processes such as their creation, termination and suspension.
2. The kernel allocates main memory for an executing process. The kernel allows the processes to share portions of their address space. It keeps the private space of processes secure and doesn't allow tampering from other processes. However, if the free memory is low with the system, then the kernel frees out some memory by writing a process temporarily to secondary memory. In case the kernel writes all the processes to the secondary memory, it is called a swapping system. However, if only the pages of memory are written onto the secondary memory, then it is called the paging system.
3. The kernel schedules processes for execution on the CPU. The time sharing concept allows the processes to share the CPU. When the time of a process has finished, the kernel suspends it and puts some other ready process for execution in the CPU. It is again the work of the kernel to reschedule the suspended process.
4. The kernel permits different processes to make use of the peripheral devices such as terminals, tape drives, disk drives and network devices as and when requested.
5. The kernel allocates the secondary memory for efficient storage and retrieval of user data. The kernel allocates secondary storage for user files, organizes the file system in a well-planned manner and provides security to user files from illegal access.

The services provided by the kernel are absolutely transparent to the user. For instance, the kernel formats the data present in a file for internal storage. However, it hides the internal format from user processes. Similarly, it makes a distinction between the regular file or a device but hides the distinction from user processes. Finally, the kernel provides the services so that the user level processes can support the services they must provide. For instance, the kernel provides the services that the shell requires to act as a command interpreter. Therefore, the kernel allows the *shell* to read terminal input, to create *pipes* and to redirect I/O. The computer users can also create private versions of the *shell* so that they can create an environment according to their own requirements without disturbing the other users.

8.5.3 Assumptions about Hardware

Whenever a process is executed by the user on the UNIX system, it is divided into two levels: User level and Kernel level. So, as and when a system call is executed by a process, the execution mode of the process changes from the user mode to kernel mode. The kernel tries to process the requests made

by the user. It returns an error message if the process fails. However, if no requests are given to the operating system to service, even then the operating system keeps itself busy with other operations such as handling interrupts, scheduling processes, managing memory and so on.

It is very true that the system runs in either the user mode or the kernel mode. However, the kernel runs on behalf of the user process. The kernel is not a separate process running parallel to user processes. The kernel forms a part of each user process.

8.5.4 Interrupts and Exceptions

The UNIX system allows devices such as I/O peripherals or the system clock to interrupt the CPU abruptly. Whenever the kernel receives the interrupt, it saves the current work it is doing and services the interrupt. After the interrupt is processed, the kernel resumes the interrupted work and proceeds as if nothing had happened. The hardware gives a priority weightage according to the order in which the interrupts should be handled. Thus, when the kernel looks into an interrupt, it keeps the lower priority interrupt waiting and services the higher priority interrupt.

The term exception is different from the term interrupt. An exception is a condition in which a process causes an unexpected event. For instance, dividing a number by zero, address illegal memory, etc. Exceptions occur in the middle of the execution of an instruction and are hovelled similar to interrupts. The system tries to start the instruction again after handling the exception. However, interrupts are considered to happen between the executions of two instructions. The system continues working on the next instruction after servicing the interrupt.

8.5.5 File System and Internal Structure of Files

A file in UNIX is a sequence of bytes. Different programs expect various levels of structure, but the kernel does not impose any structure on files, and no meaning is attached to its contents – the meaning of bytes depends solely on the programs that interpret the file. This is not true of just disc files but of peripherals devices as well. Magnetic tapes, mail messages, character typed on the keyboard, line printer output, data flowing in *pipes* – each of these is just a sequence of bytes as far as the system and the programs in it are concerned.

Files are organized in tree structured *directories*. Directories are themselves files that contain information on how to find other files. A path name to a file is a text string that identifies a file by specifying a path through the directory structure to the file. Syntactically, it contains of individual file name elements separated by the slash character. For example in */usr/rohit/data*, the first slash indicates the root of the directory tree, called the root directory. The next element *usr* is a sub-directory of the root, *rohit* is a sub-directory of *usr* and *data* is a file or a directory in the directory *rohit*.

The UNIX file system supports two main objects: *files* and *directories*. Directories are nothing but files that have a special format. So, let us first learn the representation of a file.

Representation of Data in a File

All the data entered by the user is kept in files. Internally, the data blocks take up most of the data that has been put in files. Each block on the disk is addressable by a number. Associated with each file in UNIX is a little table called inode which contains the table of contents to locate a file's data on disk. The table of contents consists of a set of disk block numbers. An inode maintains the attributes of a file, including the layout of its data on disk. Disk inodes consists of the following fields.

Field	Description
Mode	Specification of access permissions for the owner and other users
UID	ID of the user creating the file
GUID	Ids of user group having permissions on this file
Length in bytes	Number of bytes contained in the file
Length in blocks	Number of blocks to implement the file
Last modification date	Time the file was written to
Last access date	Time the file last read
Last inode modification	Time the file last modifeid
Reference count	Number of directories in which the file appears; th is field is used to detect when all the file refernces are deleted from all the directories so that the space may be released
Block reference	Pointer and indirect pointer to blocks in the file

The data on a file is not stored in a contiguous section of the disk. The reason behind is that the kernel will have to allocate and reserve continuous space in the file system before allowing operations that would increase the file size. For instance, let us suppose that there are three files A, B and C. Each file consists of 10 blocks of storage and suppose the system allocated storage for the three files contiguously as shown in Figure 8.4.

