STUDENT GUIDE
UNIX® *QUICK START*

Table of Contents

Unit 3
UNIX File System

Unit 4
The User Interface

TOC-2

Table of Contents

Course Overview

UNIX Quick Start

Course Overview

UNIX *Quick Start* is the basic course in a complete UNIX System curriculum developed and produced by the Computer Technology Group. It was developed after testing the contents with hundreds of students in both in-house and public seminars in the U.S. and abroad. The course developers have extensive backgrounds with UNIX Systems as UNIX System curriculum designers, as UNIX System instructors, and as implementors of UNIX System applications.

UNIX *Quick Start* is modular. Thus you may take only those units of the course applicable to, or of interest to you. However, it is recommended that you go through the course units in their designed sequence:

Unit 1 - Introduction to the UNIX Operating System
Unit 2 - Getting Started
Unit 3 - The UNIX File System
Unit 4 - The User Interface
Unit 5 - Daily Survival on a UNIX System
Unit 6 - The Productive User

UNIX *Quick Start* is intended to provide you with a solid understanding of the UNIX System environment and its major features and facilities. It is not intended to give you extensive hands-on skills - these will be provided in later courses in the UNIX System curriculum. Nevertheless, actual screen examples will be demonstrated and lab exercises included throughout the course to provide you with a comfort level for later interaction with the UNIX System.

We hope you enjoy the course and we look forward to your feedback.

Unit 1

Introduction to the UNIX Operating System

Unit 1

Introduction to the UNIX Operating System

Overview

Introduction

This first unit in the UNIX Quick Start program, *Introduction to the UNIX Operating System*, will discuss the development of UNIX from a single-user system to an integrated, full-featured, multi-user system. You will be introduced to operating systems and will gain an overview of the features and benefits of the UNIX system. In addition, many of the key concepts and words used in the UNIX operating environment will be introduced. Units 2 through 6 will present greater detail and specific information to allow you to function comfortably in the UNIX operating system environment.

Key Terms

The following terms are used in this unit and the accompanying video module. Please review the definitions before viewing the video and use the listing as a reference when you are completing the program notes that follow.

application program that runs on top of the base operating system performing a specific function such as word processing or database management

batch processing
type of information processing in which the computer saves several pieces of information and then processes them at one time

Overview - continued

C language machine-independent programming language that combines the structured programming capabilities of other programming languages with the low-level capabilities of machine language

directory logical component of the UNIX file system used to "hold" files and subdirectories

file collection of related data that is maintained as a unit

input information entered into the computer by the user

interactive processing type of information processing in which the computer processes all information immediately

kernel core of the UNIX operating system that performs important functions such as control of all system hardware, task management, and data storage management

machine language hardware-dependent, low-level programming language that operates at the "bits and bytes" level that the computer can interpret

MS-DOS Microsoft Disk Operating System, a single-user operating system developed primarily for use on the IBM-PC

verview - continued

)erating system
collection of programs that controls the system hardware, runs user-requested programs, schedules events, and provides interface between user and system

S/2 multi-task operating system developed by Microsoft for the IBM environment

utput the computer's response to input

)roprietary programs written by a particular vendor solely for use on their own machines

ıser interface
program that transforms user requests into actions that are executed by the operating system

ıtility program that assists the user in performing a particular task

Objectives

At the end of this unit you will be able to:

1. Match the key UNIX operating system terms with corresponding definitions.
2. Identify key points in the evolution of the UNIX operating system.
3. Describe the UNIX operating system.
4. List 5 features of UNIX and describe the benefits of each.
5. List 5 key components of the UNIX operating system.
6. Compare the features and benefits of UNIX, DOS, and OS/2.

Introduction to the UNIX Operating System

Program Notes

To begin, a one sentence summary of UNIX is in order:

> UNIX is a portable, general purpose, operating system providing a multi-user, multi-tasking, interactive environment.

What is UNIX?

Of course, this sentence contains a collection of ideas and terms that need further explanation before we continue. The key phrases from the summary sentence are expanded upon below:

general purpose	not restricted to the type of application
multi-user	supports multiple, simultaneously active users
multi-tasking	ability to facilitate simultaneous process execution
interactive	processes information immediately rather than in bunches or batch of work
portable	easily moved from one type of hardware to another

Program Notes - continued

Operating System

The operating system is a collection of software programs which controls the hardware, runs user-requested programs, schedules events, and provides an interface between the user and the computer. The operating system coordinates the activities and functions of the computer. The type of operating system used dictates the level at which the computer hardware will be able to function. For example, some operating systems permit only one user and one task to be performed at a time. Others, like UNIX, permit many users performing many different tasks to operate at one time.

Early operating systems were developed to perform specific tasks or functions. They were often cumbersome and written in low-level machine languages. These languages converse directly with the computer at a bits and bytes level.

In addition to language enhancements, the need for improved computer capabilities was recognized. Users demanded more functions and easier to use programs. In response, programmers everywhere were trying to develop operating systems that were as responsive as the current computer technology would allow. Today, the personal computer environment rivals the early mainframe machines, and there is a demand for interactive and networking environments which will bring together multiple users operating on independent hardware platforms.

Program Notes - continued

The evolution of UNIX was a result of many people and organizations working to improve the status and capabilities of computer systems. In 1969 Ken Thompson, of Bell Laboratories, was working with an interactive operating system called "Multics", which was developed at MIT. Thompson and his associates found this system to be cumbersome and expensive to use and began to develop a more efficient and practical program. The original UNIX system was written in machine language and later was converted to "B" language, also developed by Ken Thompson.

Thompson and his associates at Bell Laboratories were not content and continued to add new technological developments to the UNIX operating system. Dennis Ritchie, another Bell employee, refined and improved the "B" language which resulted in a more machine-independent "C" language. "C" incorporates some of the best features of the machine language programs with special features of other programs such as structured programming. Machine independence means that the operating system could be transported from one set of hardware to another.

In the early 70's, a number of versions of the UNIX system were released by AT&T. These included Version-6, Programmer's Workbench (PWB UNIX), Version-7, and 32-V.

Introduction to the UNIX Operating System

Program Notes - continued

The original UNIX operating system was tested at several universities. In 1978, the University of California, Berkeley produced a version of UNIX BSD, Berkeley Software Distribution) which was used extensively in the university and engineering communities. The BSD version of UNIX, derived from AT&T UNIX 32-V, was the basis for SUN Microsystem's implementation of UNIX call SunOS.

Another popular version of UNIX, called XENIX, was developed by Microsoft in the early 80's from AT&T's Version-7 and System 3 releases. Working with Santa Cruz Operations (SCO) XENIX became an extremely popular UNIX system variant.

In 1983, AT&T released UNIX System V and 2 years later announced its System V Interface Definition known as SVID. Further improvements of System V led to releases 2, 3, and 3.2. In 1989, AT&T announced System V, release 4 which integrated features of earlier versions with the advancements found in the Berkeley, SUN, and XENIX versions.

Today the UNIX operating system is a multi-user, multi-tasking interactive system. It is widely used in academic, engineering, and business environments. The UNIX system supports a variety of applications, and many third party vendors are producing an ever increasing array of specialized application programs.

Program Notes - continued

In summary, **list the 5 major features which are responsible for UNIX' acceptance in varied computer environments:**

1. _____

2. _____

3. _____

4. _____

5. _____

UNIX is a general purpose operating system. It is not limited to a specific field or application and it can be modified for a wide-variety of applications. In addition, it can be tailored to the capabilities of individual users because both menu and command line prompts are available. Because UNIX is not proprietary, businesses can operate the system on different types of equipment located in various departments.

The multi-user feature means that UNIX has enough processing power to support multiple users. Multi-users, however, require system security features, and UNIX is equipped with an internal security structure to provide for system integrity and security.

Program Notes - continued

The ability to support multi-users interactively requires that the system be powerful enough to perform many tasks at one time. While one user is creating a file, others may be running applications programs such as spreadsheets and word processing. Another user may be printing documents. This all occurs simultaneously in the UNIX environment.

Multi-tasking also means that a single user may perform more than one task at a time. With most other systems, one task must be completed before another is begun. With UNIX, the user may, for example, print a document and run a report on a lengthy file while working on yet another application.

The interactive feature means that processing is done immediately after the user makes a request. This differs from systems which use "batch" processing in which user requests are saved and then processed at one time. The interactive feature allows users to get immediate response and also to communicate with other users currently working within the UNIX environment.

The portability feature means that UNIX can easily be moved from one type of hardware to another. This portability is achieved because UNIX uses less machine-language and more "C" language and thus can be "ported" from one hardware device to another with little additional development time. New computers can be added to an existing group with minimal adjustments to the system and users trained in UNIX can move from one type of hardware to another with little or no additional training.

Program Notes - continued

Key UNIX System Components

The above features are the result of the unique structure of the UNIX operating system. The structure depends on 5 key components:

- kernel
- user interface
- file system
- utilities
- applications

The **kernel** is the core of the UNIX system. It performs most of the important system functions. For example, it controls all of the system hardware using "device drivers" which control the terminals, printers, and disc drives. The kernel also acts as the task manager and data storage manager for the computer. System scheduling is another important function performed by the kernel. It schedules the "central processor" to ensure that the work is performed adequately for each user and thereby manages the system and guarantees efficient operation during multi-user, multi-tasking use. Additionally, the kernel enforces the system security setup and maintains logs of system activity.

