VAX/VMS Software
VMS System Software Handbook

VAX/VMS Software

VMS System Software Handbook

Contents

Chapter 1 ▪ Introduction To VAX/VMS Software

Chapter 4 ▪ The System Manager

Chapter 5 ▪ VAXcluster Software

Chapter 6 ▪ VAX Networking and Communication Software

Chapter 7 ▪ The MicroVMS Operating System

List of Figures

Preface

The *VAX/VMS System Software Handbook* is a guide to the VMS operating system and a number of its associated software products. These products and services have been designed to help you control the distribution and security of computing resources throughout your organization. This handbook is part of a three-volume set titled, *VAX/VMS Software*. The other handbooks in this set include, *The VAX/VMS Languages and Tools Handbook* and *The VAX Information Management Handbook*.

Handbook Organization

- Chapter 1 — "Introduction to the VAX/VMS Productivity Environment and its Software Products" gives you an introduction to the family of VAX/VMS software and the single integrated environment it produces. If you are not familiar with our software product or their relationship to each other and to the VMS operating system, this chapter will give you the overview you need before proceeding to the specifics of distributed processing and system management and security.
- Chapter 2 — "The Digital Command Language." This chapter describes the Digital Command Language and its various components.
- Chapter 3 — "System Security" gives you an overview of VMS security features and how those features can be used to safe guard your organization's computing resources.
- Chapter 4 — "The System Manager." This chapter describes the powers and responsibilities of the VMS system manager. Many of the utilities and VMS services used by the system manager are highlighted in this chapter.
- Chapter 5 — "VAXcluster Software." This chapter gives you an overview of VAXcluster systems and the complex software components of those systems.
- Chapter 6 — "VAX Networking and Communication Software" covers DECnet-VAX and VAX PSI software products.
- Chapter 7 — "The MicroVMS Operating System." This brief chapter introduces you to MicroVMS and its components. MicroVMS is a modular version of the VMS operating system for use on MircoVAX processors.

- SCOPE OF THE VAX/VMS HANDBOOK SET

  It is not the intent of this handbook set to describe all VAX/VMS software products. Products covered in these three volumes pertain to the subject of the specific handbook they appear in. For example, the *VAX Languages and Tools Handbook* describes only the software products used for software development. Many VAX/VMS software product groups not discussed in these three volumes are covered in other handbooks or other documentation.

- RELATED PUBLICATIONS

  These publications provide additional information on VAX/VMS software products and can be obtained through your local Digital Sales Office or Sales Representative.

  - *VAX Software Source Book* Volume 1, Application Software; Volume 2, System Software
  - *The Digital Dictionary*
  - VAX/VMS *Internals and Data Structures*
  - Software Product Description (Alphabetical Index – SPD 00.01.11)
  - The Peripherals and Supplies Group's *Documentation Products Directory*

Chapter 1 ▪ Introduction to VAX/VMS Software Products

▪ Overview

Computing resources, hardware and software, enable an organization to effectively deal with a multiplicity of both day-to-day and long-range operating needs. Typically, these include data processing, program development, and information management requirements.

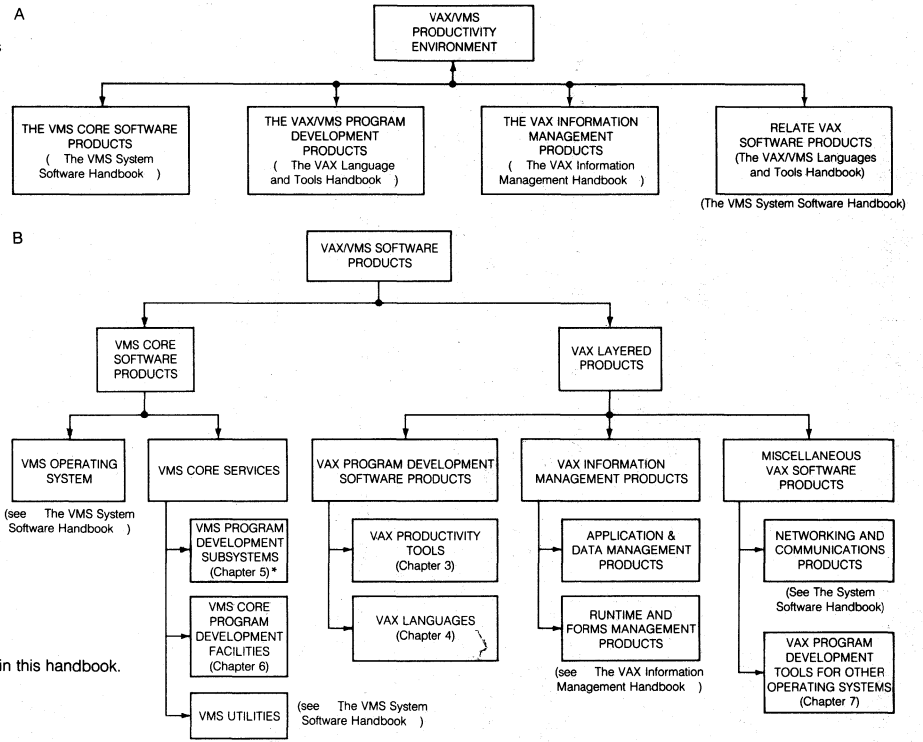
A grouping of computer resources creates a computing environment. To be effective, a computing environment must be made up of resources that are compatible with each other and that have been designed to work together toward a common goal. Digital's software products are designed to create such a computing environment. In total, these products create the VAX/VMS Productivity Environment.

VMS software products can be discussed from two distinct vantage points: from an individual, or functional basis, and from a more global, or general, view in which each product becomes part of a larger unit — the productivity environment.

This chapter gives you an introduction to the VAX/VMS environment and to the individual software products it is composed of.

This chapter introduces VMS software products and explains these products in terms of

1. The productivity environments that are created when various products are combined
2. The individual software products themselves and their relationship to the VMS operating system



*Note: Chapter references are to chapters in this handbook.

Figure 1-1 ■ Overview of Chapter 1

▪ Introduction To The VAX/VMS Software Groups

Digital's VAX/VMS productivity environment (see Figure 1-2) is made up of many different software products. For the ease of presentation, all these products are divided, by function, into four logical groups. Each of these groups is a subset of the overall VAX/VMS productivity environment and provides VAX/VMS users with specific capabilities. These four groups are

- VMS core software products.
- VAX program development software products.
- VAX information management software products.
- Related VAX software products.

The VMS Core Software Group

This group of software products includes the VMS operating system and a wide array of services and utilities. It serves as the foundation from which all VAX optional software products (and applications generated by those products) operate.

The VMS core software group provides one productivity environment to users who have many different needs. For example, program developers, whose day-to-day work environment is controlled by the security features found in the VMS operating system, can incorporate those same features into the programs they are writing. Or, transaction-processing operators, whose physical locations are scattered around the globe (and are controlled by the same security features mentioned above), can access one central database through the distributed processing capabilities found in the VMS operating system.

This volume of the VAX/VMS handbook-set, *VMS System Software*, covers the function of the VMS operating system and its subsystems in the VAX/VMS environment.

The MicroVMS operating system gives MicroVAX users the benefits of the the VMS productivity environment. This operating system supports Digital's MicroVAX processors, and is fully compatible with the VMS operating system. In fact, MicroVMS gives MicroVAX users all the same features (except PDP-11 compatibility operations) enjoyed by users of full-sized VAX processors. MicroVMS is VMS repackaged for the microcomputer environment.

The MicroVMS operating system is covered in Chapter 7.

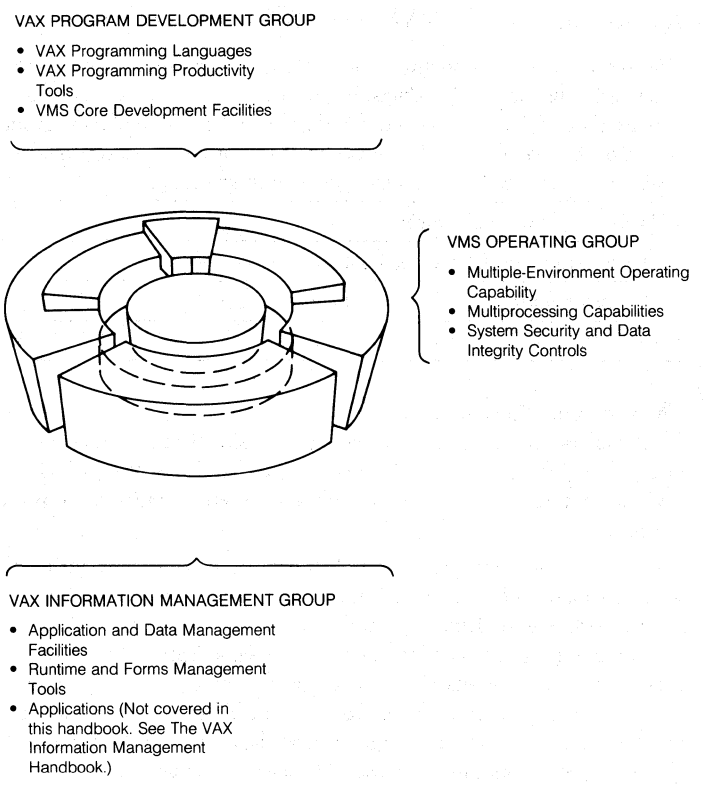


Figure 1-2 ▪ The VAX/VMS Productivity Environment

The VAX Program Development Software Group

The VAX Program Development product group incorporates many of the services and utilities from the Core software group with a number of optional software products. This group is made up of a rich set of VAX programming languages and VAX program development productivity tools. It has been designed to let you develop application programs that run across the full spectrum of VAX processors — MicroVAX to VAXcluster systems.

Included in the VMS program development products group are

- Over a dozen high-level, industry-standard programming languages that are callable from each other — including VAX Ada*, VAX BASIC, VAX C, VAX COBOL, VAX FORTRAN, VAX PASCAL, VAX PL/I, and VAX RPG II.
- Numerous program development productivity tools that simplify many of the tedious and time-consuming chores involved in large-scale program development and intensive small-scale software development.
- Tools that aid in building software systems efficiently and reliably.
- Special tools designed to aid in the migration of programs to and from other Digital operating systems.
- Tools to streamline program development activities.
- Tools designed to make "team" development an efficient operation.

The VAX Information Management Software Group

Digital's VAX Information Management Software group offers you a variety of integrated software products to meet the diverse information processing needs of your organization.

VAX Information Management products are used for

- Creating a variety of database management systems.
- Application development and control.
- Terminal management.
- Distributed access.
- Graphics.
- Report generating.
- Easy-to-use querying.

A description of The VAX Information Management Architecture and its associated software products can be found in *The VAX Information Management Handbook*, the third volume in the VAX/VMS Software Handbook Set.