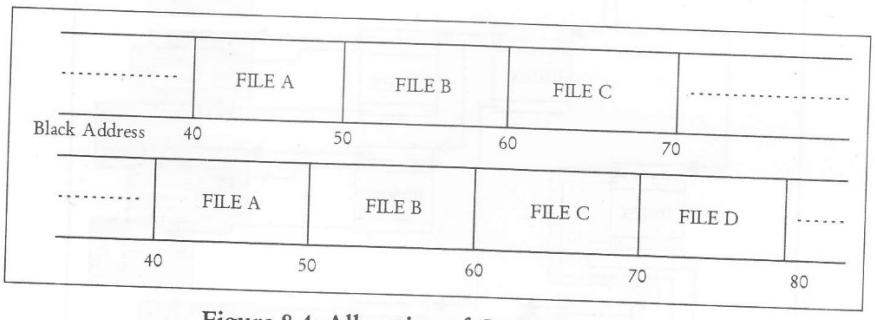

Figure 8.4: Allocation of Contiguous Files

However, if the user now wishes to add 5 blocks of data in the file B, then the kernel will have to copy the file to such a place where a file of 15 blocks can be accommodated. Moreover, the previously occupied disk block by file B's data can only be used in a case where the files have data less than 10 blocks.

The kernel allocates the file space of one block at a time. This allows the data to be spread throughout the file system. In this case, locating the data of a file becomes a complicated process. If a block contains 10K bytes, then such a file would need an index of 100 block numbers and a block of 100K bytes would need an index of 1000 block numbers. Thus, the size of the inode would keep varying according to the size of the file.

The UNIX system contains 13 entries in the inode table of contents as shown in Figure 8.5. The blocks that are marked as "direct" in the figure contain the number of disk blocks that contain the data. The "single indirect" block contains a list of direct block addresses. Thus, if the data has to be accessed through the indirect block, then the Kernel first finds out the appropriate direct block entry from the indirect block and thereafter reads the data present in the direct block. The block marked as "double indirect" contains a list of indirect block numbers and the block marked as "triple indirect "contains a list of double indirect block numbers. Let us suppose that a logical block on the file system holds 1K bytes and a block number is addressable by a 4 byte integer. In such a case, a block can hold up to 256 block numbers. The maximum number of bytes that a file can hold is found out to be 16

gigabytes. It will make use of 10 direct blocks, 1 indirect, 1 double indirect and 1 triple indirect block in the inode. Any process on a UNIX system considers a file as a stream of bytes which start at byte address 0 and continue upto the size of the file. The kernel converts the user view of bytes into a view of blocks. The file will start from the logical block 0 and goes up to the logical block number corresponding to the file size. The kernel will then access the inode. It will then change the logical file block into the appropriate disk block.

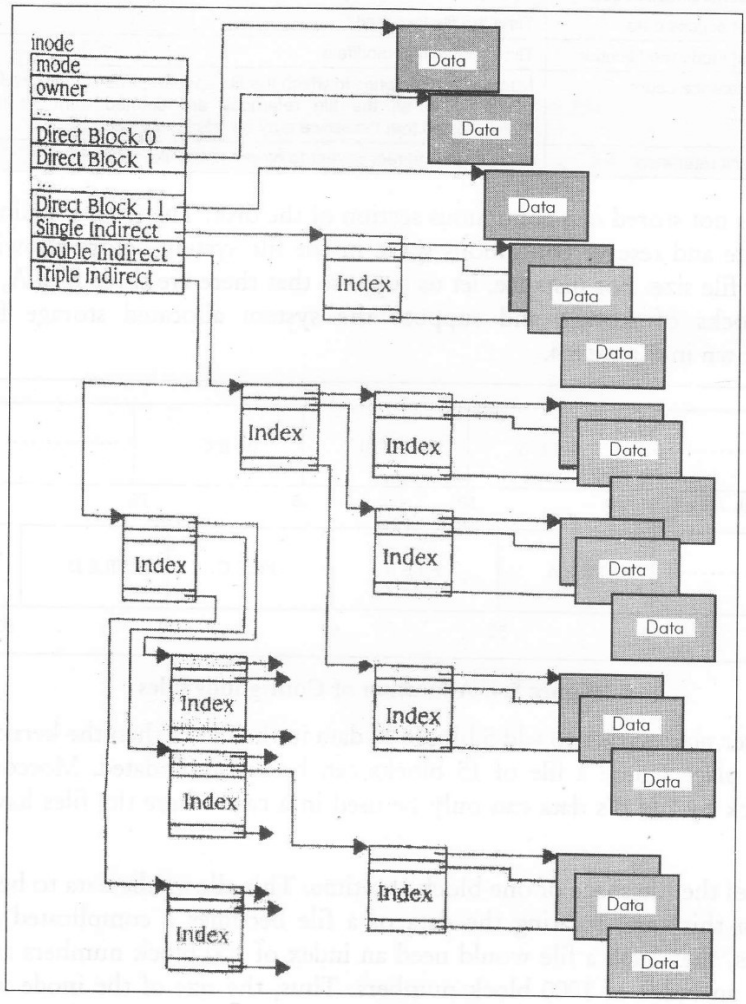

Figure 8.5: Extended-machine view of Operating System

Check Your Progress

Fill in the blanks:

1. The .....command creates a calendar of the specified month for the specified year.
2. The ..... command displays information.
3. The .....command allows the user to set or change the password.
4. The ..... command lists the users that are currently logged into the system.