The **user interface** is the software program that acts as a liaison between the kernel and the user. It interprets a user's request and brings the request to the kernel for execution. Some types of user interfaces are called shells. This important feature will be discussed in much greater detail in Unit 4, *The UNIX Interface.*

'rogram Notes - continued

. **file system** is a collection of related information
iat is maintained as a unit. In the UNIX
nvironment, the file system provides a very
tructured approach for storing data in files in
iemory. It groups related files into directories
vhich are then grouped together in a very specific
nanner. The file system structure will be discussed
n more detail in Unit 3, *UNIX File System.*

Utilities are software programs or subprograms which assist the user in performing a specific task. Utilities are also called commands. UNIX comes with over three hundred utilities built-in. Some standard utilities include communications, electronic mail, news, on-line message writing, scheduling features, programming and recompiling features, text editors and formatters, and spell checkers and dictionaries. Several of these utilities will be discussed in greater detail in other units.

Applications run on top of the base operating system. These application programs are produced by independent software developers and there are more than a thousand application programs currently available. Popular application programs include database management, communication, language compilers, accounting, and engineering.

Program Notes - continued

In summary, **list the 5 key components of the UNIX Operating System:**

1. _____
2. _____
3. _____
4. _____
5. _____

Today, there are three major operating systems available for the personal computer environment. They are MS-DOS, UNIX, and OS/2. The each have unique features and share many common features. The following is a brief look at each of these systems.

MS-DOS
UNIX
OS/2

MS-DOS also called DOS was a proprietary system produced by Microsoft for the IBM PC around 1981. DOS became the standard for personal computers due to the rapid acceptance of the IBM PC. DOS is a single-user, single task system. File sharing can be difficult. DOS has a total system software size of 600,000 bytes, a resident operating size of 25,000 to 50,000 bytes, and approximately 50 built-in utilities. As such it is of limited use for companies and organizations which require that multiple users have access to the same information.

Program Notes - continued

UNIX was developed in part to fill the void and allow multiple users to access and use the same information files. UNIX is currently used on more multi-user computer systems than any other operating system. This is due in part to its portability which allows it to be used across the range of computers - microcomputers, minicomputers, and mainframes. UNIX requires over 10 megabytes of memory and has between 100,000 and 500,000 bytes of resident operating memory. This size, plus the over 300 built-in utilities, makes UNIX the perfect choice for larger computer systems.

OS/2 is a relatively newer operating system developed by Microsoft and IBM. It is fashioned after UNIX in that both are written predominately in "C" language, both have multi-tasking capability, and both are capable of executing MS-DOS software applications. However, OS/2 is not multi-user and is not available on as many hardware platforms as UNIX.

Lab Exercises

There are no lab exercises for this unit. However, make sure that you have been assigned an account and login name on your UNIX System.

Unit 2

Getting Started

Overview

Introduction

This unit, *Getting Started*, introduces you to the key concepts required to begin using the UNIX operating system.

You will learn about the login process and will have a better understanding of the working UNIX environment. In addition, you will be introduced to commands and will gain useful information about how UNIX locates and executes a command. This information should facilitate your ability to function in a specifically defined UNIX environment.

Key Terms

The following terms are used in this unit and the accompanying video module. Please review the definitions before viewing the video and use the listing as a reference when you are completing the program notes that follow.

argument	final component of a command that defines the object the command acts upon, typically a file name
ASCII	American Standard Code of Information Interchange
boot up	to start-up the system
command	instruction to the system that causes a specific action or set of actions to take place; a command can be divided into three parts - command name, options, argument

Overview - continued

.d. (group identification number) number associated with a group name that is assigned by the System Administrator identifying the group a user belongs to

hardwired terminal connected directly to the computer system by a cable

initialization internal process which prepares the UNIX system to execute commands

login name characters used to access the UNIX system

modem device which allows a user to access the computer system via telephone lines

network communications system which provides interconnections between multiple users and one or more computer systems

option second element of a command that modifies the behavior of the command

password characters used in conjunction with the login name to ensure system security

read ahead UNIX feature that allows user to continue entering requests to the system while the system processes previously entered requests

Overview - continued

shell program that functions as a command interpreter translating user entered commands into instructions executed by the operating system

special characters
characters used in conjunction with commands that can enhance command functionality and user productivity

terminal input/output device

uid (user identification number) number assigned by the system to each user, associated with the login name

user environment
capabilities and permissions established within a user's shell that determine the scope of their functionality

Superuser System Administrator or person responsible for overseeing the activities of all users and managing, maintaining, and ensuring the security of the system

Objectives

At the end of this unit you will be able to:

1. Identify the 3 ways a terminal can be connected to a UNIX operating system.
2. Describe a typical keyboard and the command typing conventions.
3. List the steps needed to gain access to the UNIX system.
4. Describe the primary purpose and the characteristics of the login name.
5. Describe the purpose and the characteristics of the password.
6. Define the term Superuser and describe the functions of the Superuser.
7. List the 3 common UNIX shells, identify the prompt associated with each, and list characteristics they have in common.
8. List 3 basic components of the UNIX command line format.
9. Use the command to display a list of all users currently logged on.
10. Enter the command to display the date and time.
11. List the 3 basic command types.

Program Notes

Terminal-System Connections

The terminal (keyboard and screen) is the usual UNIX input/output device. It allows a user to interact with the UNIX system. There are 3 methods by which a terminal can be connected to the UNIX System. **The 3 terminal connection methods are:**

_____

_____

_____

UNIX Keyboard

Keyboards may vary depending on the type of hardware and the manufacturer. However, each keyboard used to interact with the UNIX System must have a standard set of 128 ASCII characters. The keyboard layout is similar to a typewriter with standard labeling for both upper and lower case letters of the English alphabet as well as numbers 0 through 9. In addition, there are keys with special functions such as *del* (delete) and *ctrl* (control) which will be explained later in the course.

Command Typing Conventions

When working at a UNIX terminal, the user inputs various combinations of keystrokes that are associated with a code that communicates with the computer. The keystrokes must conform to the UNIX typing conventions. Although, both upper and lower case letters are permitted for entry through the keyboard, **UNIX generally requires that commands be entered as:**

_____

Program Notes - continued

Special Characters

In UNIX, some predetermined tasks are performed by pressing one key or a combination of keys. These specialized keys have a specific meaning in UNIX. **The keystrokes associated with these specific tasks are called:**

_____

Note: A list of special characters for the UNIX operating system is included at the end of the Program Notes.

Boot-up and Login

The process of starting-up the computer is given a special jargon term. **The start-up process is called:**

_____

After booting up the computer, a symbol called a "prompt" will appear on the terminal screen. This prompt indicates that the system is ready for the user to request access. **The term given to accessing the system by entering an assigned code name is:**

_____

To login, the UNIX system usually requires that you enter two pieces of information assigned to you by the System Administrator. **To login, you need a:**

_____

_____

Program Notes - continued

The login name and password are assigned by the System Administrator. At the first prompt, the login name is entered. Then, it is common for another prompt to appear. This is the password prompt. The user now must enter his/her previously assigned password name. After the password entry is accepted, the user and UNIX system should be ready for work.

In summary, **the steps needed to gain access to the UNIX system are:**

_____

_____

_____

The login procedure can require additional steps depending upon the manner by which the terminal is connected to the UNIX system. If the terminal is hardwired, the login prompt should appear immediately. If the login prompt does not appear after booting up the system, simply press the "return" key a few times or hold down the "control" key and press "d".

If the terminal is connected through a modem or a local area network (LAN), additional login steps are required. Modem users must first dial the system's telephone number before the login prompt will appear on the screen. LAN users may be required to sign-on to the network and identify the name of the system they want to access before the login prompt will appear.

Program Notes - continued

Login Name Characteristics

Your login name uniquely identifies your presence on the system. It is established by the System Administrator (Superuser) and is used by the system to establish a form of system security by limiting users access to only selected utilities. **What is the primary purpose of a login name:**

_____

The login name may be a word, a set of letters, a set of numbers, or a combination of letters and numbers. However, there are limits to the number of characters that can be used to make-up a login name.

What are the minimum number and the maximum number of characters that can be used for a login name?

minimum number: ____

maximum number: ____

The UNIX system actually sees your login name as a number called the "User Identification Number " or uid. This number can be found in the password file. When you login, the computer verifies the correctness of the name by comparing it to the uid. This verification process is transparent to the user. Once the login name has been verified, the computer then may request a password from the user.

Program Notes - continued

Password

The use of passwords may be optional on your system, but its use is highly recommended in order to maintain the most basic form of system security. Like the login name, a password can be a combination of letters, numbers, or other characters. The basic rules for forming a password are:

1. A password must have at least 6 characters.
2. Only the first 8 characters are significant (any characters after the 8 are ignored).
3. A password cannot resemble its associated login name.
4. New passwords must differ from their old version by at least 3 characters.