The Related VAX Software Group

Related VAX software products enhance the capabilities found in the three groups of software products just discussed. These related products are used to move and develop software products between DEC systems and make it possible for VAX systems to communicate with other DEC systems in a data communications network.

* Ada is a registered trademark of the U.S. Government (Ada Project Office)

These products are

- VAX-11 RSX and VAXELN. These two optional software products allow you to use your VAX as a host development environment. VAX RSX is used to develop PDP-11 application programs on a VAX and VAXELN is used to build statically defined systems that runs on a VAX.
- DECnet-VAX and VAX-PSI. These optional software communication products make it possible for two or more DEC systems to communicate with each other or with other manufactures' systems. These products also give VMS systems the capability of operate in a Packet Switching Data Networks.

VAX- 11 RSX and VAXELN are covered in Chapter 7 of *VAX Languages and Tools handbook*. A detailed description of the DECnet-VAX and VAX PSI software products can be found in Chapter 6 of this handbook.

▪ Introduction to VAX/VMS Software Products

Now that we have briefly introduced you to VAX software products as they are logically grouped by function, let's look at the individual software products that make-up each of these groups. These products are divided into two general categories: core software products and optional (layered) software products.

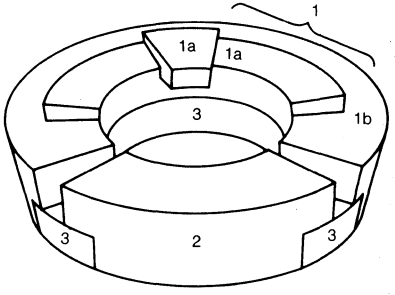
When you buy a VAX/VMS system, in addition to the various hardware components, you receive a standard set of software programs essential for the basic operation of your system. This software is called VMS core software and is comprised of the VMS operating system, a number of services (some of which we have already mentioned above) that augment the operating system's basic capabilities, and many utility programs that are essential to good programming productivity and system management.

Digital also offers you many optional software products that can be purchased in addition to basic system. These optional products have been designed to work in conjunction with the core software and perform a specific function. The VAX FORTRAN programming language or VAX Database Management System are examples of these optional software products.

Optional software products fall into three categories

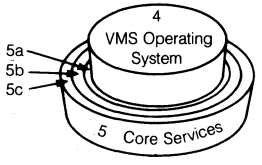
- VAX program development software products.
- VAX Information management software products.
- Related VAX software products.

VAX/VMS software products are divided into two basic categories – VMS Core and VAX Optional Software Products.



VAX OPTIONAL SOFTWARE PRODUCTS

1. VAX Program Development Software Products
   1a. VAX Productivity Tools
   1b. VAX languages
2. VAX Information Management Software Products
3. Miscellaneous VAX Software products



VMS CORE SOFTWARE PRODUCTS

4. VMS/MicroVMS Operating System
5. Core Services
   5a. Core Subsystems
   5b. Core Program Development Facilities
   5c. VMS Utilities

Figure 1-3 ▪ VMS Core And Optional Software Products

VMS Core Software Products

This section introduces you to the VMS core software products (see Figure 1-4).

VMS Core software products include

- The VMS operating system and
- VMS core services
  - Core subsystems
  - Core program development utilities
  - VMS system management utilities

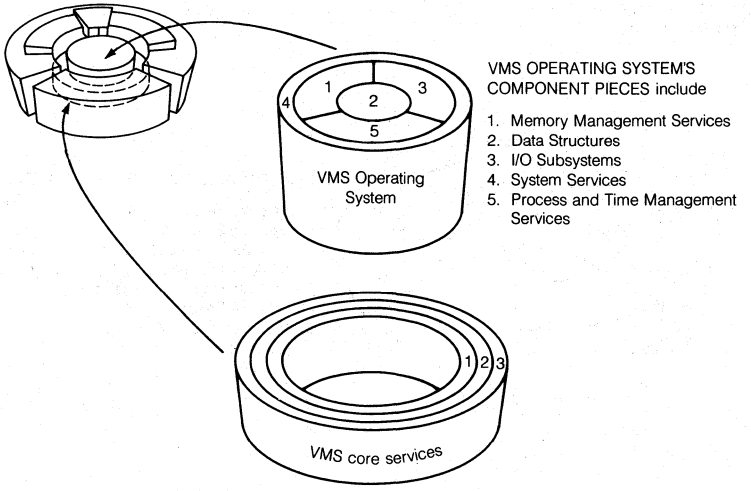


Figure 1-4 ▪ VMS Core Software Products

▪ THE VMS OPERATING SYSTEM

Coordination and management of your VAX system's resources are the central responsibilities of the VMS operating system. An operating system is the group of programs that controls computing operations. The operating system is loaded into memory when the system is initialized, and remains there until the system is powered down. It performs both automatically and manually, and provides users with a consistent operating environment. This simply means that your VAX/VMS system's resources are always used in the most efficient manner possible.

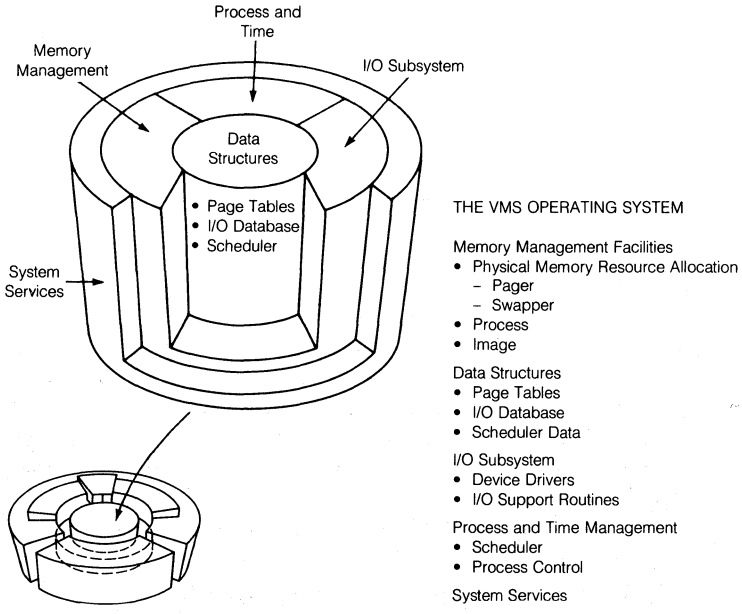


Figure 1-5 ▪ The VMS Operating System

The operating system itself is divided into five components (see Figure 1-5). These include

- Memory management facilities.
- Data structures.
- The I/O subsystem.
- Process and time management services.
- System services.

Applications can be developed and run across the entire line of VAX processors within limits of existing program size and available memory. User-mode applications developed with VMS will run on MicroVMS without modification.

Virtual memory management allows the writing of programs that are larger than physical memory and lets any number of jobs share memory.

Users gain access to VMS is by means of an easy-to-use command language. The Digital Command Language (DCL) is the standard command language available on MicroVMS and all VMS systems. This includes command-line editing and command recall.

VMS gives users extensive online help. Users can receive help by using the HELP command.

VMS system services are callable from programs written in VAX languages that adhere to the VAX common language architecture.

ENVIRONMENTAL FLEXIBILITY - VMS is a general-purpose operating system providing for the concurrent execution, that is, the apparent simultaneous execution, of multi-user time sharing, batch, and real-time applications. Applications programs running under VMS can be divided into several independent subsystems, modules, whose data and code are protected from each other but share common facilities for data sharing and communications. A main goal of VMS is to provide users with a high-performance environment for the execution of these applications.

Environmental flexibility and distributed control are the two major benefits of VMS operating system.

Environmental flexibility means that VAX and VMS can be used in a wide-range of environments. These can be as diverse as a production line CAD/CAM application or the interactive environment of a financial institution or college campus where thousands of users are networked to common databases.

Besides flexibility, the other key attribute of the VMS operating system is control. The VMS operating system gives you sophisticated controls that allow you to tailor the manner in which you distribute and safeguard computing resources throughout your organization.

In the VMS operating environment, any combination of interactive and batch applications will execute concurrently. For example, programmers can create, test, and debug applications while clerical workers are performing online data entry and reporting operations. Still other workers can be submitting jobs for batch execution.

VMS's versatile batch processing capability executes multistream batch jobs concurrently with other system operations. You create any number of batch streams, define spooling queues for these streams and for I/O devices;and create, delete, merge, or redirect queues to alternate devices at any time. (See Chapter 4.)

CONTROLLING THE DISTRIBUTION AND SECURITY OF COMPUTING RESOURCES - When considering the distribution of computing resources throughout your organization, one of your primary considerations should be how to control the availability and security of computing resources.

Because the computing needs of most organizations are so varied, the VMS operating environment gives you a number of options when controlling resource distribution. Processors and their associated resources, terminals, disk drives, or printers, for example, can either be distributed in a centralized or decentralized manner. With VMS you have the advantage using either method or a mix of both.

Decentralized Processing. If there is a need in your organization to give many individuals or small groups of individuals their own processing resources while still being tied together for the common sharing of data, then the VMS operating environment offers extensive capabilities that permit the linking of computers and terminals into flexible configurations called networks. Networks vastly increase the efficiency and cost-effectiveness of data processing operations.

Networking allows computer systems and terminals — whether located around your facility or around the world — to share resources and to exchange information, files, programs, and control. The smaller computers in a network have access to the powerful capabilities of larger systems, while the larger computers can take advantage of smaller dedicated systems for specific application environments. (For more information on VAX networking and communications software products, see Chapter 6 of this handbook.)

Centralized Processing. An alternative to networking your computing resources into a decentralized network is grouping those assets into one highly centralize "cluster" of processors, users, and mass-storage devices. With this approach to distributing processing throughout your organization, not only is it possible to expand your hardware resources on an incremental basis, but your clustered system become highly available and users can simultaneously access system-wide data to the file and record level. (For more information on VAXcluster systems, see Chapter 5 of this handbook.)

VIRTUAL MEMORY MANAGEMENT - Although your system's physical memory may vary in size, depending on the type or configuration of the processor, it is always a finite value. And because physical memory size is a limiting factor when running large multi-user programs, it is necessary to incur the expense of adding memory as the size and sophistication of your applications grow.

To optimize your specific computing environment, VMS gives you a great deal of control over the memory management operations of the operating system. This capability eliminates a traditional concern among users who have equated memory management operations with slow system response and decreased program performance. As your situation dictates, you can "turn on or off" many of the features.

With VMS's virtual memory management technique, large programs, requiring much more space than available in your processors' physical memory, can be run simultaneous with other large programs. This is accomplished by a complex sequence of operations, called virtual memory management. Large segments of code and data are actually moved in and out of physical memory as they are needed. Swapping, as this process is called, is transparent to the user because of the high rate of speed at which it is accomplished.