8.6 LET US SUM UP

There are a few of UNIX commands, that you can type them stand alone. For example, Is, date, pwd, logout and so on. But UNIX commands generally require some additional options and/or arguments to be supplied in order to extract more information. Let us find out the basic UNIX command structure. The UNIX System offers somewhere around 64 system calls. However, only 32 system calls are frequently used. These system calls carry very simple options with them. So, it becomes easy to make use of these system calls. UNIX explicitly attempts to simplify the application programming model for files and devices by making files and devices as similar as possible. The UNIX device driver API (Application Program Interface) uses operation names similar to the file interface, although they apply to physical devices rather than to storage abstraction.

8.7 KEYWORDS

API: Application Program Interface

Banner: To display information.

Cal: To display calendar on the screen.

Date: To display and set the current system date and time.

Passwd: To install or change the password on the system.

Who: To determine the currently logged users on the system.

Finger: Gives specific information about a user.

Input/Output functions: Communicating and controlling I/O device and file system.

Miscellaneous functions: Network functions etc.

8.8 QUESTIONS FOR DISCUSSION

1. Explain five commands commonly used in UNIX with examples.
2. Differentiate between process management functions and memory management functions.
3. What is UNIX Kernel?
4. Discuss the file system and structure of UNIX.

Check Your Progress: Model Answers

1. Cal
2. Banner
3. Passwd
4. Who

8.9 SUGGESTED READINGS

Andrew S. Tanenbaum, *Modern Operating System,* Published By Prentice Hall

Silberschatz Galvin, *Operating System Concepts,* Published By Addison Wesley

Andrew M. Lister, *Fundamentals of Operating Systems,* Published By Wiley

Colin Ritchie, *Operating Systems,* Published By BPB Publications

LESSON

9

PC DOS OPERATING SYSTEM

CONTENTS

9.0 Aims and Objectives
9.1 Introduction
9.2 Command Language
9.3 System Calls
9.4 Implementation
  9.4.1 Loading of DOS
  9.4.2 Computer Files in DOS
  9.4.3 Directory Structure in DOS
9.6 Let us Sum up
9.7 Keywords
9.8 Questions for Discussion
9.8 Suggested Readings

9.0 AIMS AND OBJECTIVES

After studying this lesson, you will be able to:

- Understand PC DOS operating system
- Explain command language of PC DOS operating system
- Discuss system calls of PC DOS operating system
- Explain the implementation of PC DOS operating system

9.1 INTRODUCTION

As the name proposes, the operating System is used for operating the system or the computer. It is a set of computer programs and also known as DOS (Disk Operating System). The main functions of DOS are to manage disk files, allocate system resources according to the requirement. DOS provides features essential to control hardware devices such as Keyboard, Screen, Disk Devices, Printers, Modems and programs.

Basically, DOS is the standard through which the user and external devices attached to the system communicate with the system. DOS translate the command issued by the user in the format that is understandable by the computer and instruct computer to work accordingly. It also translates the result and any error message in the format for the user to understand.

9.2 COMMAND LANGUAGE

The substance of each of the sub-directory cannot be viewed unless it is made active, or a sub-directory is specified as part of the DIR command. Doing either of these requires an understanding of the concepts of navigating around the disk.

The directory, the user is in at any point of time, is called the Working/Present/Current directory. DOS specifies which directory you are in by displaying the directory's name in the command prompt. For example, the following command prompt indicates that you are in the DOS directory: C:\DOS>. Knowing which directory is current helps you find files, and to move from one directory to another more easily. Typically, the Root Directory (\) is the initial working directory. The complete specification of directory from root is called a PATH. By itself, the DIR command is applicable to the working/present directory. The names of the sub-directories at adjacent levels are separated by backslash (\), while specifying the path to be followed while traveling to a sub-directory.

- ***Using Path to Specify the Location of Files:*** A path is the route that guides from the root directory of a drive to the file you want to use.

  For example, to access the NOS.LET file in the LETTER subdirectory of NOS directory, DOS must go from the ROOT (\) directory through the NOS directory to the LETTER directory, as shown in the following Figure 9.1:

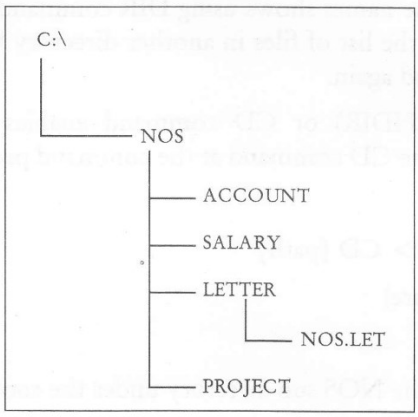

Figure 9.1

  To specify the same path at the command prompt, you would type it as shown in the following illustration:

  C:\NOS\LETTER\NOS.LET

  This is the route to the file NOS.LET. The first letter and the colon (C:) represent the drive the file is on. The first back slash (\) represents the root directory. The second backslash separates the NOS directory from the LETTER sub-directory. The third backslash separates the LETTER sub-directories from the file name, NOS.LET.

- ***Dir Command:*** The DIR command provides the list of is there on the disk that is mounted on the active drive.