System Administrator

What is the term sometimes used when referring to the System Administrator?

_____

The Superuser alone has the power to perform very specific duties. **What are some of the Superuser's functions?**

_____

_____

_____

Program Notes - continued

The Superuser is able to bypass file access permissions and other restrictions that are denied the ordinary user. He/she must take special care to ensure that each entry in the password file contains specific information about the user. This is done by issuing a series of commands which follow a very precise format so that the UNIX system integrity will be maintained.

Once in the UNIX system, a user must issue commands to the computer in order to make it respond in an appropriate manner. Every user must follow a prescribed format so that the computer can correctly interpret and respond to the command.

An Aside: Line Entry Fields

When the Superuser adds a new user to the system care must be taken to ensure that each entry in the password file contains specific information about the user. Each entry will consist of a line containing a set of fields. Each field is separated by a colon.

Field 1:	user's login name
Field 2:	encrypted version of the user's password
Field 3:	user identification number (uid)
Field 4:	group identification number (gid)
Field 5:	comment field
Field 6:	user's login or home directory
Field 7:	program name to be executed

Program Notes - continued

The group identification number identifies a user as belonging to a particular group. The user id and the group id are used in conjunction with file permissions.

The comment field is the only optional field in the password file. It can be left blank or specific data can be input, for example, the user's actual name.

Using the information for the fields described above, **create a representative password file line entry:**

_____

User Environment

Some users can be restricted to performing limited functions within the UNIX environment by the Superuser. For example, a user can be restricted to working on one specific program. Each time that a user executes a successful login procedure, one and only one program would be available to the user. When the program terminates, the user is logged out. In other words, the Superuser assigns each user with the ability to access very specific parts of the UNIX System.

Altogether, the specific parts of the UNIX operating system made available to each user, such as the home directory and shell program, define the user's working arena. **What is the working arena called?**

_____

Program Notes - continued

Common UNIX Shells

Shells are used for user interface with the UNIX System. There are 3 common shells for the UNIX operating system. Each has unique characteristics and some features in common with the others. These characteristics and features will be discussed in detailed in Unit 4, *The User Interface*.

The common UNIX shells are:

1. _____
2. _____
3. _____

Five features the UNIX shells have in common are:

1. _____
2. _____
3. _____
4. _____
5. _____

Regardless of which shell is used, all commands are issued at the shell prompt. When a shell is ready to accept a command, a prompt is displayed.

Program Notes - continued

What prompt is displayed for each of the 3 common UNIX shells:

Bourne Shell _____

C Shell _____

Korn Shell _____

Command Line Format

All commands are issued at the shell prompt. Each UNIX line has a common format consisting of 3 basic components. **The 3 UNIX line format components are:**

1. _____
2. _____
3. _____

The command name identifies the command to be executed. After the command is entered, a command option can be specified to modify the behavior of the command. It is issued after the command name as a number or letter and is usually preceded by a hyphen (-) or minus sign. A command argument is typically the name of a file.

For example, when using the list (**ls**) command which requires both options and arguments, the following results occur:

Program Notes - continued

Command: **ls**
User is presented with a list of all the file names in the current directory.

Command with Option: **ls -**
User is presented with a long list of all the files in the current directory, including data on file size, file permissions, and the login name of the file creator.

Command with Argument: **ls file 1**
User is presented with one name, file 1.

Command with Option and Argument:
ls -l file 1
User is presented with long comprehensive list of information about file 1.

Regardless of whether an option or argument is used with a command, the rule for entering these items on the terminal is that command names, options, and arguments be separated from each other by a space.

The **date** command and the **who** command are two simple commands which display basic status information.

***date* and *who* Commands**

Program Notes - continued

What status information does the date command furnish?

_____

_____

What status information does the who command furnish?

_____

_____

_____

The **who** command, when used in conjunction with the **-u** option, provides two columns of information: when the last activity occurred at each terminal and the process identification number (p.i.d.). The **-H** option provides column headings above the output of a **who** command, and when combined with the **-u** option the user receives additional information along with column headings. The **who am i** command displays information about the user's own terminal.

When a command is entered, the system evaluates the entry, locates the command, and executes it. The shell actually locates the command the user entered by referencing internal tables and variables. If the command exists as an executable file, a new

UNIX Command Process

Program Notes - continued

process is created and the file is executed. If the command is a built-in or a function, the command is executed as a sub-routine of the shell and no new process is created. Unlike an executable file, when the built-in or function completes its process the shell does not terminate.

What are the 3 steps performed by the system when a command entry is made?

1. _____

2. _____

3. _____

Program Notes - continued

Special Characters

Control Characters (press and hold down the "control" key along with another key)

ctrl-d	End of File
ctrl-s	Suspend Output
ctrl-q	Resume Output
Del	Interrupt
Break	Interrupt

File Name Generation (FNG) Metacharacters

* . ? []

(File Name Generation characters are discussed in Unit 4.)

Lab Exercises

1. Connect to your UNIX system and raise a login prompt.

2. Login.

3. Enter the **date** command.

4. Enter the **who** command to list all logged-on users.

5. Enter the **who** command with the **-u** option

6. Enter the **who** command with the **-H** option.

7. Enter the following commands:

   who -u -H
   who -H -u
   who -uH
   who -Hu

   Notice that regardless of how the options are specified, the output is the same.

8. Enter the command **who am i**

9. Logoff.

Unit 3

UNIX File System

verview

Introduction

is unit takes a close look at the *UNIX File System* ucture and its components. You will learn about s, directories, special files, and the naming nventions for each. In addition, you will be posed to multiple file systems supported by ysical disk drives. The final topic in this unit will a close look at several directory commands which e used to organize and use a directory structure in e UNIX environment.

Key Terms

ie following terms are used in this unit and the companying video module. Please review the finitions before viewing the video and use the sting as a reference when you are completing the rogram notes that follow.

ull path name
directions that start at the root directory and lead through a unique sequence of directions to a desired location (sometimes called absolute pathname)

ierarchy organization pattern in which directories and files are subordinate to one another; the relationship allows for many layers of files and directories

iome directory
directory assigned to a unique user; cannot be added to or deleted by another user without permission

Overview - continued

naming convention
combination of symbols and alphanumerics that can or cannot be used to name directories or files

ordinary files
collection of file characters stored on a disk

path name
unique name which shows the location of a file or directory and provides directions for reaching it

relative path name
directions that start in the current working directory and lead up or down through a series of directories to a particular file

root directory
source directory which can be used to reach other major file system directories

special file
file representing a physical device such as a terminal, disk drive, magnetic tape drive, or communication link

Objectives

At the end of this unit you will be able to:

1. Describe the organization of the UNIX file system.
2. List and describe the contents of the 3 types of files in the file system.
3. Describe a "root" file system, also known as a "hierarchical" file system.
4. Describe the term home directory.
5. Define pathname and explain the functions of both full and relative pathnames.
6. List the rules for naming directories and files.
7. List and describe the 4 UNIX commands used to organize and use a directory structure.
8. Discuss the importance of file protection and the symbols used in the file notation to assign or restrict permission to use a file.
9. Define the following commands: **cat, file, more.**

Program Notes

System Organization

The UNIX file system provides a logical method for managing, storing, and retrieving information. Its organization is hierarchical and resembles an organization chart or an inverted tree.

Types of Files

The UNIX system consists of three file types. **What are the 3 file types in a UNIX system?**

1. _____
2. _____
3. _____

An ordinary file is a collection of characters stored on a disk. Ordinary files consist of either text or program code. Once created, a file can have information added or deleted and it may be deleted from the system when no longer needed.

A directory is a collection of files or directories or sub-directories. Directories are used to organize groups of files. For example, a directory could contain a file on every salesperson in a given territory.

Special files typically represent a physical device such as a terminal, disk drive, magnetic tape drive, or communication link. The system reads and writes to special files in the same way it does to ordinary files. However, the system's read and write requests do not activate the normal file access mechanism. Instead, they activate the device handler associated with the device.

Program Notes - continued

Match the file type with the definition:

ordinary file ___

special file ___

directory ___

A. physical devices (e.g. disk drive, terminal, tape drive, or communication link)

B. collection of characters stored on disk

C. collection of files or subdirectories

Unlike other operating systems, UNIX views all files alike. This makes the UNIX system file structure easy to use. As we stated above, the files are organized into a hierarchical structure resembling an organization chart or an inverted tree shape.

Root Directory

The root directory is the source with branches for other directories, subdirectories, and files. In this structure, we have what could be called a parent/child relationship, as in a family tree.

All parts of the file system may be reached by moving through the branches which extend from the root or source. This allows a user to easily access all of the directories and files in the file system. This permits many layers of files and directories. In fact, there is no limit to the number or layers of files and directories that could be created.