EVENT-DRIVEN PRIORITY SCHEDULING - The VMS operating system schedules processor time and memory residency on a preemptive priority basis. This means that all processor operations are assigned a priority and executed based on that priority. Thus, a real-time process, controlling a production line operation with a tolerance of + or–of a millisecond, does not have to compete with lower-priority operation for scheduling services. Scheduling also rotates among processes of the same priority.

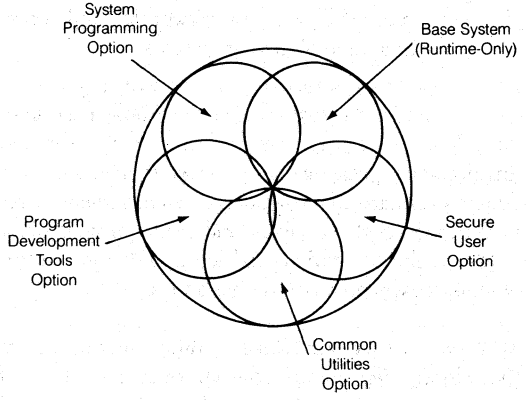
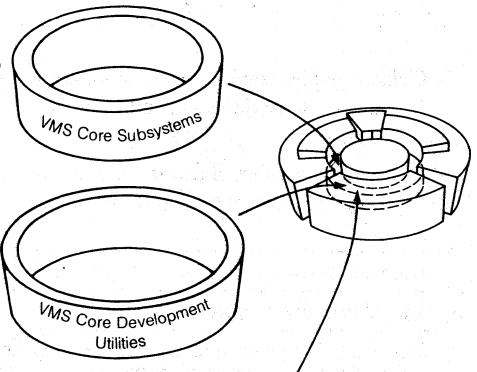


Figure 1-6 ▪ The MicroVMS Operating System

- THE VMS CORE SERVICES

  Complementing the operating system are three layers of services essential for basic system operation and program development. These are called the VMS core services (see Figure 1-7) and include

  - Core subsystems
    - The Digital Command Language (DCL)
    - The VMS Runtime Library (RTL)
    - VMS Record Management Services (RMS)
  - Core program development utilities
    - VMS text-processing utilities
      - * VMS Text Editors
      - * The DSR text-formatting utility – VAX DEBUGGER
    - VAX Linker
    - VAX Sort/Merge
    - VAX Librarian
  - Other VMS core utilities



VMS CORE SUBSYSTEM

The Digital Command Language (DCL)

VMS Runtime Library (RTL)

VMS Record Management Services (RMS)

VMS CORE PROGRAM DEVELOPMENT FACILITIES

VMS Text Processing Facilities
- EDT Text Editor
- TECO Text Editor
- DSR Text-Formatting Utility

VMS Program Development Utilities
- Major Utilities
  - DEBUG-32 (Debugger)
  - Soft/Merge
  - Linker
  - Librarian

SYSTEM MANAGEMENT AND USER UTILITIES

Major Utilities for:
- System Generation (SYSGEN)
- System Accounting (ACCOUNTING)
- Assigning System Privileges (AUTHORIZE)
- Analyzing System Failures (SYSTEM DUMP ANALYZER)
- Monitoring Cluster Activity (SHOW CLUSTER)

Figure 1-7 ▪ VMS Core Software Services

VMS CORE SUBSYSTEMS - Adjacent to the VMS operating system is a set of program routines that provide a communication interface with the operating system (the Digital Command Language,) a device-independent method of handling I/O within an application program (The VMS Record Management Services), and a common library of runtime routines (the VMS Runtime Library.)

These three sets of routines are call core subsystems. Their function in the development of application and system software are thoroughly explained in Chapter 5 of the *VAX/VMS Languages and Tools Handbook*. The Digital Command Language (DCL) is also covered in the Chapter 2 of this handbook.

THE DIGITAL COMMAND LANGUAGE (DCL) - The Digital Command Language (DCL) is the language through which you communicate with the VMS operating system. DCL contains an extensive set of commands that allow you to

- Develop and execute programs
- Work with files
- Work with storage media — disks, tapes, or other devices
- Obtain information about your VAX/VMS system
- Modify your work environment

DCL also provides commands for system managers to establish a computing environment that maximizes the efficient use of system resources and provides adequate security.

DCL provides a number of features that permit users to write command procedures to tailor the command interface and even write their own utilities. These features include

- String and integer variables.
- Control flow commands (IF, GOTO)
- Many lexical functions that operate on variable and return flow information.
- File I/O including parameter passing.

For more information on DCL, see chapter 2.

VMS RECORD MANAGEMENT SERVICES (RMS) - Two of the most tedious tasks of writing application programs are categorizing and creating files for the storage of data to be handled within the program. Typically, many different I/O devices, each with its own interface characteristics, are used to store these files; this means that the data in a file must be tied to a specific device (that is, it is device dependent.)

VAX Record Management Services (RMS) are used by programmers to handle I/O within a program. RMS routines are system routines that provide an efficient and flexible means of handling files and their data, and allow sharing of files across languages. RMS is the default I/O service for all VAX languages. Therefore, files written by a program written in one VAX language can be read and used by another program, even if it is written in a different language. (DBMS and Rdb/VMS, optional file management services, are also available. See *The VAX Information Management Handbook.*)

The RMS utilities, and a file definitions language (FDL) complement RMS procedures by providing an efficient means of creating and maintaining RMS files. (For more information on VAX RMS Utilities see Chapter 5 of *The VAX/VMS Languages and Tools Handbook.*)

THE VMS RUNTIME LIBRARY (RTL) - The VMS Runtime Library reduces application design time with a set of language-independent and language-dependent routines. These can be called from any language that adheres to the VAX Common Language Architecture. (See Chapter 2 of *The VAX/VMS Languages and Tools Handbook.*)

Because application programs can call VMS system facilities for data management, peripheral interfacing and networking, you can skip designing many of those elements and instead concentrate on the central application.

The Runtime Library provides runtime support for VAX native-mode, high-level languages. All routines in the Runtime Library follow all standard call and condition handling conventions.

The common Runtime Library provides general string manipulation, I/O and I/O conversions, terminal-independent screen handling, and many more procedures.

VMS TEXT PROCESSING UTILITIES - Two text editors currently included with VMS are EDT and TECO. Digital's Standard Runoff text formatting facility (DSR) is also provided with the core services.

VAXTPU. (For an overview of the VAX Text Processing Utility (VAXTPU), see the *VAX/VMS Languages and Tools Handbook.*)

The EDT Text Editor. EDT is Digital's easy-to-learn editor ideally suited for the novice or general-purpose user. With EDT you can edit in line mode or keypad mode. EDT also has an extensive HELP facility, which is available online during an editing session.

The TECO Text Editor. Digital's TECO is a character-oriented editor that enables advanced users to edit any form of ASCII files — program source files, manuscripts, or correspondence. Because TECO is character oriented, it does not require an entire line of text to be replaced each time a character is changed.

The DSR Text-formatting Utility. Digital's Standard Runoff (DSR) text-formatting utility extends the basic functions of VMS's text editors into a sophisticated text processing system, ideal for creating and maintaining the extensive documentation necessary to support any program development effort. With DSR's powerful command set, the user can skillfully create documentation ranging from a simple form letter to a multichaptered manual.

Readers interested in a more indepth description of the VMS Text Editors and DSR should turn to Chapter 6 of the *VAX/VMS Languages and Tools Handbook.*

VMS PROGRAM DEVELOPMENT UTILITIES - As a part of the core services, the VMS program development utilities are used by programmers in conjunction with VAX Languages and productivity tools. A few of the many VMS program development utilities include

- The VAX DEBUGGER symbolic debugging utility
- The VMS Sort/Merge Utility
- The VMS Linker Utility
- The VMS Librarian Utility
- Other VMS Program Development Utilities

VMS SYSTEM MANAGEMENT AND USER UTILITIES - VMS provide system managers and users many powerful utilities to control the day-to-day operations of their systems.

VAX systems can be fine tuned, optimized, and maintained with the following utility programs

- BACKUP
- SYSGEN
- ACCOUNTING
- AUTHORIZE
- EDIT/ACL
- MOUNT
- SYSTEM DUMP ANALYZER (SDA)
- VERIFY
- INSTALL
- SHOW CLUSTER
- Other VMS Utilities

For more on these utilities, see Chapters 3, 4, and 5.

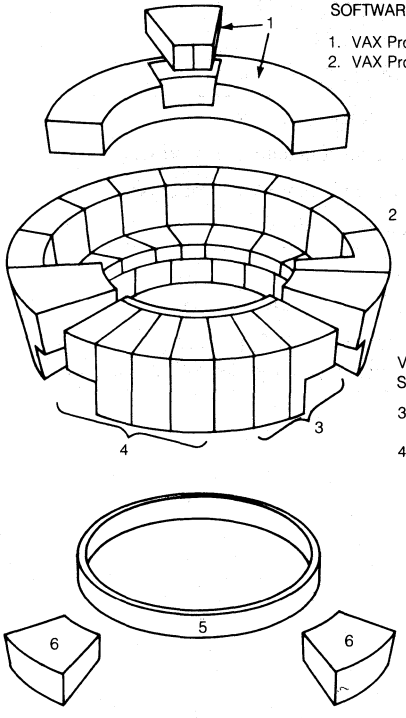
Optional VAX Software Products

Optional VAX software products, often called layered products (see Figure 1-8), fall into three general categories:

- VAX program development software products
  - VAX programming languages
  - VAX program development productivity tools
- VAX information management software products
  - VAX application and data management software products
  - VAX runtime and forms management software products
  - VAX information management applications
- Related VAX software products
  - Communications and networking software products
  - Tools for development and migration of PDP-11 application and system programs — VAX-11 RSX — and statically defined VAX application systems.