  Syntax: C:\> DIR A:\> DIR

Example: A:\> DIR

Volume in drive A has no label

Directory of A:\

COMMAND	COM	23612	10-20-88	11.30a
DISKCOPY	COM	4235	10-20-88	12.00p
FORMAT	COM	15780	03-12-89	12.00p
3 files		325012	bytes free	

A:\>

As can be seen, on typing DIR followed by <Enter> key at DOS prompt, five columns of data followed by the number of files and bytes that are free in the disk are displayed. The first column contains the primary name of each file resident on the disk. However, most files are named with an extension, which appear in the second column. Whereas, the third column contains the size of the file in bytes, and the fourth and fifth columns show the date and time on which the files was created or last modified. The last line displays the number of file(s) and remaining disk space free in bytes. It is important to note that the DIR command only displays the names of the files and not their contents.

- *Changing A Directory:* All the names shows using DIR command that have <DIR> besides them are directories. You can see the list of files in another directory by changing to that directory and then using the DIR command again.

  The Change Directory (CHDIR) or CD command enables the user to travel around the directories in a disk. Type the CD command at the command prompt.

  Syntax:

  A:\> CHDIR {path} or a:\> CD {path}

  Examples: (Refer to the Figure)

  1. A:\>CD \NOS

     This command makes the NOS sub-directory under the root directory (\) active.

  2. A:\>CD \NOS\LETTERS

     The backslash indicates the root, and LETTERS, which is a sub-directory under the NOS directory, becomes the working directory.

  3. A:\> CD \

  The root directory develop into the working directory; i.e. You will change back to the root or main directory. The slash typed in this command is a backslash (\). No matter which directory you are in, this command always returns you to the root directory of a drive. The root directory does not have a name, it is simply referred to by a backslash (\).

- Making or Creating Directory: As the number of files increases in a disk, a need is felt to organize them in a meaningful way by creating sub-directories to store a group of logically related/similar files.

  To create a directory, DOS provides the MKDIR (Make Directory) or MD command.

Syntax:

A:\>MKDIR [drive:] {pathname} or A:\>MD [drive:] {pathname}

Square brackets indicate that [drive:] entry is optional.

The MD or MKDIR command creates a new empty directory whose name is the last item specified in the pathname, in the specified drive. If active, the drive need not be specified. If the directory is to be created as a sub-directory of the working directory on the active drive, typing MD {directory name} at the DOS prompt or command prompt is sufficient.

Examples:

1. A:\> MD \ACCT\SALARY

   Makes a SALARY directory in the: drive, under ACCT directory.

2. A:\> MD C:\> SALARY

   Makes a salary directory in the C: drive, under root directory.

- Deleting A Directory: You may want to delete or remove a directory to simplify your directory structure. DOS provides RD (Remove Directory) to delete a directory.

  Example:

  1. A:\> RD \ACCT\SALARY

     Removes the SALARY sub-directory in ACCT directory.

- *Copying Files:* To copy a file, DOS gives `COPY' command. When you use `copy' command, you must use the following two parameters; the location and the name of the file you want to copy, or the source; and the location and the file name to which you want to copy the file or the target (destination). You separate the source and the destination or target with a space. The syntax of the `COPY' command is

  COPY {source} {destination} or,

  COPY [drive:] [path] [filename] [drive:] [path] [filename]

  i.e. the first set of drive, path and filename refers to the source file, and the second set of drive, path and filename refers to the destination file.

- *Copying Single File:* To copy the DEBUG.EXE file from the DOS directory to the NOS directory

  1. Return to the root directory by typing the following command prompt: CD\

  2. Change to the DOS directory by typing the following commands at the DOS prompt: CD DOS

  3. To copy the file DEBUG.EXE file from the DOS directory to the NOS, directory type the following at the command prompt:

  Copy c:\dos\debug.exe c:\nos and the following message appears: 1 file (s) copied.

  Example

  A:\> copy a:\letter\office.doc \letter\office.bak

  Makes a copy of the office.doc file in the current or working directory with a new name office.bak

- ***Renaming Files:*** To rename a file, DOS offers REN command. The REN command stands for "Rename". When you use the REN command, you must contain two parameters. The first is the file you want to rename, and the second is the new name for the file. You separate the two names with a space. The REN command follows this pattern:

  REN oldname newname

  Example: REN NOS.DOC NOS.MEM

  Rename the old filename NOS.DOC to a new filename NOS.MEM.

- ***Deleting Files:*** This section give details how to delete or remove a file that is no longer required in the disk. DOS provides DEL command, which means to delete.

  Syntax: DEL {drive :} {path} {filename}

  Example:

  1. DEL \DOS\EDIT.HLP

     Delete the EDIT.HLP from the DOS directory under ROOT directory.

- **Printing A File:** The `PRINT' command of DOS works more or less like `TYPE' command, but at the same time, it allows the content of a text file to be printed on a paper.

  Syntax:

  A:\> PRINT [drive:] {path} {filename}

  Example:

  A:\> PRINT \AIAET\LETTER\AIAET.LET

9.3 SYSTEM CALLS

Some of the system calls return error information, to which there may be references in the system call description.

- ***The Program Terminate call*** reinstate terminate, Ctrl-Break, and critical error exit addresses to the values saved in the Program Segment Prefix (PSP). These are the values saved on entry to the terminating program.
- ***Keyboard Input*** examines a character from the standard input device of the calling process, writes the character to the standard output device, and then returns the character in the register. If a character is not ready to be read, Keyboard Input waits for one before returning to the calling process.
- ***Console Output*** writes the character to the calling process's standard output device. If DL contains a backspace character, Console Output moves the cursor left one position.
- ***Direct Console Input*** does not check for Ctrl-PrtSc or Ctrl-Break input from the console.
- ***Print String*** sends each character in the ASCII string addressed and terminated by a $ character to the standard output device. If the string contains a backspace character, Print String shifts the cursor left one position.
- ***Buffered Console Input*** reads characters from the calling process's standard input device and writes them to the input buffer addressed by registers.