Program Notes - continued

An exercise to reinforce the hierarchical nature of the file system, refer to the chart below. **Use the following information to label the boxes and complete the file structure.**

label the primary source Root Directory

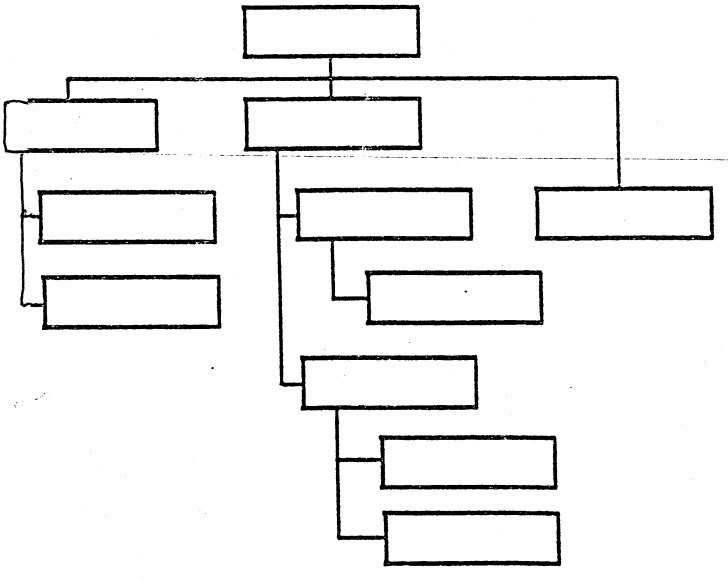
label the two directories at the second-level Directory A, Directory B, and a file: file c

Directory A has two files: File a1, File a2

Directory B has two branches: Subdirectory B1, Subdirectory B2

Subdirectory B1 contains a file: file B1-1

Subdirectory B2 contains two files: file B2-1, file B2-2



Program Notes - continued

Home Directory

the completion of the login procedure, the user is
.ced at a specific point in the file system structure,
led the login or home directory. Each legitimate
er is assigned a unique home directory in the file
stem. It is your directory and no other user can
ıd or write files in your home directory without
rmission.

om your home directory, you can move through
e system hierarchy to work in any directory or file
access other user's files with permission.

s a summary, **complete the following statements
)out the home directory:**

The home directory is also called the _____ directory and is assigned to _____ user. No one can access your home directory without your permission and therefore information cannot be _____ or _____ by another user.

Path Name

:ach directory, subdirectory, and file in the UNIX file
ystem is identified by a unique path name which
hows the location of the file or directory and
)rovides directions for reaching it. There are two
ypes of path names - full and relative.

:ogram Notes - continued

scribe the concept of a path name:

_____

_____

_____

Full Path Name

ull path name, also called absolute path name, es directions that start at the root directory and d down through a sequence of directories to a rticular directory or file. The full path name ways starts at the root of the file system and its ding character is always a slash(/). The final me in a full path name is either a file name or a rectory. All other names in a path must be rectories.

r example, imagine that you are working in the ***rplane*** subdirectory located in home directory ***ser1.*** The full path name of your working rectory is:

/user1/airplane

here the slash (/) that appears as the first aracter in the path name is the root of the file stem; ***user*** represents the system directory one vel below the root directory in the hierarchy. The cond slash delimits the directory names ***user*** and ***irplane***, where ***airplane*** is the working directory.

ogram Notes - continued

Relative Path Name

lative path name gives directions that start at working directory and lead up or down through a es of directories to a particular file or directory. ing down from the current directory accesses and directories owned by the user. Moving up n the current directory accesses layers of parent ctories and ends at the root directory. From the :, the user may move anywhere in the file system.

elative pathname begins with one of the following, irectory name; a dot (.) which represents the rent directory; or a dot dot (..) which represents directory immediately above the current ectory (called the parent directory) in the file tem hierarchy.

example, if the user is currently in the directory *plane* of the above described sample file system, d ***airplane*** contains directories named ***draft***, ***ters***, and ***bin*** and a file named ***mbox***., the relative h name to any of these is simply its name, such ***draft*** or ***mbox***.

Naming Conventions

ming conventions within the UNIX system must low certain rules. First the name of a directory or e can be from 1 to 14 characters long. These aracters may be a combination of alphanumeric d underscores. All characters except / are legal, t the following should be avoided:

SPACE, TAB, BACKSPACE and

? @ # $ ^ & * ( ) [ ] \ ¦ ; ' " `

ogram Notes - continued

, avoid using +, -, or . as the first character in a
ıame. Remember that upper and lower case
·acters are distinct to the UNIX System. The
em considers a directory or file named "draft" to
ifferent from one named "DRAFT".

summary, **complete the following rules for
ıing directories and files:**

The name of the directory or file can have between _____ and _____ characters.

The characters may be any combination of

_____

Although some characters should be avoided to reduce confusing them with system commands and directions, all characters are legal *except* _____.

Directory Structure Commands

re are 4 basic UNIX commands that enable a
r to organize and use a directory structure. They
make directory, list, change directory, and
ıove directory. **What are the UNIX notations for
se basic commands:**

make directory _____

change directory _____

remove directory _____

list _____

rogram Notes - continued

ese commands may be used with full or relative
th names, and two of the commands, **ls** and **cd**,
n be used without a path name.

kdir enables the user to make new directories and subdirectories within the current directory. To create a directory, enter the command name followed by the name you are giving the new directory.

lists the names of all the subdirectories and files in a directory. If a directory is not specified, the **ls** command lists the names of files and directories in the current directory. A useful option under this command is **ls-l** which displays the contents of a directory in long format giving mode, number of links, owner, group, size in bytes, and time of last modification for each file. In addition, the **pwd** command is used to display the path name of the current directory.

d enables the user to change location in the file structure from one directory to another. To use the cd command, enter **cd** followed by a path name to the directory to which you want to move. If you do not specify a path name, the command will return you to the home directory. Once you have moved to a new directory, it becomes your current directory.

mdir enables the user to remove an empty directory.

)irectories and subdirectories when used in a logical
nd meaningful scheme will facilitate the user's
bility to retrieve information from the file structure.
f all files pertaining to one subject are put together
n a directory, the user will know where to find them.

rogram Notes - continued

File Protection

e protection is a function of the change mode
nmand (**chmod**). **chmod** allows users to decide
o can read, write, and execute files. Users can
ow path names to various directories and read
d use files belonging to one another only if they
ve permission to do so. The file's creator identifies
o will have access to the file and can also restrict
missions for directories with the change mode
nmand, **chmod**.

e following three symbols are used to assign
rmission:

	allows system users to read a file or copy its contents
w	allows system users to write changes into a file or copy it
x	allows system users to run an executable file.

ıe following three symbols are used to specify
hich users will be granted or denied permission:

u	represents the owner of the file and is short for user
g	represents the members of group, and could consist of examples of project team members or members of a department
o	represents all other users

'rogram Notes - continued

'hen a file or a directory is created, the system ıtomatically grants or denies permission to use to ıembers of the user's group or other system users. his automatic function may be altered by the wner.

or example, to change the permission of the files nemo" so that all system users could read the file, nter the command:

chmod o+r memo

iles may be kept private and reserved for the reator's exclusive use; permission may be granted) read and write changes to files by group members nd other users; or permission to execute the rogram may be granted to group members or other ısers.

hree additional commands needed to work in the JNIX file structure are: **file**, **cat**, and **more**.

Commands: file, cat, more

he **file** command followed by a file name is used to dentify the type of file specified.

he **cat** command displays the contents of a file or les.

he **more** command lets the user examine the ontents of a text file. However, the **more** command vill display the contents of a file or files a screen full t a time.

Lab Exercises

1. Login.

2. Change your current working directory to your "home" directory. Issue the command that will show you the full path name of your "home" directory.

3. List the names of ALL your files in your "home" directory.

4. How many files in your "home" directory are directories?

5. Do you have a file named ".profile" or ".login".

6. Use the **file** command to determine what type of data is in your ".profile" or ".login" file.

7. Use the **cat** and **more** commands to display the contents of your ".profile" or ".login" files.

8. Make a directory under your home directory called "mydir".

9. Change the mode of the directory "mydir" so it is readable by all other users and verify using the **ls-l** command.

Lab Exercises - continued

10. Change directory to "mydir" and verify that you are in that directory.

11. Return to your "home" directory.

12. Remove "mydir".

13. Logoff.

Unit 4

The User Interface

Overview

Introduction

This unit, *The User Interface*, will examine the most important user interface called a shell. As the UNIX system developed so did the shell concept, and today there are three commonly used shells available. They are the Bourne Shell, the C Shell, and the Korn Shell. Each of these will be discussed with an emphasis on those features which they have in common. In addition, this unit will present information on a new generation of user interfaces, the user agent, which feature a stylized graphic displays using icons, windows, and pull-down menus.