VAX OPTIONAL SOFTWARE PRODUCTS

VAX's optional software products fall into three general categories:



VAX OPTIONAL PROGRAM DEVELOPMENT SOFTWARE PRODUCTS include

1. VAX Program Development Productivity Tools
2. VAX Programming Languages

VAX INFORMATION MANAGEMENT SOFTWARE PRODUCTS include

3. VAX Application and Data Management Software Products
4. VAX Runtime and Forms Management Software Products

MISCELLANEOUS VAX SOFTWARE PRODUCTS include

5. VAX Communication and Networking Software Products
6. VAX Tools For the Development and Migration of Application Programs

Figure 1-8 ▪ Optional VAX Software Products

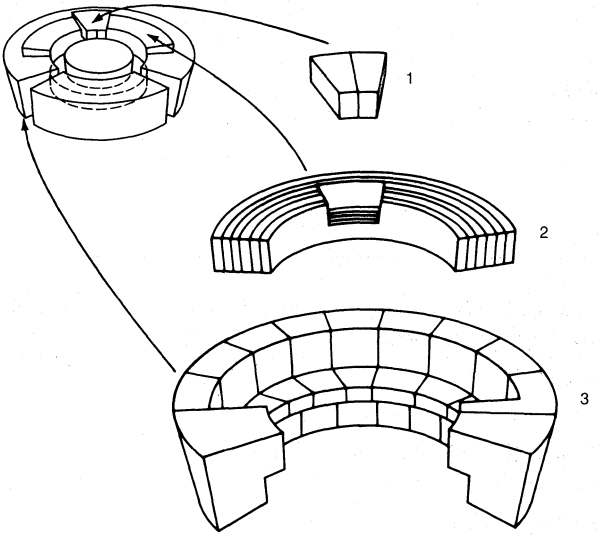
- OPTIONAL VAX PROGRAM DEVELOPMENT PRODUCTS
  Optional VAX program development software products (see Figure 1-9) offer programmers a robust program development environment that includes

  - Over a dozen higher-level programming languages including
    - VAX Ada™
    - VAX BASIC
    - VAX C
    - VAX COBOL
    - VAX FORTRAN
    - VAX PASCAL
    - VAX PL/I
    - VAX RPG II

  Chapter 4 of the *VAX/VMS Languages and Tools Handbook* further describes each of the VAX language products.

  - Productivity tools
    - VAX DEC/CMS (Code Management System)
    - VAX DEC/MMS (Module Management System)
    - VAX DEC/Shell
    - VAX DEC/Test Manager
    - VAX Language-Sensitive Editor
    - VAX Performance and Coverage Analyzer

  In addition to the brief descriptions included in this chapter, you can turn to Chapter 3 of the *VAX/VMS Languages and Tools Handbook* for a more detailed presentation of each of these products.



VAX OPTIONAL PROGRAM DEVELOPMENT
SOFTWARE PRODUCTS

Digital's VAX Program Development Group
is comprised of three integrated sets VAX optional software products

1. VAX Device-independent
   Graphics Application
   Development Tools
2. VAX Program Development
   Productivity Tools
3. VAX Languages

Figure 1-9 ▪ Optional VAX Program Development Software Products

VAX PROGRAM DEVELOPMENT PRODUCTIVITY TOOLS - The VAX program development productivity tools (see Figure 1-10) help extend the programmer's productivity beyond the range of VMS core development utilities and VAX language processors.

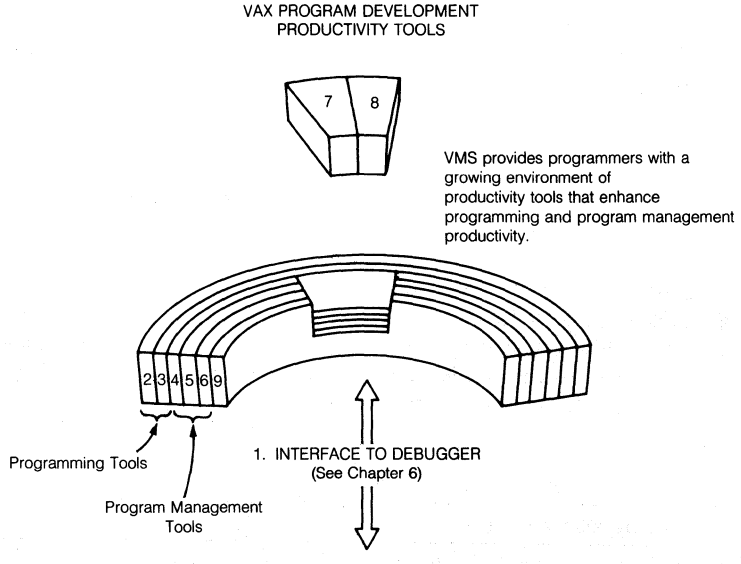


Figure 1-10 ▪ Optional VAX Program Development Productivity Tools

VAX productivity tools enhance the programmer's ability to write, compile, link, and maintain systems and applications designs by addressing multiple phases of the software development life cycle.

Besides serving as the basis for all development work done by VAX/VMS programmers, a subset of these tools, the VNX product set, give ULTRIX or UNIX® programmers the ability to access the VMS Productivity Environment with many commands, utilities, and tools similar to those they use to write programs under the UNIX* or ULTRIX operating systems.

One of these products, VAX DEC/Shell, accesses VMS through a Bourne Shell interface. VAX DEC/Code Management System (CMS) and DEC/Module Management System (MMS) offer programmers capabilities similar to those found in an ULTRIX or UNIX program development environment. VAX C is an excellent compiler for programmers familiar with that language. It is often used in a UNIX program development environment. These tools, together with the rest of the VMS productivity environment, are part of Digital's evolving family of VAX/VMS productivity tools.

VAX DEC/CMS - VAX DEC/Code Management System (CMS) (see Figure 1-10) is a program librarian for the development and evolution of application software in the VMS Productivity Environment. It is comprised of a set of commands that enables software developers to manage the files of ongoing projects. A few of the many operations CMS performs are

- Storing ASCII files in a project library
- Maintaining a report of all file modifications
- Managing concurrent modifications
- Identifying and "freezing" software for release or milestones

VAX DEC/MMS - Digital's VAX Module Management System (MMS)(See Figure 1-10) is designed to manage "system builds" for developers during day-to-day development, maintenance, and implementation of a software system.

In the development of a large software system many of the dependent modules, of which the system is made, are typically in various states of completion. VAX DEC/MMS determines which modules need to be recompiled, and ensures the software system is recompiled and linked with all the latest changes. VAX DEC/MMS can also interact with VAX DEC/CMS and extract files from CMS libraries when building a software system.

VAX DEC/Shell - VAX DEC/Shell (See Figure 1-10) enables programmers, familiar with UNIX V7, to operate in the VMS Productivity Environment with UNIX-style command language and utilities. (These are based on the Bourne Shell.)

UNIX is a registered trademark of Bell Laboratories

When using the DEC/Shell, you are not just limited to its commands and utilities. All the power and features of the Digital Command Language (DCL) are also available. For example, you can use DEC/Shell to write a command procedure with "Shell" and DCL commands. This important feature of the DEC/Shell, which allows you to mix commands in such a manner, is called pipes.

A pipeline allows you to direct the output of one command to be the input of another command. Therefore, you can pipe the output of a DEC/Shell command to the input of a DCL command or that of a DCL command to that of a DEC/Shell command.

The three major components of VAX DEC/Shell are the command-line interpreter, utilities, and the Shell script language.

DEC/Shell

- Can be used as a programming language.
- Provides a small runtime library with some specific and general-purpose routines.
- Is compatible with DEC/CMS and DEC/MMS.
- Offers VMS users the most commonly used UNIX utilities.

VAX DEC/TEST MANAGER - VAX DEC/Test Manager (see Figure 1-10) is an automated regression testing system. That is, it automatically executes user-defined tests on existing software products and compares the test output against the product's benchmarks to determine if the software is performing as expected. It gives the programmer flexibility in organizing and selecting tests for execution, in running those tests, and in verifying and reviewing then tests results.

VAX DEC/Test Manager makes software maintenance more manageable and helps developers during implementation. Because it is simple for developers to insert tests of a specific functionality into a group and to then call those tests (or those of other developers) independently, software system integration is increased while normal unit testing is being done.

Testing is a necessary and costly part of software development. VAX DEC/Test Manager manages the testing process to help you produce higher-quality products with less maintenance.

VAX LANGUAGE-SENSITIVE EDITOR - The VAX Language-Sensitive Editor is a multilanguage, multiwindow, screen-oriented editor designed specifically for program development. It is "language sensitive" in that it provides information to programmers on all the VAX languages that it supports. It gives you full access to detailed templates of all of the major constructs in each of these languages. This enables faster, more accurate program development.

The VAX Language-Sensitive Editor, working with numerous VAX programming languages and the VAX Debugger, provides you with a highly productive, online, interactive programming environment. Within a single editing session you can write code, edit, compile, and review and correct compilation errors. You can customize and extend the editor to meet your unique programming needs.

VAX PERFORMANCE AND COVERAGE ANALYZER - The VAX Performance and Coverage Analyzer (see Figure 1-10) is a program development tool for measuring and tuning the performance of user-mode applications programs. It also measures test coverage, showing that routines, lines, or even individual code paths in a program are executed by a given suite of test input.

The VAX Performance and Coverage Analyzer consists of two parts: the Collector that collects performance or coverage data from a running user program, and the Analyzer that later processes the data to produce various reports. The Collector can collect program counter sampling data, page fault data, system service counts, I/O usage data, exact execution counts at specified locations, and test coverage data.

The Analyzer produces performance histograms and tables that show which parts of a program consume the most resources such as CPU time, page faults, or I/O services. These displays can be at a coarse level (showing resource usage by routine) or at a very detailed level (showing resource usage by individual source line).

The detailed reports can be obtained as source listings annotated with the desired performance or coverage data. The VAX performance and Coverage Analyzer is fully symbolic and can be used with all languages that have DEBUGGER support.

VAX GRAPHICS DEVELOPMENT PRODUCTIVITY TOOLS - An important part of the VMS program development environment, VMS graphics productivity tools are well suited for a number of uses and applications. These tools are intended for the development of device-independent graphics applications programs.

VAX GKS - VAX GKS (see Figure 1-10) is a productivity tool for developing device-independent graphics applications. VAX GKS is ideally suited for writing applications to be used in automotive, CAD/CAM, chemical, educational, environmental, and scientific areas.

With VAX GKS, programmers create one application program for many different graphics I/O devices. No longer do you have to recode applications every time a new device is incorporated into your computing environment. You can produce a FORTRAN program using a compatible version of GKS on an IBM computer, for example, then move it to VAX. It will work there without changes.

VAX GKS is based on the ANSI and ISO graphics standards. VAX GKS, Level 0b, provides basic output and input capabilities.

VAX DECOR - VAX DECOR is a graphics subroutine package that provides an interface between an application program and graphics devices. The interface is device-independent and supports user-developed device handlers, as well as those supplied with VAX DECOR.

VAX DECOR is Digital's implementation of the SIGGRAPH/ACM Graphics Standards Planning Committee's CORE proposal. It provides a device-independent interface between an application program and supported graphic devices.

VAX DECOR is optimized to take advantage of the VAX Information Management capability and is designed to be a development tool for applications programmers in the CAD/CAM area. VAX DECOR is a two-dimensional implementation of the CORE standard.

VAX PROGRAMMING LANGUAGES - At the center of the VAX/VMS program development capability is the family of VAX programming languages (see Figure 1-11). Applications and system programmers have a diversified range of higher-level and assembly-level programming languages at their disposal. In addition to the assembly-level language, VAX MACRO, VAX programming languages cover the gamut from Ada to RPG II. All of these languages take advantage of the native-mode (VMS operating environment). Compatibility-mode (RSX-11 operating environment) capabilities available through VAX/VMS are accessible from PDP-11 compilers that run on the VAX.