- *Disk Reset* flushes all file buffers by writing the contents of all customized buffers to disk. The call does not update disk directory information for those files which were left open and have changed in size.
- *Open File* searches the current directory for a specified file and places information required for I/O operations in the File Control Block (FCB).
- *The Close File* call closes a file when all read or write operations have completed. You must call Close File when you have finished with the file.
- *After Search for First Entry* finds a match to an vague filename, you can call Search for Next Entry to look for the next match. An ambiguous filename is one that contains question mark characters. The Entry parameters and returned values are the same as those for Search for First Entry (11H).
- *The Delete File* call deletes all entries that match the particular filename in the current directory.

9.4 IMPLEMENTATION

9.4.1 Loading of DOS

The BOOT Record into the computer memory loads DOS.BOOT Record in turn is triggered by ROM program already there in the computer.

The system start-up routine of ROM runs a reliability test called Power On Self Test (POST) which initializes the chips and the standard equipment attached to the PC, and check whether peripherals connected to the computer are working or not. Then it tests the RAM memory. Once this process is over, the ROM bootstrap loader attempts to read the Boot record and if successful, passes the control on to it. The instructions/programs in the boot record then load the rest of the program. After the ROM boot strap loader turns the control over to boot record, the boot tries to load the DOS into the memory by reading the two hidden files IBMBIO.COM and IBMDOS.COM. If these two are found, they are loaded along with the DOS command interpreter COMMAND.COM. COMMAND.COM contains routines that interpret what is typed in through the keyboard in the DOS command mode. By comparing the input with the list of command, it acts by executing the required routines/ commands or by searching for the required routine utility and loads it into the memory.

9.4.2 Computer Files in DOS

A file may enclose a program or any other kind of information. Generally, a file must be given a name that can be used to identify it. DOS permits the user to assign a name consisting of two parts to a file - primary and secondary names. Primary name can be of a maximum of eight characters consisting of Characters, Alphabets, Number and Hyphen), and the Secondary name should consist of three characters, which is optional. The primary name and the secondary (or extension) name, if any, are to be separated by a dot (.).

Primary name can be connected to proper name, while extensions are like surnames of people. Using an extension with the file name is preferable, though optional. However, once the extension is specified, using the complete name (primary name and extension, with the period separating them can only refer the file). Using extensions can be an excellent way of naming a file so that it can be identified easily.

DOS has a way of presenting which disk drive is currently active. The floppy disk drives are allocating alphabets A and B, whereas the hard disk drive is assigned the alphabet C. If your PC has a single floppy drive, the drive would be A and if it has two, they would be termed as A and B. If your PC includes a hard disk, besides a FDD (Floppy Disk Drive), the drive names would be A and C. If the prompt is A, then it implies that the first floppy disk drive is active. Where as the DOS prompt would be C, if the hard disk is active? Data as well as instructions reside in a file stored in a disk.

9.4.3 Directory Structure in Dos

The files in the computer come from a variety of sources. Some files come with DOS, while other comes with publications such as a word processor. These files hold codes and other information that is necessary to make the computer application operational. Before long, there will be hundreds or even thousands of files in the computer, which can make it difficult to locate specific files.

The names of all the files created in a disk are stored in its directory. Directory is just like a file folder, which holds all the logically related files. DOS files are organized in a hierarchical or an inverted tree-like structure. The general analogy is with a file cabinet containing a number of drawers, which in turn may contain folders. The content of these folders is the needed information.

The file box here is the Root Directory, the drawer is Individual Directory, the folders are Subdirectory and the information in these folders may in turn be classified as Files.

If not, the large number of files that get created for various purposes in a disk can make the directory huge and difficult to view and manage. Therefore, DOS enables the user to organize the files in a disk into directories and sub-directories in a hierarchical structure. Directories can contain other directories. A directory within another directory is called a sub-directory.

Of course, there may be sub-directories of sub-directories, but a file name is the farthest you may descend down the (inverted) tree of directories and files. Thus, a file name corresponds to a tree leaf, a sub-directory to a branch, the directory to the trunk, and the root directory to the root of the tree, hence the name ROOT DIRECTORY.

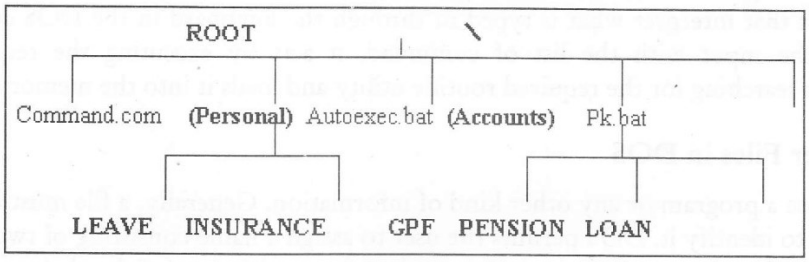

Figure 9.2: Sample of Directory Structure

Check Your Progress

Fill in the blanks:

1. To copy a file, DOS provides .....command.
2. The .....call deletes all entries that match the specified filename in the current directory.
3. The .....command gives the list of is there on the disk that is mounted on the active drive.
4. A path is the route that leads from the .....of a drive to the file you want to use.