Key Terms

The following terms are used in this unit and the accompanying video module. Please review the definitions before viewing the video and use the listing as a reference when you are completing the program notes that follow.

command alias
shell's ability to create a macro

command history
shell's ability to keep a log of all commands entered during a session

GUI (graphical user interface)
type of User Agent that relies on icons to represent system commands, devices, and utilities

Overview - continued

I/O redirection
method to change the source of standard input and the destination of standard output

macro technique that provides a shorthand method of executing commands

metacharacter
one of a set of characters that can be used as shorthand notation when referencing file names, sometimes referred to as file name generation characters

pipeline shell feature that permits the user to connect the standard output of one command to the standard input of the following command

shellscript simple shell program consisting of a single UNIX command or multiple commands stored in an executable file

user agent type of user interface that uses full-screen or window-oriented menus and/or graphics to provide assistance in using the operating system

Objectives

At the end of this unit you will be able to:

1. List the functions of a user interface.
2. Define the term shell.
3. List the limitations of a restricted shell.
4. Compare and contrast the features of the Bourne, C, and Korn shells.
5. Describe the use of the I/O Redirection feature.
6. Describe the function of Pipelines.
7. Describe the functions of the 3 Metacharacters.
8. Define Shellscript and give an example of its use.
9. Define Variable Assignment.
10. Give 2 examples of Conditional Constructions.
11. Define Command Substitution.
12. Describe the purpose of a User Agent.
13. Describe the term Graphical User Interface.

Program Notes

User Interface

The user interface is a software program that transforms user requests into actions that can be executed by the operating system. User interfaces are called up after the login procedure is successfully completed. The most common forms of user interfaces are called shells.

What is the purpose of a User Interface?

_____

_____

Shells

A shell functions as a liaison between you and the UNIX Kernel. It consists of a powerful programming language which provides conditional execution and control flow features and provides a unique command interpreter that allows communications with the operating system. A shell is considered to be interactive because it accepts user input and responds appropriately.

Complete the following thought about a UNIX shell:

A shell is a program that assists the user ...

_____

_____

Program Notes - continued

Restricted Shell

One special type of shell, known as a restricted shell, limits user access to certain operations. When a restricted shell is used, the user can not operate in any directory other than the login directory. Nor can the user add or remove directories, gain access to any program outside of the current directory and path, or use output redirection.

Based on the limitations imposed on the restricted shell, **what is a major use of the restricted shell in the operating system?**

_____

_____

Bourne Shell
C Shell
Korn Shell

The UNIX system is not limited to one shell, and today there are three very popular shells that are commonly used. They are the:

Bourne Shell
C Shell
Korn Shell

The Bourne shell has long been considered to be the standard UNIX shell along with the C shell which is the standard shell for BSD versions of UNIX. The C and Korn Shells were developed as enhancements to the Bourne Shell.

Program Notes - continued

The common features of the Bourne, C, and Korn Shells were introduced in Unit 2 and will be discussed in greater detail later in this unit. The common features are:

- I/O redirection
- pipelines
- metacharacters
- variable assignment
- conditional operators (constructions)
- command substitution.

Again, as we discussed in Unit 2, each shell displays a unique prompt when it is ready to accept a command input. These prompts are:

Bourne shell	$	(dollar sign)
C shell	%	(percent sign)
Korn shell	$	(dollar sign)

With each of these shells, once a command is entered, the shell processes the command and then redisplays the prompt when it is ready for a new command. The interpreted programming language of the shells has many capabilities including variable assignment and referencing; conditional constructions for branching and looping; and command substitutions.

Program Notes - continued

When a user logs-in to the system, one or more shell programs are executed to establish the login or user's environment. This environment can be modified by the Superuser to suit your particular needs. The programs and files that constitute your user's environment are located in your Home Directory. Each of these three shells use one or more startup files which contain information that modifies your environment when you login. The Bourne and Korn shells use a file called **.profile** while the C Shell uses 2 files called **.cshrc** and **.login**. In addition, the C Shell also references a file called **.logout** that is executed when you logoff.

The C and Korn Shells have features not provided by the Bourne Shell. Both have a history function and an alias feature.

Command History

The command history refers to the shell's ability to log all the commands entered during a session and enables the user to recall and re-execute a previous command. This feature is very valuable when a user is debugging a complex pipeline or shellscript.

Common Alias

A command alias refers to the shell's ability to implement a shorthand method of executing commands. The command name is defined to the shell with a special alias operator and includes the definition of a real command to substitute when the alias is seen in the command line. The command alias allows you to customize sessions and commands to fit your specific needs.

Program Notes - continued

The Korn Shell, unlike the Bourne and C Shells has, in addition to the above features, an array of built-in integer arithmetic functions.

I/O Redirection

A shell expects that all input comes through the keyboard (standard input) and the user expects that all output is displayed on the terminal screen (standard output). Each of the shells allows the user to change the source of the input and the destination of the output. This is called input/output **(I/O) redirection**.

Input Redirection	<	less than symbol
Output Redirection	>	greater than symbol

Example: Input Redirection

mail user1 < file1

Entering the **mail** command followed by a user's login name, the less than symbol, and a file name permits the user to send the contents of a file to the terminal screen of the identified other user.

Example: Output Redirection

ls > file1

Program Notes - continued

Entering the command **ls** followed by a greater than symbol and a file name permits the user to capture the output of the list command in the named file.

It should be noted that to avoid overwriting existing data within a file while redirecting output, you should enter 2 greater than (>>) symbols. This will preserve current data and place the new data at the end of a file. Command redirection is used only when going from a command to a file.

Pipelines

Pipelines permit the user to effectively create new commands by "joining" existing UNIX commands together. The function of a pipeline is to allow you to use the output of one command to be the input to another command. Remember that a pipe can only be used when redirecting the output of a command to another command, not a file.

What is the symbol used for the pipeline command?

_____

The number of commands that can be connected in a pipeline are virtually unrestricted. This feature expands the number and variety of functions that you can perform.

Program Notes - continued

Metacharacters

Metacharacters (also referred to as File Name Generation or FNG characters) are used in a command line as shorthand notation to specify a file or group of files rather than naming each individual file. There are 3 basic metacharacters and each is a unique character.

Match the metacharacter symbol below with its corresponding function:

[ ] ___

* ___

? ___

! ___

A. matches any sequence of zero or more characters in a file name

B. matches any single character in a filename

C. matches files that have a range of characters or a certain "class" of characters

D. matches a range exception

Shellscript

A **shellscript** or **shell program** is nothing more than a file containing UNIX and/or shell commands. A user can then enter the name of this file and the shell will sequentially execute the commands in it. Shellscripts can be particularly helpful if you have a series of commands that you use repetitively. By

Program Notes - continued

storing these commands in an executable file, you can run the whole series of commands by simply entering the name of the shellscript file.

Variable Assignment

Variable assignment is a feature which can store a value that can be retrieved as needed. You assign a variable by entering the variable name followed by an equal sign followed by the value. For example

vary = 8

will assign the string "8" as the value of the variable "vary". This technique applies to both the Bourne and Korn Shells. For the C Shell, variable assignment is accomplished by using the **set** command.

So, to perform the same asssignment in the C Shell, enter:

set vary = 8

For all of these shells, to subsequently display the contents of the assigned variable, "vary", enter the command:

echo $vary

You can reference the contents of a variable by specifying a dollar sign followed by the variable name. For example

$vary

Program Notes - continued

Conditional constructions are used for branching and looping. The branching construct is used when a decision is required concerning which of two sets of alternative instructions are to be implemented. A looping construct is used when a set of instructions must be performed repeatedly.

Conditional Constructions

Command substitution allows the user to substitute a reference to a command with its output.

Command Substitution

A **user agent** is a type of interface which relies on menus and is considered to be user friendly. It is designed to make the system administration tasks easier or to function as a front end for a turnkey application package. User agents are purchased from software vendors and are not considered to be part of the UNIX system.

User Agent

Graphical user interfaces (GUI) are user agents. They are highly stylized graphic displays that use icons to represent system commands, devices, and utilities. GUIs require a high resolution terminal screen and a mouse used as a pointing device to manipulate images on the terminal screen.

GUI

Lab Exercises

1. Logon

2. Enter the command: **date > dfile.1**

3. Enter the command: **who . wfile.1**

4. Enter the command: **ls -l . lfile.1**

5. Use the **cat** command to display the contents of the files dfile.1, wfile.1, and lfile.1

6. Repeat steps 2 through 4, creating the following files:

   dfile.2
   dfile.3
   wfile.2
   wfile.3
   lfile.2
   lfile.3

7. Enter the **ls** command that lists only the files that end with ".2".

8. Enter the **ls** command that list only the files that begin with "wfile".

9. Enter the **ls** command that lists all files ending with ".1" or ".3".

Lab Exercises - continued

10. Enter the command: **date >> dfile.1**
and examine the content of the "dfile.1" file.

11. Repeat step 10, what is happening to the file?

12. Enter the command: **date > dfile.1**
and rexamine the content of the file. What happened?

13. Your prompt is stored in a variable called "PSI". Use the **echo** command to display its current value.

14. Change the value of your PSI variable to: "READY>".

15. In this exercise, we will combine the use of variable with the technique of command substitution:

    a. Change directory to /usr/lib.

    b. Enter **pwd** to verify your location.

    c. Enter the command: **a='pwd'** or
    set **a='pwd'** (for C Shell users)

    d. Enter the command: **echo $a**

    e. Return to your home directory.

Lab Exercises - continued

f. Enter the command: **cd $a**

g. What is your present location?