User applications may employ more than one language. The ability of languages to call one another allows concatenation of application segments written in a variety of languages.

THE FAMILY OF VAX PROGRAMMING LANGUAGES

The family of VAX languages give application and systems programmers a Common Language Architecture in which applications and systems programs

- Can be written in more than one language. Many VAX languages conform to a common calling standard.
- Use a common debugging utility (VMS DEBUGGER).
- Use a common runtime library VMS RTL).
- Can call operating system services and use a common handling of exceptions.

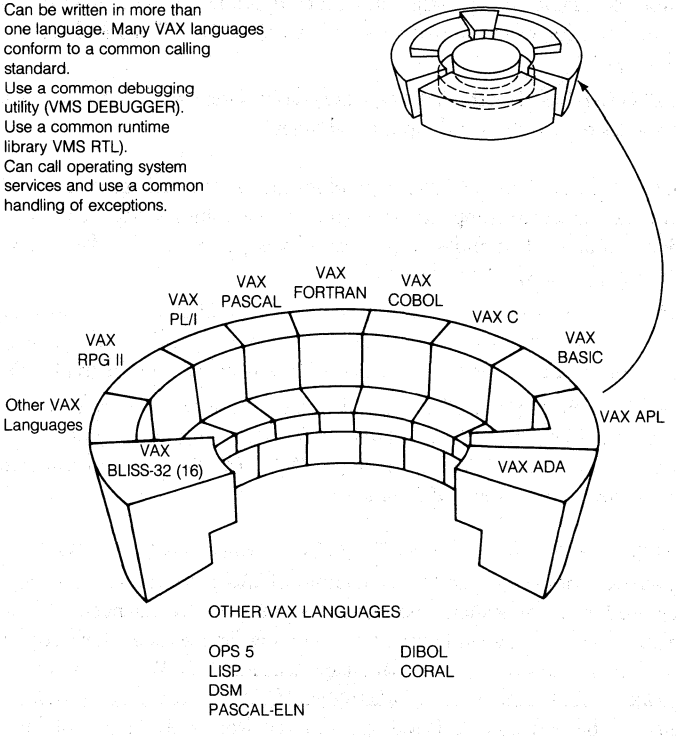


Figure 1-11 ▪ VAX Programming Languages

- VAX INFORMATION MANAGEMENT SOFTWARE PRODUCTS

VAX Information Management software products (see Figure 1-12) are grouped into the following categories.

- Application and data management products
  - VAX CDD
  - VAX DBMS
  - VAX Rdb/VMS
  - VAX ACMS
- Runtime and forms management software products
  - VAX DATATRIEVE
  - VAX TDMS
  - VAX FMS

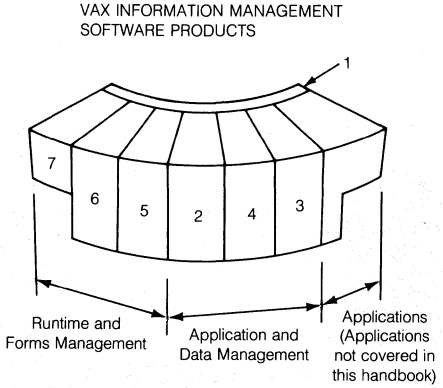


Figure 1-12 ▪ The VAX Information Management Software Products

These software products are all briefly discussed in the following section. For a more indepth discussion of these products and their relationship to the VAX Information Management Architecture see *The VAX Information Management Handbook.*

APPLICATION AND DATA MANAGEMENT SOFTWARE PRODUCTS - In addition to its standard file and record management services (RMS,) the VMS Program Development Environment offers programming users three optional facilities for data and database management. (See Figure 1-13.) These products can be used in conjunction with many VMS languages, development tools, and other VMS software products. These products are the VAX Common Data Dictionary (CDD), VAX Database Management System (DBMS), and the Relational Database System for the VMS operating system (Rdb/VMS).

VAX INFORMATION MANAGEMENT FACILITIES-APPLICATION AND DATA MANAGEMENT PRODUCTS

1. VAX Common Data Dictionary (CDD)
   - Stores data definitions for many VMS software products
   - Uses CDDL, a special language used to create data definitions
2. VAX Database Management System (DBMS)
   - Is used to create and maintain multiple databases
   - Can tailor a user's 'view' of data
   - Maintains data security and integrity
3. VAX Relational Database Management System (VAX Rdb/[VMS, ELN])
   (a) (b)
   - Provides facilities for creating, accessing, and maintaining relational databases
   - Dynamically establishes relationships between records
   - Tailors a user's view of data
   - Recovers from errors and/or inadvertent terminations
4. VAX Accounting Control and Management System (ACMS)
   - Provides tools for the development and control of complex online applications

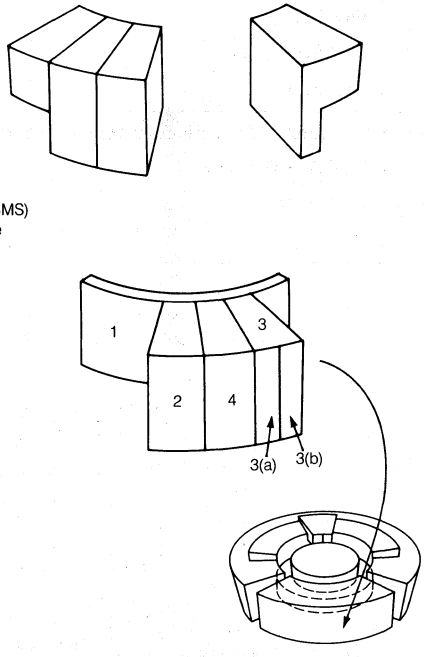


Figure 1-13 ▪ VAX Application and Data Management Products

The common benefit each of these products brings to the application development environment is data independence. Data independence means that it is easier to write applications that share data. You can, therefore, change a program's data definitions or storage locations without extensive rewrite of the program.

Data and database management facilities also provide comprehensive data security and integrity features that can be used to limit access to data, to ensure the accuracy and consistency of data, and to recover from hardware failures. Sophisticated techniques for modeling data structures and data relationships are additional benefits obtained when you use Digital's data and database management facilities.

With VAX ACMS you can manage complex, multiuser application systems. The typical VAX ACMS application involves simultaneous access to a common database by many users with little or no computer experience. Applications well suited to management by VAX ACMS include hotel reservations systems, personnel administration systems, transaction processing, and funds transfer systems.

VAX COMMON DATA DICTIONARY - The CDD (see Figure 1-13) is a central location for the storage of data definitions used by many VMS software products. The CDD is required when using VAX DATATRIEVE, VAX Rdb/VMS, VAX TDMS, and the VAX ACMS product set. The CDD can also be used in conjunction with several VAX programming languages, which access record definitions in the CDD at compile time. Using the CDD you can

- Create sharable definitions with a data definition language (CDDL) that can be understood by many VAX programming language compilers and several other VAX Information Architecture Products.
- Modify data definitions in the dictionary without editing the programs and procedures using the definitions.
- Specify which users have access to individual definitions, using thirteen separate access privileges and four possible user identification criteria.
- Copy definitions at compile time into a program written in one of the VAX programming languages.
- Document the use of a particular definition by making entries into the definitions' history list.
- Maintain an area of the dictionary for the storage of data definitions for privileged use.

VAX DATABASE MANAGEMENT SYSTEM (DBMS) - VAX DBMS (see Figure 1-13) provides sophisticated capabilities for creating, accessing, and maintaining large databases. The major advantages of VAX DBMS are its efficiency and its ability to control the sharing of the same data by a large number of users, all using the system at the same time.

VAX DBMS is also compatible with other VMS software products. You can access a VAX DBMS database from any application program written in a high-level language that conform to the VMS calling standard. This is accomplished through an interface to VAX DBMS, called Database Query (DBQ).

With VAX DBMS, you can

- Create and maintain multiple databases.
- Separate data definitions from the applications programs that use them.
- Centralize all data definitions from the applications programs that use them.
- Define useful relationships between records.
- Separate data definitions from data storage.
- Tailor user views of data.
- Centralize the administration of data.
- Maintain data integrity and security.
- Allow concurrent access to databases.

Who Uses VAX DBMS?

VAX DBMS is intended to serve as a data management system for organizations in which

- There is a large amount of data retrieval.
- There are many concurrent users.
- Predictable performance is required.
- Data relationships are complex.
- Database implementations are long term.
- A professional database designer/administrator is available.

VAX RELATIONAL DATABASE MANAGEMENT SYSTEM (VAX Rdb/VMS) - VAX Rdb/VMS (see Figure 1-13) provides facilities for creating, accessing, and maintaining relational databases. In a relational database, data is stored in the form of two-dimensional tables, instead of in the form of complex hierarchies or networks. VAX Rdb/VMS provides all the advantages of a full-feature database management system, including data security and optimized data access. At the same time, it provides easy-to-use features inherent in a relational-style database.

VAX Rdb/VMS is easy to use and understand. With it, you maintain and create a database without the services of a professional database administrator.

Two VAX Rdb products are currently available — VAX Rdb/VMS and VAX Rdb/ELN. VAX Rdb/VMS runs on the VMS operating system; VAX Rdb/ELN, on the VAXELN system. Aside from this basic difference, the two products are very similar.

With VAX Rdb/VMS you can

- Create and maintain relational databases.
- Store and retrieve data.
- Separate data definitions from the application programs that use them.
- Store all data definitions either in the database files or in the VAX Common Data Dictionary (CDD), for common use and maintenance.
- Establish dynamic relationships between records.
- Tailor user views of data.
- Centralize the administration of data.
- Maintain data integrity and security.
- Use a database concurrently with other users.
- Ensure against data inconsistency during concurrent updating.
- Recover from errors and/or inadvertent terminations.

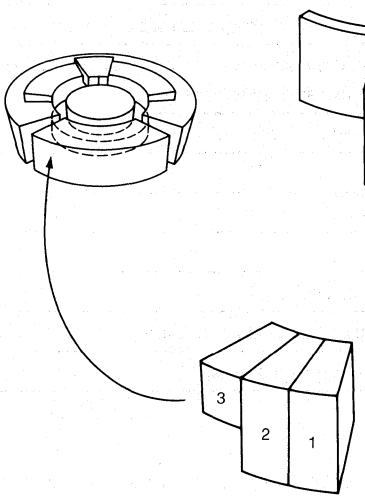
VAX ACMS PRODUCT SET - The VAX Application Control and Management System (ACMS) (see Figure 1-13) product set provides tools for the development and control of complex online applications. These tools can reduce application development and maintenance time replacing significant amounts of control and application code with definitions stored in the VAX Common Data Dictionary.