9.5 LET US SUM UP

As the name suggests, the operating System is used for operating the system or the computer. It is a set of computer programs and also known as DOS (Disk Operating System). The main functions ofDOS are to manage disk files, allocate system resources according to the requirement. DOS provides features essential to control hardware devices such as Keyboard, Screen, Disk Devices, Printers, Modems and programs. The content of each of the sub-directory cannot be viewed unless it is made active, or a sub-directory is specified as part of the DIR command. Doing either of these requires an understanding of the concepts of navigating around the disk. Some of the system calls return error information, to which there may be references in the system call description. The BOOT Record into the computer memory loads DOS. BOOT Record in turn is triggered by ROM programalready there in the computer.

9.6 KEYWORDS

DIR command: Gives the list of is there on the disk that is mounted on the active drive.

The Program Terminate call: Restores the terminate, Ctrl-Break, and critical error.

Auxiliary Input: Reads the next byte from the standard auxiliary device and returns it in register AL.

Auxiliary Output: Writes the byte specified in register DL to the standard auxiliary device.

The Delete File: call deletes all entries that match the specified filename in the current directory.

9.7 QUESTIONS FOR DISCUSSION

1. What is the path to specify the location of files?
2. Explain five commands used in the DOS with examples.
3. What is the difference between keyboard input call and console output call?
4. Explain the directory structure in DOS.

Check your progress: Model Answers

1. COPY
2. Delete File
3. DIR
4. Root directory

9.8 SUGGESTED READINGS

Andrew S. Tanenbaum, *Modern Operating System,* Published By Prentice Hall

Silberschatz Galvin, *Operating System Concepts,* Published By Addison Wesley

Andrew M. Lister, *Fundamentals of Operating Systems,* Published By Wiley

Colin Ritchie, *Operating Systems,* Published By BPB Publications

LESSON

10

NETWARE OPERATING SYSTEM

CONTENTS

10.0 Aims and Objectives

10.1 Introduction

10.2 Communication Management

10.3 History of Netware

10.4 Netware 386 Architecture

10.5 Netware Features

10.6 Let us Sum up

10.7 Keywords

10.8 Questions for Discussion

10.9 Suggested Readings

10.0 AIMS AND OBJECTIVES

After studying this lesson, you will be able to:

- Describe communication management in Netware
- Explain the history and features of Netware
- Understand Netware 386 architecture

10.1 INTRODUCTION

The most noticeable new feature of NetWare is its internationally distributed Directory. NetWare Directory Services (NDS) is more than just a worldwide naming service. It also provides a simply managed and more secure network environment. With NDS, it is now probable to integrate a diverse network of resources into a single, easy-to-use environment. It introduces the basic concepts behind NetWare Directory Services. After giving a quick overview of NDS, it discusses NDS objects and properties and explains how these objects form the Directory tree.

10.2 COMMUNICATION MANAGEMENT

Communication between any two workstations entails a certain amount of responsibility on both sides to ensure that no information is lost. NICs maintain error checking at the bit level in the

NetWare environment. File servers and workstation shells handle packet- and session-level error checking; each preserves a table to handle this level of error checking. The NCP governs the way that the connection control information is exchanged. (It is a common misconception that SPX is used for packet level error checking between workstations and servers; however, SPX is only used for peer-to-peer interaction.) Every NCP packet submitted to a NetWare file server by a client must have a connection number and sequence number attached to it. The connection number is the number that client was assigned by the file server when the connection was established. The sequence number identifies each packet so that both the server and the shell can determine when a packet is lost.

10.3 HISTORY OF NETWARE

In the early 1990s, Novell comprehended the concept of a small single-server network with the introduction of NetWare 4. With the beginning of Novell Directory Services (NDS), Novell's new network directory, network managers could collect multiple servers into easily managed groups. Also, Novell introduced a GUI-based central management program called NetWare Administrator (NWAdmin). All the while, NetWare still provided the best file and print services for growing networks.

By the late 1990s, networks produced even larger. Networks that once were partial to a department here and there now suddenly found themselves connected to larger networks. Some networks were located within the same building; others were separated by continents and connected with WANs. And, of course, you can't forget the impact of the Internet on business.

Novell responded by releasing intraNetWare. The intraNetWare release incorporated an improved version of NetWare 4.1 called NetWare 4.11. However, the name was misleading. The new version of NetWare included with intraNetWare was more than a tenth of a point revision—it included many enhancements that made the operating system easier to install, easier to operate, faster, and more stable. It also included the first fully 32-bit client for Windows-based workstations.

As the name intraNetWare entails, NetWare 4.11 made it easier to generate intranets and link networks to the Internet. Novell bundled handy tools, such as the IPX/IP gateway, to ease the connection between IPX workstations and IP networks. And, for the first time, Novell included an application called Webserver, which lets you create and host Web sites on NetWare servers. It also began integrating Internet technologies and support through features such as natively hosted Dynamic Host Core Protocol (DHCP) and Domain Name System (DNS).

As Novell worked to recover NetWare and add potential to meet business needs, the target market continued to grow and change. Interest increased in linking large networks and attaching them to the Internet and WANs. Although NetWare was considered a strong choice for file and print services, businesses viewed the product as a weak choice for applications they wanted to deploy on their servers. End users' storage and networking needs increased as databases and applications grew to sizes once unimaginable. Administrators wanted an easy way to tie things together and make managing large and complicated networks simpler. To address these needs and apparent weaknesses, Novell introduced NetWare 5.