16. Logoff.

Unit 5

aily Survival on a UNIX System

Daily Survival on a UNIX System

Overview

Introduction

This unit, *Daily Survival on a UNIX System*, will discuss the everyday user environment including key information which will help you to effectively manipulate the system to produce your desired results and be aware of daily changes and upgrades to the environment. The role of the System Administrator or Superuser in the management of current activities on the system is vital. You will learn how to find out what activities the System Administrator is performing, how to communicate with other users of the system, and how to print files. This information will facilitate your ability to function in the UNIX environment.

Key Terms

The following terms are used in this unit and the accompanying video module. Please review the definitions before viewing the video and use the listing as a reference when you are completing the program notes that follow.

calendar program for organizing and managing appointments and other items of interest

cpio (copy in/out) utility used to backup and recover files

mail command for sending messages (electronic mail) to one or more users

motd (Message of the Day) information file maintained by the System Administrator

Overview - continued

.news_time file located in your home directory used to track which news items have been read

request id number assigned by the system to uniquely identify print jobs

/usr/news (User News Directory)
files of news items which the System Administrator believes are important to users

write command that allows interactive communication between two active users via their terminals

Objectives

At the end of this unit you will be able to:

1. List the function of the following:
   motd File
   /usr/news Directory
   .news_time File
   calendar Program
2. List the names of the utilities used by the System Administrator to perform administrative functions.
3. Use the mail command to communicate with other users.
4. Use the write command to communicate with other users.
5. Use the 3 commands needed to print files.

Program Notes

As we discussed in Unit 2, *Getting Started*, the System Administrator or Superuser is responsible for operation and maintenance of the system. This unit will examine the responsibilities of the Superuser in more detail. If you need to review the role of the Superuser, please review your program notes from Unit 2 at this time.

The System Administrator uses the id "root" to log into the system. This gives him/her unrestricted file access and permits the Administrator to examine all files and directories in the UNIX file system.

System Information Management

To perform system administration duties, certain files, directories, and utilities have been established. They are generally maintained by the System Administrator and used by individuals to receive and send system status information. These include:

motd (message of the day)
This file is maintained by the System Administrator. You should read it each time you log into the UNIX system. This file will notify you of timely information including scheduled shut downs of the system.

/usr/news (User News Directory)
This directory consists of files with messages from the System Administrator and you may read them at your convenience. The **news** command will access the files in the **/usr/news** directory.

Program Notes - continued

.news_time (News Time)
This file located in your home directory keeps a record of news items you have read. Each time the news command is invoked, the file is updated so that only new and unread news items are displayed. To reread an old news item, use the **mail** command with the **-a** option (**mail -a**).

calendar The calendar program is a UNIX utility that allows you to implement your own personal reminder service. This utility references a file also called "calendar" in the current directory which contains dates and reminder messages. When executed the **calendar** program will print out all lines from the "calendar" file that contain today's or tomorrow's date in them.

Note: Information on creating your own reminder service using the **calendar** program is found at the end of the Program Notes section of Unit 5.

System Administration Utilities

In addition to general system maintenance, the System Administrator is responsible for monitoring the commands used on the system and terminating any processes that excessively degrade system performance.

Within a multi-user environment several functions must be performed on a periodic basis to ensure smooth and continued system operations. To do this the UNIX system includes utilities designed to support these functions. They include:

Program Notes - continued

Accounting These utilities permit analysis of system utilization and provide data for user billing. They will also print reports based on the data in the accounting files.

Backup These utilities are used to make backup copies of the system's stored data. They are especially useful for error recovery procedures and will be used to periodically copy active files from disk to a floppy disk, magnetic tape, or other backup medium. This protects users against accidental destruction or modification of important files.

cpio (Copy In/Out)
A utility used to compress and archive files. The **cpio** utility is typically used by the System Administrator when performing backups, but is also available to all users. This utility is particularly helpful when you want to transfer files to another UNIX system.

fsck (Files System Check)
This utility, available only to the Superuser, checks directories and file relationships as they appear on the disk and reports inconsistencies and errors.

Program Notes - continued

passwd This utility facilitates the tasks of assigning and changing user passwords.

mail

The **mail** command allows you to send electronic mail to one or more users on the system. At login you will be notified that mail is waiting. A mail file exists for every user as a text file. It is uniquely identified by the user's login name.

Sending mail

To send a message, type "mail" at the shell prompt followed by a space and the login name(s) of the user(s) followed by a carriage return. For example, if you are in the Bourne Shell and want to send a message to another user whose login name is carrier2, you type:

mail carrier2

To execute, you press the carriage return key and the cursor will move to the beginning of the next line. You then enter the message. To sent the message and exit **mail**, type a period (.) on the next new line.

Program Notes - continued

To mail an existing file a slightly different procedure is followed. You execute the command "mail" followed by the login name(s), followed by a less than symbol, followed by the name of the text file to be sent. For example, if you want to send the file "finance" to login carrier2, you type:

mail carrier2 < finance

If you specify an incorrect login name, the mail program will save your mail message in a file called **dead.letter.** If you are sending mail to multiple users and you specify an invalid login name, the mail program will send your message to all the valid logins and also save it in the **dead.letter** file. In the following example we will resend the mail message saved in the **dead.letter** files to the login "carrier3".

mail carrier3 < dead.letter

This technique uses the input redirection symbol (<) to read the mail message from a file rather than from the terminal.

Reading mail

At login you will be notified if mail is waiting. After login you will be notified of incoming mail within ten minutes after it is received. You may check for mail messages at any time by typing "mail" and pressing the return key or carriage return.

Once the **mail** command is entered the messages will be displayed in a last in/first out order. The last message received will appear on the screen followed

Program Notes - continued

by a question mark (**?**) - the mail prompt. To get a listing of all the mail subcommands, type a question mark (**?**) or an upper case **H** at the mail prompt.

The following summarizes some of the basic mail subcommand functions. To:

read next message	Press: Return or Type: **n** or Type: +
read previous message	Type: -
reread current message	Type: **p**
delete current message	Type: **d**
undelete last deleted message	Type: **u**
reply to sender of current message	Type: **r**
reply to sender and other users	Type: **r** "login names"
forward current message to other users	Type: **m** "login names"
exit mail	Type: **q** or Type: **x**

Program Notes - continued

Note: When exiting mail, the **q** command will preserve all unread mail messages, allowing you to read them at a later date. By using the **x** command you will exit the mail program, leaving all messages, read and unread, as if you never invoked the mail program.

write

If an immediate response is needed the **write** command is used instead of the mail program. This command allows an interactive two-way conversation to be conducted between users.

To initiate the **write** command first use the **who** command to see if the person to be communicated with (for example, user2) is currently logged in the system. Type:

write user2

After you press the return key, user2 will be notified that you (user1 for this example) are writing to him/her. Typically, user2 would enter the **write** command followed by your login name and begin a dialogue with you. To terminate your **write** session, type **control-d**. This will display **[EOF]** on the screen of the user to whom you were writing.

If you do not wish to be interrupted by write messages, type the command **mesg** followed by a space and the letter **n**. This command can be placed in the user's **.profile** file and will prevent interruptions by other users. Any user trying to contact a user who has entered a no message

Program Notes - continued

command will be notified. To start receiving messages again type **mesg** followed by a space followed by the letter **y**.

Printing Files

There are 3 commands associated with the printing of files, they are:

lp
lpstat
cancel

UNIX uses a spooling system to place print file orders in a queue and print them in an orderly sequence.

The **lp** command is used to print a file. For example, the command

lp memo

would direct the system to print the file "memo" on the system's default printer. After entering this command, the system would respond with a request-id that uniquely identifies your print job.

If you are updating a file and initiate a print order, the original version of the spooled file may not be printed. The updated version will be printed instead. To create a copy of the original file in this circumstance type **lp** followed by the minus c option (**-c**) followed by a space and the name of the file.

Program Notes - continued

This will create a copy of the original file and allow the user to update and not print the revised file. For example:

lp -c memo

To print multiple copies the **-n** option is used. Type **lp** followed by the number of copies desired followed by the filename. For example, to print 2 copies of the file "memo", enter the command:

lp -n2 memo

The **-m** option cause the **lp** command to send you mail after your files have printed. For example:

lp -m memo

All of the print options can be combined when initiating a print command. For example:

lp -cmn4 memo

will print 4 copies of the file "memo", first making a copy of the file and send you mail when it has printed.

The **lpstat** command is used to check the status of print jobs. The system will list all of your jobs waiting to be printed. If the **lpstat** command returns the shell prompt instead of a listing, then all of your jobs have been printed.

Program Notes - continued

The following is one example of using the **lp** command to print a file, using the **lpstat** command to check on the printing status, and then using the **cancel** command to cancel the printing of the file.

```
$ lp myfile
request id lp-4166 (1 file)
$ lpstat
lp-4166    user1    240 Dec 7 11:43
$ cancel lp-4166
request "lp-4166" canceled
$
```

To remove a file from the print queue, the **cancel** command is used. You must first issue the **lpstat** command to learn the request id of the file, then type **cancel** followed by a space followed by the request id.