The VAX ACMS product set consists of ACMS/AD and ACMS. ACMS/AD gives programmers the tools necessary to design and maintain online applications. ACMS is used to create the operational environment they run in.

INFORMATION MANAGEMENT RUNTIME AND FORMS MANAGEMENT SOFTWARE PRODUCTS - VAX Information Management products offer users a number of software products that include

- VAX DATATRIEVE
- VAX TDMS
- VAX FMS

VMS INFORMATION MANAGEMENT RUNTIME AND
FORMS MANAGEMENT SOFTWARE PRODUCTS



1. VAX DATATRIEVE
   - Stores or modifies information in VAX RMS files, VAX DBMS database, or Rdb/VMS databases
   - Creates graphic displays, or formatted reports using specific selections of data
2. TDMS
   - Is used to create forms-intensive terminal application programs
   - Is a fourth-generation interface for defining the data exchange between a terminal's screen and an application program
   - Is used in conjunction with some VMS third-generation programming languages
3. FMS
   - A programming aid used to design application programs that use video forms

Figure 1-14 ▪ VAX Runtime and Forms Management Software Products

VAX DATATRIEVE - VAX DATATRIEVE (see Figure 1-14) is an easy-to-use tool for managing and manipulating data either interactively at a terminal or from an applications program. If you know as few as ten simple, English-like commands and statements, you can interactively retrieve, store, modify, and sort data, and report on it in meaningful ways. With VAX DATATRIEVE, you can

- Create data definitions that can be used to store and retrieve data uniformly, either interactively or from application programs
- Store or modify data in RMS files, VAX DBMS databases, or VAX Rdb/VMS databases.
- Retrieve data from RMS files, VAX DBMS databases, or VAX Rdb/VMS databases and display the data on a terminal, write it to a data file, or print it.

- Produce formatted reports using specified selections of data display or data collection.
- Create pie charts, bar and line graphs, and plots based on specified selections of data.
- Use forms to format the terminal screen for data display or data collecting.
- Use a text editor to correct syntax errors and typing mistakes.
- Access data files located on remote systems.
- Call VAX DATATRIEVE functions (including data access) from a program written in a high-level VAX programming language such as COBOL, FORTRAN, or PL/I.

Who Uses VAX DATATRIEVE?

VAX DATATRIEVE can be used in many different applications, by many different types of users. Some of the organizations using VAX DATATRIEVE are

- Managers who need ready access to different views of data for decision-making and/or monitoring purposes.
- Organizations that need a data-storage system that can be run by clerical personnel.
- Organizations that need to access data on a distributed network without being concerned where the data is actually located.
- Applications programmers who want to save coding/debugging time and source space by having VAX DATATRIEVE handle such functions as finding and opening data files, performing input and output operation, formatting data, converting data types, and handling error and end-of-file conditions.

With its set of simple, English-like statements, VAX DATATRIEVE is designed both for users with only modest computer experience and computer professionals. It includes a special tutorial facility called Guide to help the user get started with the facility.

VAX TDMS - Digital's VAX Terminal Data Management System (TDMS) (see Figure 1-14) is a productivity tool designed to reduce the high life-cycle costs of developing and maintaining forms-intensive terminal applications on VAX/VMS systems.

TDMS offers a wide range of features that make it easy to develop applications that display and collect information , relieving the programmer of many of the burdens associated with conventional forms-based application.

TDMS provides a fourth-generation interface for defining the data exchange between screen(s) and program(s). It is used in conjunction with standard third-generation languages such as VAX COBOL, VAX BASIC, or VAX FORTRAN for defining programming logic. It replaces the often cumbersome coding of program/screen interaction with definitions, which are stored independently in the VAX Common Data Dictionary.

Use of a record-level interface provides both terminal and data independence at the application program level because the application need only exchange record buffers with the interface — as simple as programming to a disk file. TDMS manages the moving, mapping, and data type conversion between the fill record data types and screen data types, freeing the applications of all such logic.

VAX FMS - The VAX Forms Management System (FMS) (see Figure 1-14) is designed to aid in the development of application programs that use video forms. FMS manages these forms for application programs that use Digital's family of VT100 and VT200 compatible terminals. Forms defined using VAX FMS provide a programmer with the ability to use the following features of those terminals

- Individual character attributes of reverse video, bold, blinking, and underline
- Line attributes of double-width, double-height, and scrolling
- Screenwide attributes such as 80- or 132-column lines and reverse video
- Alternate character sets including the VT100 special graphics character set for line drawing

- RELATED VAX SOFTWARE PRODUCTS

Related VAX software products include

- Digital networking and communications software products.
- Cross development tools for PDP-11 and VAX.

VAX NETWORKING AND COMMUNICATIONS SOFTWARE PRODUCTS

Digital's communications products (see Figure 1-15) make it possible for VAX systems to communicate with each other and with other manufacturers' products. Included in these products are DECnet-VAX and VAX PSI.

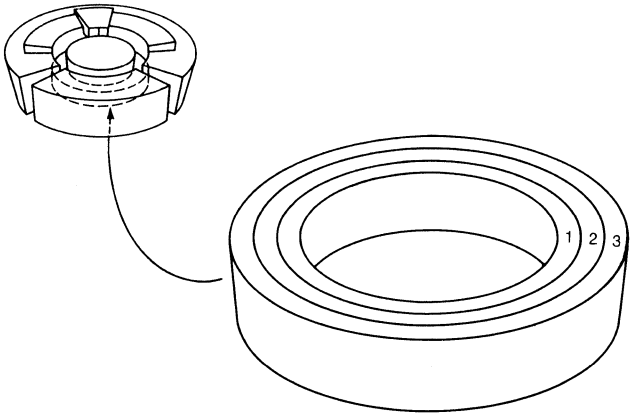


Figure 1-15 ▪ VAX Networking and Communications Software Products

A network is an entity of two or more computer systems, called nodes, connected, by microwave or satellites for the purpose of exchanging data and sharing resources.

Digital offers a range of products to link VAX/VMS systems, other Digital systems, or other vendors operating systems. VAX/VMS systems are connected to these systems in three ways: VAX to other Digital systems (DECnet-VAX software products), VAX to other manufacturers' systems (DECnet-VAX Internet software products), and VAX to public packet switched networks (VAX PSI software products).

Communication circuits operate over physical lines. A circuit is a communications data path. Communication between individual processes on different systems takes place through logical links. A logical link is a connection between two processes at the user level. A circuit can support many logical links, all communicating at the same time.

In a network of more than two nodes, directing data from one node to another is called routing. With DECnet, a very important feature DECnet network is that its nodes are permitted to route messages through the most cost-effective path. If a circuit is disabled, nodes can seek alternative routes to send the data through.

DECnet-VAX SOFTWARE PRODUCTS - DECnet-VAX (see Figure 1-15) is a group of software communications products that allows a properly configured VMS operating systems to participate in a network environment.

Phase IV DECnet supports large and small networks with single and multiple area networks. In a single area network, a maximum of 1023 nodes is possible. The optimum number, however, is much smaller — perhaps 200 to 300 nodes depending on the physical relationship of the nodes. In a multiple-area network, as many as 63 subnetworks (single-area networks, of 1023 nodes each) can be connected.

As a part of DECnet-VAX, DECnet's Internet communications software products enable VAX systems to communicate with non-Digital computing environments. For example, the VAX 3271 Protocol Emulator enables VT100 users on a VAX/VMS system to communicate interactively with applications running on an IBM 370 class system.

VAX-PSI - VAX-PSI communications software products (see Figure 1-15) enable suitably configured VMS systems to connect to and access industry-standard Packet Switching Data Networks (PSDNs). This product set consists of two options: full-function VAX PSI and a subset of that product, VAX PSI Access.

Full-function VAX PSI allows the use of DECnet-VAX facilities over X.25 circuits in addition to DECnet's support for Ethernet or point-to-point links. VAX PSI Access makes it possible for DECnet-VAX systems in a point-to-point or Ethernet DECnet environment to make logical connections to PSDNs. This enables process-to-process and terminal communications between the accessing VAX and remote Data Terminal Equipment (DTE).

A more indepth coverage of these products and VAX networking theory of operation can be found in chapter 6 of this handbook.

CROSS DEVELOPMENT TOOLS FOR PDP-11 and VAX SYSTEMS - Of prime importance to the continued success of your computing environment is being able to move your existing applications to new systems. The two layered products described in this section have been designed to move new and existing applications between VAX and PDP-11 systems — VAX-11 RSX — and to create statically defined systems on a VAX that will be run on a target VAX or MicroVAX system — VAXELN.

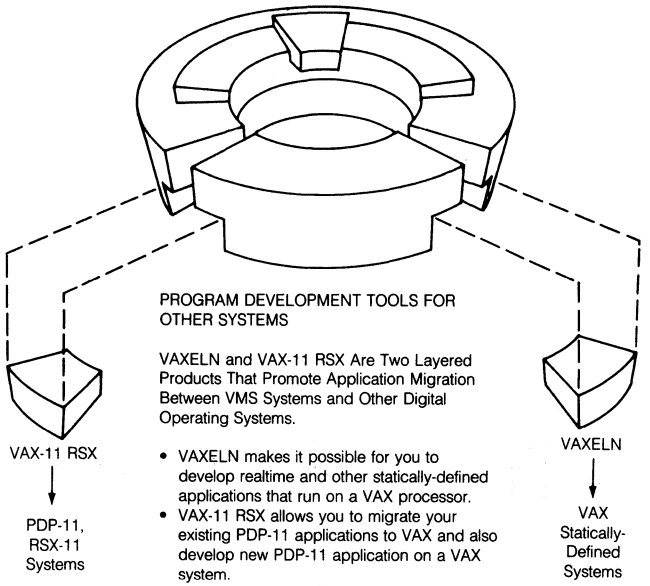


Figure 1-16 ▪ Program Development Tools for PDP-11 and MicroVAX Statically Defined Systems

VAX-11 RSX - A major feature of the VAX/VMS operating system is its compatibility with the PDP-11 family of minicomputers.

When using VAX-11 RSX (see Figure 1-16), programmers familiar with the PDP-11 program development environment will find a great deal of similarity between the instruction sets and languages of both systems.

The VMS operating system may serve effectively as a high-performance RSX-11M/S program development system. RSX-11M/S programs can be edited, compiled, and linked on a VAX system. In addition, the program can be partially debugged on a VAX system. That is, the software development can largely be accomplished on a VAX system and need only migrate to the target RSX-11M/S system for final debugging and execution.

The VMS operating system, through DECnet communications software, supports downline loading of RSX-11S systems and RSX-11Mtasks. However, the VMS system and the RSX-11M/S system must have either a common communications link or a mass storage peripheral of the same type on both systems in order to transfer the RSX-11M task or RSX-11S system to the target machine.