But NetWare 5 isn't unaided. Other new network operating systems are emerging at a rapid clip. Microsoft is releasing Windows 2000 (a.k.a. NT 5.0); Sun has released new versions of Solaris; almost everywhere you turn, you hear about the latest version of Linux; even IBM is readying a Version 5 of Warp Server. But through it all, Novell has worked to make NetWare 5 the leader of the pack.

10.4 NETWARE 386 ARCHITECTURE

Netware 386 is a big help to network computing and its adoption to MIS world. Novell makes Netware 386 developers platform of choice for distributed applications. Netware 386 architecture contains Netware loadable module which consists of extended file system, Netware file system and system executive. There are Application & Utilities and services provided to Netware loadable module

Application and Utilities

- Developer application tools
- Server Utilities
- Protocol Engine
- Disk subsystem interface

Services are further divide into

- Distributed services
- Communication services
- Message Handling services
- Database services
- File and print services

10.5 NETWARE FEATURES

This section contains the major new features in NetWare.

Novell Cluster Services "In the Box"

In an attempt to hold back the tide of information, reduce cost of ownership, decrease IT staff, and enlarge network efficiency, many businesses are pulling the storage off their distributed servers and bringing it into a central location. The clear challenge with consolidation of stored resources is the risk of failure. If a single server fails, several hundred users could be without service. If a shared disk array fails, potentially thousands of users are without service-resulting in lost production, lost revenue, and in the case of e-commerce, lost business as millions of customers head to your competition. However, the benefits of consolidation are many. Management is simplified, IT staff can be reduced or re-allocated, and, most importantly, users can be sure that their information will be available.

Storage Area NetWork (SAN) technology provides brilliant, high-performance storage options. A centralized SAN arrangement utilizes a single storage system- usually a RAID device with plenty of capacity and redundancy-to serve multiple hosts. The storage system uses a separate, back-end network for communication between storage devices and the server that connects the SAN system to the corporate network.

SAN technology radically expands the storage capability of a network, which results in significant advantages. The entire management overhead for individual storage units in the SAN falls to the SAN server, not to the distributed hosts accessing the storage system. Relieved of the storage management overhead, hosts can perform tasks more quickly.

Consistency becomes more critical than ever when host storage becomes separated from the server in a SAN configuration. A dead SAN server blocks storage files, making every host on the network useless. Network downtime due to dead storage will hurt any company, from small shops to large global Web sites.

Novell's explanation is to use a cluster of NetWare servers to control your SAN. NetWare 6 includes free cluster software for up to two systems. With additional software, you can form a cluster of up to 32 NetWare servers, each of which is automatically ready to pick up the work of any other server in the cluster. The software constantly monitors the health of the cluster. In the event of a failure, the sick node's processes are automatically forwarded to the remaining healthy nodes. This failover process eliminates any single point of failure and is transparent to your users.

Novell iFolder: Non-Stop Access to Your Files

Novell iFolder shatter the chains that, until now, have linked users to particular hardware. iFolder also eliminates location as the most important aspect of file access. New with NetWare 6, iFolder provides the technology tools to access, synchronize, and back up your files and applications anywhere and at any time.

iFolder executes data-file synchronization automatically, visibly, and securely. iFolder intelligently sends only file changes back and forth across the network, speeding performance by eliminating complete file replacements. iFolder does all this synchronization while also reducing the security hassles that are created by linking a client computer through the Internet to a remote server via Virtual Private Networking.

Three components work jointly to power Novell iFolder:

- The iFolder client permit to access to current files on a personal computer, whether or not it is connected to the network during the work session. Running on Windows 9x/ME/NT/2000, the iFolder client performs synchronization whenever connected to the network, keeping all files up to date and properly backed up. The iFolder client software also guarantees security by providing an encryption option for files stored at the central server.
- An iFolder plug-in for Web browsers offers secure authentication links to the central iFolder server. The iFolder browser addition gives users the normal file operation tools they expect (copy, delete, rename, and so on).
- The iFolder server provides the essential infrastructure for secure file synchronization and access. Using LDAP for authentication, the iFolder server runs on NetWare and comprises modules for the Apache Web server on NetWare. Encryption between the client and the iFolder server, LDAP authentication, and stored file encryption provide peace of mind for iFolder users and network managers.

Novell Internet Printing (iPrint): An End to Printing Hassles

Novell has offered shared printing services as the earliest versions of NetWare. Through the initial wave of laser printers, NetWare allowed companies to repay several thousand dollars worth of laser printer among multiple employees. As printing demands grew, so did Novell's support for printing options. NetWare led the way among all network operating systems in supporting network-attached print servers, server-controlled printing through workstation- attached printers, and remote printer control options.

Multi-Processor Enabled File System and Services

By humanizing on the Multi-Processing (MP) ability of NetWare 5.x, Novell has made NetWare 6 fully compliant with Intel's MP architecture. What this means to you is that you don't need to tie up a single processor on your server with both the NetWare OS and your applications; you can farm out your MP-complaint applications to other available processors on your MP system hardware.

Novell eDirectory

Novell eDirectory (formerly known as Novell Directory Services) is the established leader in directories. With more than 139 million users globally, eDirectory has become a domestic name by pure acceptance. The version of eDirectory which ships with NetWare 6 adds polish to an already smooth performer.