Program Notes - continued

Calendar Reminder Service

If you would like to build your own reminder service using the UNIX calendar command, perform the following steps.

1. After you have logged in, type: **cd**
   to ensure that you are in your home directory.

2. Next, install the calendar command in your **.profile** file.

   Type: **echo calendar >> .profile**

   Now, every time you login the system will automatically invoke the **calendar** command.

3. To create reminder messages,

   type: **echo date message >> calendar**

   where:
   date = the date of the event (e.g. 1/12/90)
   message = your reminder message

Example

Enter:
echo 3/20/90 First Day of Spring! >> calendar

Now, on 3/19/90 and 3/20/90 the message:

3/20/90 First Day of Spring!

will display on your terminal after you have logged in.

Lab Exercises

1. Login, paying special attention to any messages of the day.

2. Enter: **news** and read all news articles.

3. Enter: **mail** to check if you have received any mail.

4. Use the **mail** command to send yourself a mail message.

5. Use the **mail** command to read your meassage and save it in a file called "mfile".

6. Use the **write** command to communicate with another active user.

7. Use the **lp** command to generate a printout of your file "mfile".

8. Use the **lpstat** command to check on the status of your print job.

9. Attempt to cancel your print job using the **cancel** command.

Unit 6

The Productive User

Overview

Introduction

This is the last unit in the Quick Start series. It focuses on ways to help you become a *Productive User*. This unit will present additional information on topics which are essential to your effective use of the UNIX Operating System. It will discuss the UNIX Reference Manual, on-line help features, running applications, programming facilities, and the need for user maintenance. To most effectively use this unit, the UNIX Reference Manual for your particular system should be available for your use.

Key Terms

The following terms are used in this unit and the accompanying video module. Please review the definitions before viewing the video and use the listing as a reference when you are completing the program notes that follow.

on-line help
: data base facility containing information that can assist you to perform a specific task

reference manual
: system-specific user guide containing detailed system operating information

Objectives

At the end of this unit you should be able to:

1. State in general terms the type of information found in each section of the UNIX Reference Manual for your particular system.

2. Use the on-line help facility to extract information on UNIX commands and topics.

3. List the 5 general categories of applications.

4. List the standard programming facilities available on a UNIX system.

5. Identify commands and techniques that users can use to efficiently maintain their files and directories.

Program Notes

Reference Manual Contents

The UNIX Reference Manual provides a written reference for all levels of users. The organization of the manual and a synopsis of the contents of each section are listed below:

Section I: *Commands and Application Programs* addresses UNIX commands that you can invoke, such as general purpose commands (**ls, date, who**), communication commands (**uucp** and **cu**), and system maintenance commands

Section II: *System Calls* presents documentation on lower-level kernel routines and is used primarily by C programmers to find information they can use to create their own programs

Section III: *Subroutines* describes the available subroutine libraries that can be used in C, assembler, and Fortran programs

Section IV: *File Formats* details the structure of system files

Section V: *Miscellaneous Facilities* describes character sets, text formatting macro packages, and terminal names

Section VI: *Games* documents various educational programs

Program Notes -- continued

Section VII: *Special Hardware Device Files* discusses special hardware devices such as disks, tape drivers, printers, and terminal interfaces

Section VIII: *System Maintenance Procedures* addresses those tasks that are essential to maintaining the UNIX system

Sections 1 and 6 are of primary interest to the general user population. Sections 2 through 5 are geared for programmers, and Sections 1M and 8 are intended for System Administrators.

Reference Manual Format

Each page in the manual follows a standard format and all information is categorized under a series of headings which include *name*, *synopsis*, *description*, *examples*, *files*, *diagnostics*, *warnings*, *see also*, *bugs*, and *origin*. However, not all headings are included for every command.

name gives command name and associated names followed by a brief one-line description of the command function

synopsis provides command format and syntax

description explains exactly what the command does

Program Notes - continued

examples provides one or more ways in which the command might be used

files lists any other file or files associated with the command

diagnostics common error messages generated by the command.

warning details the possible problems that may be encountered when executing the command

see also references related commands

bugs describes any documented problems that may occur when executing the command

origin identifies the source provider of the command

In concluding our overview of the UNIX Reference Manual, its should be noted that in some larger UNIX systems the reference manual is stored on-line and can be accessed by typing **man** followed by a space and a command name. For example, to access information on the **cat** command, type:

man cat

On-line Help

The on-line help feature provides a database of information to help you perform a specific task. It is invoked by using the **help** command. This feature

Program Notes - continued

requires a very large amount of disk space and is therefore not found on all systems.

When the help command is entered, the system responds with a menu of 6 options. Four of these options are directly related to finding information:

- starter
- locate
- usage
- glossary

These options are discussed in detail below.

starter

The **starter** option furnishes general information for the novice user. It includes commands and terms that are considered prerequisites to using the UNIX system. In starter, inputting an **s** will activate another menu which identifies the type of information available. Inputting **c** furnishes information of the simplest commands and features of the system. **d** provides a bibliography of useful documents. **l** supplies information relevant to the capabilities of the user's specific machine, and **t** lists other teaching aids which are available on-line.

locate

The **locate** option identifies UNIX commands. Commands can be identified by keyword and function. Inputting a **l** will produce a submenu. On that submenu **k** permits the user to identify a list of key words. For example, if "print" is listed as a keyword, the system locates a number of commands associated with printing files.

Program Notes - continued

usage

The **usage** option displays information about specific UNIX commands. You enter the command name and the system lets you determine whether you want a description of the command, its options, or an example of how the command is used. By entering a **u** at the on-line help menu, the user is presented with specific information on how to use a command.

glossary

The **glossary** option is used to look up the definitions of UNIX system terms. If the glossary option is selected from the on-line help menu, the user simply enters the term or character and the help system retrieves a short but useful definition which is not available in the user's manual.

Once in the help subsystem, you must enter the appropriate command (starter, locate, usage, and glossary) at the shell prompt. If the command is input without arguments, the appropriate submenu will be displayed. If the command is entered with an argument the help system immediately furnishes the information requested.

In summary, there are two source of reference materials available to you. **What are the two available reference sources?**

1. _____

2. _____

Program Notes - continued

The two additional help facilities which are available on some systems are "apropos" and "learn".

Apropos finds commands in the manual by identifying keywords. Learn is a UNIX tutorial. The user should check with the System Administrator to determine which help facilities are available for use.

Applications Programs

Applications programs are available in the following categories: word processing, financial, accounting spreadsheets, database management, communications, and computer graphics.

Most UNIX systems provide some standard applications in these categories, however you can typically find more sophisticated and user-friendly applications from third party vendors.

The two word processing programs most frequently available for UNIX systems are **nroff** and **troff**. Both are text formatters and act similar to compilers for a text formatting language. Their names refer to both the formatting language and to the commands in the UNIX system that process the source documents.

nroff and **troff** use embedded commands which are translated by the system into formatting actions. These commands begin on a new line in the source document and are interspersed with lines that contain text. These are not "what you see is what

Program Notes - continued

get" (WYSIWYG) word processors. **nroff** and **troff** are powerful word processors, but they are also very difficult to learn and many UNIX system users purchase word processing packages from third party vendors.

The communications programs on the UNIX system include **mail, write, uucp**, and **cu. mail** allows the user to send and receive electronic mail. **write** lets two users communicate interactively through their respective terminals. The **mail** and **write** commands were discussed in greater detail in Unit 5.

uucp permits files to be transferred between UNIX systems across a network. **uucp** is a very complete, sophisticated data movement package which can transfer files between machines as well as control command execution on a remote system. The **uucp** subsystem contains many commands, functions, and customization capabilities for different communication networks, security features, logging and debugging tools, and a number of different data transfer protocols.

cu is actually a part of the uucp data transfer system. It uses **uucp** control files and an external and/or a built-in modem. The **cu** command is executed with a system name as an argument. The **cu** command consults the **uucp** control files to find out how to set up a connection to the named machine. Then it calls that system through a modem over telephone lines, a hardwired permanent data cable, or across a local area network.

Program Notes - continued

If the connection is not made an error message appears. If the connection is made, cu returns a "connected" message and waits for the user to enter data through the keyboard. The data is transmitted to the remote machine and any response from the remote is also sent through **cu**.

The System Administrator is responsible for installing and executing these and any other applications programs.

Programming Capabilities

Programming capabilities found on most UNIX systems include Shell programming features and C language development tools. In addition to the Shell's ability to function as a command interpreter and user interface, it provides features of a programming language. This allows users to store command sequences in executable files and then subsequently execute them by entering the name of the file.

Other shell program features include variable assignment, control structures such as if-then-else, for, while, case statements, I/O redirection, and command substitution. The details of these constructs are covered in the course on shell programming.

The C language is one of the finest programming languages available. It provides flexibility, accommodating both low and high level abstract programming concepts. There are several specific C

Program Notes - continued

language programming tools that are important. They are the C compiler, the make utility, and the sccs (Source Code Control System).