Under the VMS operating system, programs may execute in either of two modes — native or compatibility. Native-mode programs use the VAX instruction set and execute under the VMS operating system. Compatibility-mode programs, however, are those that can execute on other PDP-11 systems.

In order to provide cross-development and migration capability, an RSX-11M Applications Migration Executive has been implemented that allows most non-privileged RSX-11 tasks to execute on the VAX/VMS system with little or no modification of the image.

VAXELN - Designing and developing realtime application have always been a demanding task, even for the most adept programmer. Implementing sophisticated multitasking programs at the assembly language level is tedious and exacting. But traditionally, that's been the only way to get the best in realtime performance.

The VMS productivity environment offers the realtime programmers an alternative to this complex productivity problem with the VAXELN toolkit. VAXELN lets you program in an extended version of PASCAL, with all of its high-level, easy programming features.

The VAXELN toolkit is a layered product that supports the development of standalone, statically defined software systems (VAXELN systems) that run on the entire range of VAX processors, including the MicroVAX microcomputer. VAXELN systems are developed under VMS, then run on the target system as a standalone system, without VMS present.

Typical applications include industrial automation, workstations, Ethernet server networks, and any others networks in which individual processors have dedicated functions and are not needed simultaneously for general computing, for which a general operating system, like VMS, is not necessary.

Because a VAXELN system is a subset of the VMS operating system, it contains the logic and services needed to perform its function in a realtime environment.

VAXELN simplifies the design and implementation of realtime applications by letting programmers work in the high-level language, PASCAL. VAXELN also uses only conceptually simple, object-oriented operating system code (the kernel), and includes only the pertinent operating system services, and device drivers that implement a file system, network communication facilities, and I/O device handling via the PASCAL Language.

For more information on VAX-11 RSX and VAXELN see Chapter 7 of the *VAX/VMS Languages and Tools Handbook.*

Chapter 2 • DCL (The Digital Command Language)

• Introduction

The Digital Command Language (DCL) is the language — interface — through which you communicate with the VMS operating system. DCL contain an extensive set of commands that allow you to do tasks such as

- Develop and execute programs
- Device and data file manipulation
- Interactive and batch program execution and control
- Operational control

Commands exist for program development and execution, for resource allocation, environmental control, job control, file maintenance, utilities, and operational control.

- Program development and execution commands include commands to invoke each compiler, the assembler, the editors, and the linker, as well as to run any prelinked program.
- Resource allocation commands include the ability to allocate and deallocate devices and mount and dismount volumes.
- Environmental commands include assign and deassign logical names and set and show parameters such as job status, terminal type, and default directory.
- Job control commands include the ability to continue and stop execution, a GOTO command to transfer control, and IF and ON commands to specify error handling.

DCL also includes commands to login and logout, to submit batch jobs, to send messages to the operator, and to prompt the user for input. File maintenance commands include append to files, copy, create, and delete files, list directories, initialize volumes, print and type files, and rename files.

Command Format

DCL commands are composed of English words. Any file name can be given a logical name for mnemonic reference. Command parameters can be supplied on the same line as the command verb. Missing parameters will be prompted for by the VMS command interpreter. To make it easier to learn the VMS system, an extensive HELP facility gives guidance on the use of commands and the meaning of system messages.

Typical VMS commands are brief because of the extensive use of defaults. The user also can define additional commands and use them just as the system-defined commands are used. All command verbs and qualifiers can be abbreviated to the shortest unique form.

File specifications can be as simple as the user-given name of the file only, or as complex as a full specification of network node, device (including type, controller, and unit), directory, file name, file type, and version number. Logical names can be defined for complex file specifications so that repetitive typing can be avoided.

The general format of a command is:

```
$command-name[qualifiers][parameter-1]...[parameter-n]
```

where the following rules apply:

- Dollar sign ($) — The dollar sign [$] must appear in position 1 of a command to be executed in a command procedure. Optionally, it may appear in a command executed in interactive mode.

- Brackets — In the description of commands in this specification, brackets ([and]) are used to surround optional values. For example:

  ```
  COPY[qualifiers]
  ```

  indicates that the user does not need to supply any qualifiers to issue a valid COPY command.

- Labels — Any command may be labeled. Labels are used to transfer flow of control via the GOTO command. They can also be used for documentation purposes. The maximum length of a label is 15 characters. A label precedes the command name and is separated from it by a colon (:).

- Command names — The command name indicates the action the command is to perform.

- Qualifiers — A qualifier is used to modify the default action of a command. There are defaults for all qualifiers, so qualifiers are never required. A qualifier always begins with a slash (/). Both command names and parameters can be qualified.

Examples:

```
PRINT/DELETE  MYFILE.DAT
SET TERMINAL/LOWERCASE
```

Many qualifiers have associated qualifier values. The qualifier is separate from the qualifier value by an equal sign (=) or a colon (:), that is, /COPIES=3. Whenever a qualifier requires a list of values, that list must be enclosed in parentheses:

```
/BLOCK=(5,6)
```

A qualifier can not contain any blanks; however, blanks are allowed in qualifier values following the left parenthesis, preceding the right parenthesis, if enclosed in quotes,and before or after a comma. No other blanks are permitted in qualifier values.

Some qualifiers can be negated. When this is permitted, the word NO prefix the qualifier name.

Example:

```
/OBJECT   :produce an object file
/NOOBJECT :do not produce an object file
```

- Parameters — A parameter either specifies a value that a command is to use when executing or further defines the action a command is to take. At least one space or tab must separate the first parameter from the command name; parameters are then separated from each other by one or more spaces and/or tabs. Interactive users may supply parameters in response to prompts.

- Commas and ellipsis — Some commands permit the user to replace a single parameter by a list of values. When this is done, the items in the list are separated by commas. The commas can optionally, be surrounded by blanks.

  Examples:

  ```
  DELETE  A,B,C

  Delete files A, B, and C.

  COPY  A,B  C

  Copy files A and B into C.
  ```

  In the description of a command's format, ellipsis (three dots ...) indicate that a list of values of the same type may replace a single value.

- Continuation character — A hyphen (-), which may optionally be followed by blanks and/or a comment, is used to indicate that a command is to be continued on the next line.

  Example:

```
COPY A.DAT -
B.DAT
```

- Comment character — An exclamation mark (!) delimits the start of a comment. The comment includes everything from the exclamation mark to the EOL. The EOL is the comment terminator. Comments can occur only after the last character of a command or after a hyphen. Comments are for the user's information only and do not affect the processing of the command.

  Example:

```
COPY A.DAT B.DAT
!FILE A TO FILE B
!COMMAND PROCEDURE FOLLOWS
```

- Concatenation character — A plus sign (+) indicates concatenation, that is, the records in the file specified on the left of the plus sign are processed followed by the records in the file specified on the right of the plus sign.

  Example:

```
FORTRAN  A+B
```

  The FORTRAN statements in file A.FOR followed by the FORTRAN statements in file B.FOR are read by the FORTRAN compiler to produce a single object module, A.OBJ.

- Lowercase Characters — Lowercase characters will be processed as their upper case equivalents except for characters within a quoted string. The SET TERMINAL/[NO]LOWER command controls conversion of characters entered interactively at the terminal; however, it has no effect on data entered via a command procedure.

- Abbreviation rule — All command names, qualifiers and parameter keywords can always be abbreviated to the first four letters. The implementation will recognize, in each case, the minimal unique abbreviation. Qualifiers and keywords must be unique only within the command containing them. Additional letters are acceptable, for example, LOGOUT, LOGOU, and LOGO are all correct.

- End of Data — In interactive mode, CTRL/Z is used to terminate input to a command or a user program, for example, CTRL/Z will generate an end-of-file.

Conventions For Language-Name Commands

- When the input file specification in a language-name command consists of a list of concatenated files, that is, A + B + C, then the language processor is invoked once and a single object file is produced. If this object file is not explicitly named, the left most file specification will be used for the default. (Note that not all language processors permit the specification of a concatenated list.)

- When the input file specification in a language-name command consists of a list of file specifications separated by commas — that is, A, B, C — then the language processor is invoked separately for each file specification and a separate object file is produced for each. If the object files are not explicitly named, the name of the corresponding input file specification is used for the default. A qualifier on a file specification overrides a corresponding qualifier on the command name for that file specification.

  Example:

```
FORTRAN/LIST  A, B/NOLIST, C
```

- In interactive mode, /OBJECT, for example, produces an object file, and /NOLIST are the defaults. These defaults are also used when a command procedure file is invoked from interactive mode. In batch mode the defaults are /OBJECT and /LIST.

Command Procedures

A command procedure is a file that contains a list of DCL commands. When you execute a command procedure, the DCL command interpreter reads the file and executes the commands in it.

Command procedures can be used to automate a sequence of commands that you use frequently. For example, if you always issue a DIRECTORY command after you move to a subdirectory, you can write a simple command procedure to issue the SET DEFAULT and DIRECTORY commands for you as illustrated in the following example.

```
$ SET DEFAULT [SMITH.ACCOUNTS]
$ DIRECTORY
```

Instead of issuing each command, you could greatly simplify the operation and reduce the number of keystrokes by using a command procedure (named GO_DIR.COM) that would be executed when you type:

```
$ @ GO_DIR
```

This command tells the DCL command interpreter to read the file GO_DIR.COM and to execute the commands in the file. Therefore, the command interpreter sets your default directory to [SMITH.ACCOUNTS] and issues the directory command.

You can write complex command procedures that resemble programs written in high-level programming languages. In this sense, a command procedure provides a method of writing programs in the Digital Command Language.

You can, for example, revise GO_DIR.COM to allow you to move to any directory and obtain a list of the files in the directory and obtain a list of the files in the directory:

```
$ INQUIRE DIR_NAME "Directory name:
$ SET DEFAULT 'DIR_NAME'
$ DIRECTORY 'DIR_NAME'
```

When you execute GO_DIR.COM, the INQUIRE command prompts for a directory name, the SET DEFAULT command move you to the directory, and the DIRECTORY command lists the file names.

Formatting Command Procedures

Use a text editor (or the DCL command CREATE) to create and format a command procedure. When you name the command procedure, use the default file type COM. If you use this default file type, you do not have to include a file type when you execute the procedure with the "@" command.

Command procedures contain DCL commands that you want the DCL command interpreter to execute and data lines that are used by these commands. Commands must begin with a dollar sign ($). You can start the command string immediately after the dollar sign, or you can place one or more spaces or tabs before the command sting to make it easier to read.

Data lines, unlike commands, do not begin with a dollar sign. Data lines are used as input data for commands (or images.) Data lines are used by the most recently issued command; these lines are not processed by the DCL interpreter.