Features of eDirectory

- *Cross-Platform Support:* Novell eDirectory sustain all Internet clients, and eDirectory server software runs on all common Internet servers. Besides NetWare, which is the first and highest-performing eDirectory platform, eDirectory runs on Linux.
- *Robust Security:* With its RSA-based algorithms, eDirectory offers robust security for networks large and small. Security features include Novell International Cryptographic Infrastructure (NICI), encrypted passwords, private/public key encryption, and secure authentication services.
- *Scalability:* One of the directory group's preferred demonstrations shows eDirectory safely and securely managing over one billion objects. Will you ever need to manage one billion objects? No. Do we recommend loading a production directory with that many objects? Again, no. But it does remove any question about scalability and eDirectory's ability to manage a large corporate network.
- *Solid Partitioning and Replication:* Partitions permits eDirectory to split a large directory database into smaller portions for better client support and performance. Replication allows distributed databases to update each other quickly in the background, ensuring that changes in one portion of the directory will become available to the entire network as quickly as possible
- *Simplified Information Management:* eDirectory is intended to act as your company's data repository. It prepares a foundation for capturing, storing, and organizing a wide range of user and customer information. eDirectory can be used to model your company, regardless of the size of the company. Additionally, eDirectory can be used to leverage customer demographics, product interest, and transactions.

ConsoleOne Improvements

ConsoleOne, the Java-based management service for NetWare, persists to get better in NetWare 6. Now any client with a Java Virtual Machine (JVM) running on it can execute ConsoleOne. All eDirectory administrative functions can be performed via ConsoleOne, which uses a secure, encrypted connection. Figure 5 shows a sample screen from ConsoleOne.In addition to eDirectory functions, ConsoleOne controls many NetWare file system components, such as disks, volumes, folders, and files. You can change specific rights and attributes, and even set disk space limits for individual users. New ConsoleOne snap-ins are being created all the time to allow Novell services- from DirXML to clustering-to be administered in one convenient utility interface. With ConsoleOne, almost every administrative function can be run from a wide variety of clients.

NetWare Remote Manager for Browser-Based Management

Browser-based management, which first emerges with NetWare 5.1 as the NetWare Management Portal, permit network administrators to work from any client that is equipped with a Web browser. In NetWare 6, this popular tool has been rechristened as the NetWare Remote Manager utility. With this utility, complete server-specific information displays clearly and securely within the client browser. A monitoring page with a traffic signal icon (green, yellow, and red lights) immediately indicates the health of dozens of servers. With one click, a system administrator can select any server to see more detailed information or to make configuration changes. Encrypted password exchange via SSL keeps the management connection secure.

NetWare Remote Manager provides far more than just minor configuration changes. Server console screens that allow a network administrator to control all server-based applications may be viewed through the browser. The directory tree and established partitions are displayed, and servers may be downed, restarted, or reset through the browser.

Novell Storage Services (NSS)

Near the beginning, server operating systems dedicated most of their attention to better disk performance in the hope of minimizing the time it took to serve up requested files. Today's operating systems must perform many tasks while maintaining exceptional file service. As storage demands increased and NetWare servers began hosting multiple high-capacity disk drives, NetWare customers demanded more from their file systems. Customers wanted to control higher total disk capacities and larger single files, while mounting the file system and making disks available more quickly than ever before. Novell responded by developing Novell Storage Services, or NSS.

Open Standards

NetWare 6 supports the following unwrap standards, which are critical for customers interested in using the Internet, the World Wide Web, and any type of e-commerce applications:

- TCP/IP (Transmission Control Protocol/Internet Protocol)
- LDAP (Lightweight Directory Access Protocol)
- XML (Extensible Markup Language)
- SQL (Standard Query Language)
- ODBC (Open DataBase Connectivity)
- JDBC (Java DataBase Connectivity)
- Java Beans
- JNDI (Java Naming and Directory Interface)
- SSL (Secure Sockets Layer)
- HTTP (HyperText Transfer Protocol)
- DNS (Domain Name System)
- DHCP (Dynamic Host Configuration Protocol)
- J2EE (Java 2 Enterprise Edition)

NetWare 6's support for these standards translates to more options and better value for customers. For example, NetWare Web Access allows secure access to file, print, and e-mail via a Web browser. No longer is a VPN connection required.

Check Your Progress

Name four Netware feature.

10.6 LET US SUM UP

Communication between any two workstations requires a certain amount of responsibility on both sides to ensure that no information is lost. In the early 1990s, Novell extended the concept of a small single-server network with the introduction of NetWare 4. With the introduction of Novell Directory Services (NDS), Novell's new network directory, network managers could cluster multiple servers into easily managed groups.

Netware 386 is a big help to network computing and its adoption to MIS world. Novell makes Netware 386 developers platform of choice for distributed applications.

10.7 KEYWORDS

NDS: NetWare Directory Services

DHCP: Dynamic Host Core Protocol

DNS: Domain Name System

SAN: Storage Area Network

10.8 QUESTIONS FOR DISCUSSION

1. What is communication management in communication management?
2. What are the enhanced features of Netware?
3. Explain Netware 386 architecture.

Check Your Progress: Model Answers

Novell cluster service, Novell ifolder, Novell internet printing, Novell edirectory.

10.9 SUGGESTED READING

Andrew S. Tanenbaum, *Modern Operating System,* Published By Prentice Hall.

Silberschatz Galvin, *Operating System Concepts,* Published By Addison Wesley.

Andrew M. Lister, *Fundamentals of O* NETWARE OPERATING SYSTEM *perating Systems,* Published By Wiley.

Colin Ritchie, *Operating Systems,* Published By BPB Publications.