The C compiler is the heart of C programming. It is accessed through the **cc** command and allows a programmer to convert C language source code into executable programs. The **make** utility reduces the development time required for building applications which are a compilation of multiple C programs. **make** relies on a series of pre-determined source files and target application dependencies to calculate the minimum amount of time needed to recompile an application. It then rebuilds or reconstructs both the module and the application. The source control system (**sccs**) allows a programmer to maintain multiple versions of an application. **sccs** provides the tools necessary to update, extract, and manage this development history.

User Maintenace

All users must organize and maintain their own files and directories in the UNIX environment. Files are created as a product of a particular programming task, or through an editor. As more and more files are generated, the method used to name files becomes crucial. By creating subdirectories you can organize your files in a logical structure. Two commands essential to organizing files are make directory (**mkdir**) and move (**mv**). The make directory command creates a directory while the move command takes an existing file and relocates or renames it.

Program Notes - continued

You can create subdirectories as a means of organizing files. You should also remove unwanted files. To remove a file from a directory you execute the remove (**rm**) command. Once a file is removed it cannot be brought back unless it was on the system when backup tapes were written.

Lab Exercises

1. Obtain a copy of the UNIX System User's Reference Guide and review the introductory chapters.

2. Login

3. Use the man command to display manual pages for the **who**, **date**, and **ls** commands.

4. Use the help command to invoke the online help facility. Browse through the **starter**, **locate**, **usage**, and **glossary** functions.

5. Logoff

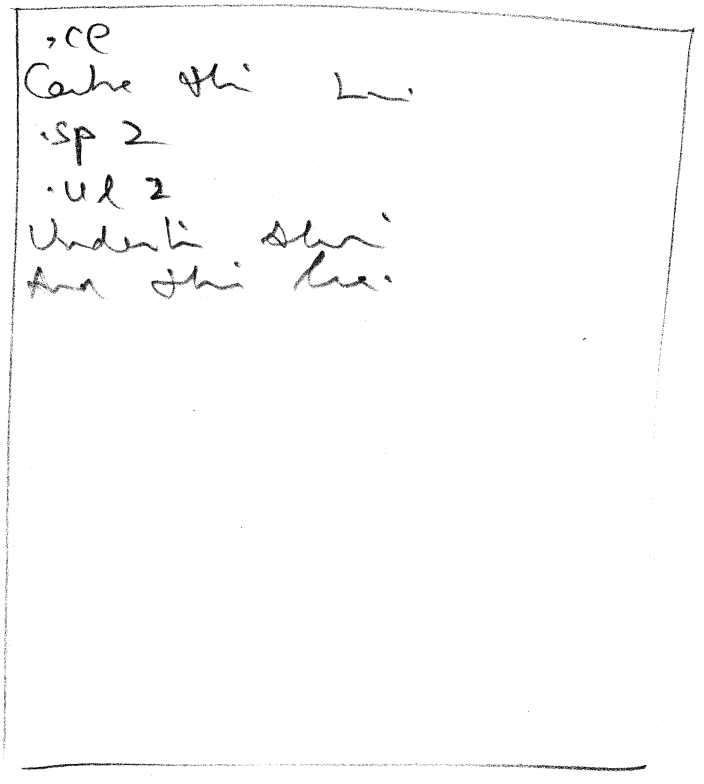
VI editor commands

These "dot" commands are suitable for use with the UNIX formatter "nroff"

Command	Meaning
.sp #	space next # lines
.ce #	centre next # lines
.ti #	indent next line # spaces to right
.ul #	underline next # lines
.ll #unit	set line length at # units eg 10cm for 10 centimetres
.pl #unit	set page length at # units
.po #unit	print offset # units
.nf	no fill on next lines till .ff used
.fi	fill lines of text
.na	no justify right margin
.ad	adjust right margin
.bp	page break

When used in a document the comamnd to then format the file is
nroff filename

/usr/ucb/nroff



~/nroff file > file

Additional Macros are Available for use with nroff

.PP	include a space line and indent first line 5 spaces to right
.LP	include a space line, with no left indent
.IP	include a space line, all lines indented
.IP "label"	a labelled IP
.IP string #	indent # spaces for string eg .IP Example\0-3-12
.QP	a quoted paragraph
.XP	an extendend paragraph
.SH	section header
.NH 1	numbered heading
.NH 2	numbered heading now at 1.1
.NH 3	numbered heading now at 1.2.1
.NH 3	numbered heading now at 1.2.2

.NH 2

Usage nroff -ms filename

/usr/ucb/nroff -ms flex > fb

3901437 Roger Murphy.

Keyboard Mappings

Map a string to a keyboard character

:map % ohello Use % key to open line below, insert hello

:map ! o.PP^V Report,^V^MMr.^V Spock^V^M^V^MYes,^V Captain^[

Open a line below, ^V used prior to space, return, backspace

Abbreviations

:ab sta Start Trek Addicts

:unab sta

Glossary

UNIX Quick Start

Glossary

A

application program that runs on top of the base operating system performing a specific function such as word processing or database management

argument final component of a command that defines the object the command acts upon, typically a file name

ASCII American Standard Code of Information Interchange

B

batch processing
type of information processing in which the computer saves several pieces of information and then processes them at one time

boot up to start-up the system

C

C language machine-independent programming language that combines the structured programming capabilities of other programming languages with the low-level capabilities of machine language

calendar program for organizing and managing appointments and other items of interest

command instruction to the system that causes a specific action or set of actions to take place; a command can be divided into three parts - command name, options, argument

command alias
shell's ability to create a macro

command history
shell's ability to keep a log of all commands entered during a session

cpio (copy in/out)
utility used to backup and recover files

D

directory logical component of the UNIX file system used to "hold" files and subdirectories

F

file collection of related data that is maintained as a unit

full path name
directions that start at the root directory and lead through a unique sequence of directions to a desired location (sometimes called absolute pathname)

G

gid (group identification number)
number associated with a group name that is assigned by the System Administrator identifying the group a user belong to

GUI (graphical user interface)
type of User Agent that relies on icons to represent system commands, devices, and utilities

H

hardwired terminal connected directly to the computer system by a cable

hierarchy organization pattern in which directories and files are subordinate to one another; the relationship allows for many layers of files and directories

Glossary

home directory
directory assigned to a unique user; cannot be added to or deleted by another user without permission

I

initialization
internal process which prepares the UNIX system to execute commands

input information entered into the computer by the user

I/O redirection
method to change the source of standard input and the destination of standard output

interactive processing
type of information processing in which the computer processes all information immediately

K

kernel core of the UNIX operating system that performs important functions such as control of all system hardware, task management, and data storage management

L

login name characters used to access the UNIX system

M

machine language
hardware-dependent, low-level programming language that operates at the "bits and bytes" level that the computer can interpret

macro technique that provides a shorthand method of executing commands

mail command for sending messages (electronic mail) to one or more users

metacharacter
one of a set of characters that can be used as shorthand notation when referencing file names, sometimes referred to as file name generation characters

modem device which allows a user to access the computer system via telephone lines

motd (Message of the Day)
information file maintained by the System Administrator

MS-DOS Microsoft Disk Operating System, a single-user operating system developed primarily for use on the IBM-PC

N

naming convention
combination of symbols and alphanumerics that can or cannot be used to name directories or files

.news_time file located in your home directory used to track which news items have been read

network communications system which provides interconnections between multiple users and one or more computer systems

O

on-line help
data base facility containing information that can assist you to perform a specific task

option second element of a command that modifies the behavior of the command

operating system
collection of programs that controls the system hardware, runs user-requested programs, schedules events, and provides interface between user and system

ordinary files
collection of file characters stored on a disk

OS/2 multi-task operating system developed by Microsoft for the IBM environment

output the computer's response to input

P

password characters used in conjunction with the login name to ensure system security

path name
unique name which shows the location of a file or directory and provides directions for reaching it

pipeline shell feature that permits the user to connect the standard output of one command to the standard input of the following command

proprietary programs written by a particular vendor solely for use on their own machines

R

read ahead UNIX feature that allows user to continue entering requests to the system while the system processes previously entered requests

reference manual
system-specific user guide containing detailed system operating information

relative path name
directions that start in the current working directory and lead up or down through a series of directories to a particular file

request id number assigned by the system to uniquely identify print jobs

root directory
source directory which can be used to reach other major file system directories

S

shell program that functions as a command interpreter translating user entered commands into instruction executed by the operating system

shellscript simple shell program consisting of a single UNIX command or multiple commands stored in an executable file

special characters
characters used in conjunction with commands that can enhance command functionality and user productivity

special file
file representing a physical device such as a terminal, disk drive, magnetic tape drive, or communication link

Superuser System Administrator or person responsible for overseeing the activities of all users and managing, maintaining, and ensuring the security of the system

T

terminal input/output device

U

uid (user identification number)
number assigned by the system to each user, associated with the login name

user agent type of user interface that uses full-screen or window-oriented menus and/or graphics to provide assistance in using the operating system

user environment
capabilities and permissions established within a user's shell that determine the scope of their functionality

user interface
program that transforms user requests into actions that are executed by the operating system

/usr/news (User News Directory)
files of news items which the System Administrator believes are important to users

utility program that assists the user in performing a particular task

W

write command that allows interactive communication between two active users via their terminals