The following example illustrates command lines in a command procedure.

```
$ MAIL
SEND
THOMAS
MY MENU
Do you have a few minutes to talk about the ideas
I presented in my memo?
$
$ SHOW USERS THOMAS
```

In the preceding example, the first line is a command and must start with a dollar sign. The next lines are data lines that are used by the MAIL utility; these lines must not start with a dollar sign. Note that data lines must correspond to the way the image (invoked by the command) expects the data. Therefore, the data lines provide a MAIL command (SEND,) a recipient (THOMAS,) a subject (MY MEMO) and the text of of the mail message. When the command interpreter finds a new line that begins with a dollar sign, the the MAIL utility is terminated.

Terminal Function Keys

Table 2-1 ▪ Terminal Function Keys Used In Association With DCL

CR RETURN	Carriage return. Transmits the current line to the system for processing.
CTRL/X	Cancels type-ahead. Discards any characters that have been typed but not yet read by a program. It is identical to a CTRL/U.
CTRL/C	Before terminal session, initiates login sequence. During command entry, cancels command processing. Note: Certain system and user programs may provide special routines to handle CTRL/C interrupts. If CTRL/C is pressed to interrupt a program that does not handle CTRL/C, CTRL/C has the same effect as CTRL/Y and echoes as Y.
CTRL/I	Duplicates the function of the TAB key.
CTRL/K	Advances the current line to the next vertical tab stop.
CTRL/L	Form feed.
CTRL/O	Alternately suppresses or continues display of data at the terminal. (But it does not suspend execution of an image or command procedure.)
CTRL/Q	Restarts terminal output that was suspended via CTRL/S.
CTRL/R	Retypes the current line during input and leaves the cursor positioned at the end of the line.
CTRL/S	Suspends terminal output until CTRL/Q is pressed. (It also suspends execution of image or command procedure.)
CTRL/U	Cancels the current line and discards it.
CTRL/Y	Interrupts commands or program execution and returns control to the command interpreter.

(continued on next page)

Table 2-1 ▪ Terminal Function Keys Used In Association With DCL (Cont.)

CTRL/Z	Terminates a file input from the terminal.
DELETE or RUBOUT	Deletes the last character entered at the terminal and backspaces over it.
ESCAPE or ALTMODE	Have uses pertinent to particular commands or programs.

The Commands

The following tables list DCL commands in functional categories. This is only a partial listing.

- General session information (Table 2-2)
- Batch and command procedure specific control (Table 2-3)
- Volume and device resource control (Table 2-4)
- File manipulation (Table 2-5)
- Program development and execution control (Table 2-6)

Table 2-2 ▪ DCL General Session Control Commands

Command	Description
LOGIN	The user initiates an interactive session with the system by typing CTRL/C, CTRL/Y, or by pressing the carriage return on a terminal not currently in use. The system then prompts for user name and password and validates them.
HELP	Displays information to assist the user in selecting the proper command syntax and qualifiers.

(continued on next page)

Table 2-2 ▪ DCL General Session Control Commands (Cont.)

Command	Description
SHOW	Displays any of the following information: – current day, date, & time – current default directory and directory name – status of devices in the system – logical device name assignments – current characteristics and status of specified mag tape device – name, number, and status of local network node, and list available remote nodes – default characteristics of the system printer – status of current process – current file protection to be applied to all new files created during terminal session or batch job – current file protection to be applied to all new files created during terminal session or batch job – current status of entries in printer/batch job queues – current disk quota – current RMS default multiblock and multibuffer counts – stature of currently executing image in process – current value of a local or global symbol – displays a list of processes and status information in the system – current characteristics of a specified terminal – logical name translation – display of working set quota and limit assigned to current process.

(continued on next page)

Table 2-2 ▪ DCL General Session Control Commands (Cont.)

Command	Description
SET	The DCL SET command – sets default card reader translation mode – controls whether command interpreter receives control when CTRL/Y is pressed – changes the user's default device name or directory name – defines default characteristics for specific magtape device – determines whether command interpreter performs error checking following execution of commands in command procedures – changes execution characteristics of currently executing process – invokes VMS COMMAND DEFINITION utility (CDU) to add commands – changes a file's protection – changes the characteristics of a file – changes current status of attributes of file queued for printing or for batch job execution defines default values for multiblock and multibuffer counts use by RMS – changes characteristics of a specified terminal – controls whether or not lines in command procedures are displayed at terminal or printed in batch job log – redefines default working set size for the current process.
ASSIGN	Assigns logical name to a given character string (equivalence name) and stores the pair of names in a process, group, or system logical name table.
DEFINE	Creates a logical name equivalence. (same as ASSIGN except for syntax.
DEASSIGN	Breaks the correspondence between a logical name and its equivalence name (see ASSIGN and ALLOCATE), or deletes a symbol (see DEFINE).
MCR	Signifies that the given command or following command lines are to be interpreted by the RSX-11 command interpreter.
PHONE	Invokes the VMS PHONE utility that allows users on one system to interactively communicate with users on any other VAX system connected by DECnet-VAX.

(continued on next page)

Table 2-2 ▪ DCL General Session Control Commands (Cont.)

Command	Description
MAIL	Invokes the mail utility that allows users to send, receive, file, forward, and reply to MAIL messages.
LOGOUT	Terminates an interactive session and releases all resources allocated to the user.
REQUEST	Displays a message at a system operator's terminal and optionally requests a reply.

Table 2-3 ▪ DCL Batch and Command Specific Control Commands

SUBMIT	Places a given batch command file or command procedure in a batch queue for processing.
$PASSWORD	Specifies the password associated with the user name specified on a JOB card for a batch job.
$JOB	Indicates the beginning of a batch command file and provides job control information (such as time limit).
$INQUIRE	Requests interactive assignment of a value to a symbol and assigns the symbol a name.
$GOTO	Transfers flow of control to a given labeled line.
$ON	Transfers flow of control to a given labeled line if an error of a given severity or greater is encountered at any time during command procedure processing.
$IF	Transfer flow of control to a given labeled line.
$EOD	Signifies the end of data in the input stream following a $DATA command.
$EXIT	Terminates the command procedure.
$EOJ	Marks the end of a batch job submitted through the system card reader. EOJ performs the same function as the LOGOUT command.
DECK	Marks the beginning of an input stream for a command or program.
	Command names preceded by a dollar sign ($) are meaningful only in a batch command file or command procedure. All other commands listed in this table can either be issued interactively or used in batch command file or command procedure.

Table 2-4 ▪ DCL Volume and Device Resource Control Commands

MOUNT	Requests the operator to make a volume available to the user and optionally associates a logical name with volume or volume set.
INITIALIZE	Writes a directory file and other volume structuring information on a disk or magnetic tape volume to prepare it for use.
DISMOUNT	Requests the operator to break the logical association of this device with the user's job.
ALLOCATE	Obtains exclusive ownership of a device and enables the user to assign a logical name to the device.
DEALLOCATE	Releases allocated devices.

Table 2-5 ▪ DCL File Manipulation Control Commands

DIRECTORY	Reports information (size, protection, ownership, creation time) on a given file or set of files.
CREATE	Creates a new file from data subsequently entered in the input stream (user at terminal or batch stream).
EDIT	Opens a text file and accepts commands to insert, delete, or modify data in the file.
DELETE	Deletes one or more files from a mass storage disk volume.
DELETE/ENTRY	Deletes one or more entries from a printer or batch job queue.
DELETE/SYMBOL	Deletes one or more symbol definitions from a local symbol table or from the global symbol table.
PURGE	Deletes all but the latest version of a given file or files, optionally keeping the latest two or more versions.
RENAME	Changes the name of one or more existing files.
COPY	Copies the contents of a file or files, creating another file or files.
APPEND	Concatenates the contents of sequential files to given sequential files.
DIFFERENCES	Compares two files and reports the differences between the two.

(continued on next page)

Table 2-5 ▪ DCL File Manipulation Control Commands (Cont.)

SORT	Creates a file by rearranging the records in a given file based on the contents of key fields within the records.
OPEN	Opens a file for reading or writing at the command level.
CLOSE	Closes a file that was opened for input or output with the open command and deassigns the logical name specified when the file was opened.
READ	Reads a single record from a specified input file and equates the record to a specified symbol name.
WRITE	Writes a record to a specified output file.
PRINT	Sends the contents of a given file or files to a spooled output device, such as a lineprinter.
TYPE	Displays the contents of a given file or files on the device identified by the logical name SYS$OUTPUT: (default generally the user's terminal).
DUMP	Produces a printed listing of the contents of a file, ignoring any print formatting characters that may appear in the records.
ANALYZE	– provides a description of the records comprising an object module library – provides a description of the contents of an image file or sharable image file – invokes the RMS utility to inspect and analyze the internal structure of an RMS file – invokes the System Dump Analyzer to examine a running system or to examine a specified dump file – invokes the VERIFY utility to check readability and validity of Files-11 disk volumes.

Table 2-6 ▪ DCL Program Development and Execution Control Commands

LIBRARY	Creates, deletes, or maintains libraries of object modules, sharable images, or macro source modules.
LINK	Links modules to produce images
RUN	Starts interactive debugging session after interrupting program image in that process context.
DEBUG	Starts interactive debugging session after interrupting program image execution by typing a CTRL/C or a CTRL/Y.
EXAMINE	Displays the contents of a location in virtual memory.
DEPOSIT	Replaces the contents of a location in virtual memory with the given data.
CONTINUE	Resumes execution of a program interrupted by typing a CTRL/Y or CTRL/C.
SPAWN	Creates a subprocess of the current process. Portions of the current process context are copied to the subprocess
STOP	Terminates the program currently interrupted by a CTRL/C or a CTRL/Y.
SUBMIT	Enters a command procedure in the batch job queue.
SYNCHRONIZE	Places the process executing a command procedure in a wait state until a specified batch job completes execution.
WAIT	Places the current process in a wait state until a specified period of time has elapsed.
CANCEL	Cancels scheduled wakeup requests for a specific process. This includes wakeups scheduled with the run command and with the schedule wakeup ($SCHWDK) system service.

Chapter 3 ▪ System Security

▪ Overview

One of the negative aspects of the enormous growth in computer-based information systems over the past decade is the rise of business crime involving those systems. The only way to protect your system's integrity is to understand the security safeguards that are an inherent part on the VMS operating environment. This chapter is a general overview of VAX/VMS security features.

Topics covered are:

- Your security environment
- VMS security features for the system manager
- Security considerations for the system user

▪ Introduction

Like any other corporate asset, the information in your organization's databases must be protected from would-be computer criminals. These range in expertise and intent from an academic type, out to challenge the intellect of a computer, to the sophisticated white-collar criminal who attempts to defraud your corporation of millions of dollars.

The responsibility for protecting a corporation's informational assets rests squarely on your shoulders and your those of your system manager. Both these individuals have specific security responsibilities and are ultimately responsible for breaches in the integrity of their computing environment. The system manager is responsible for the physical aspects of a computer's security; the MIS manager is responsible for the overall integrity of system use.

VMS security features give you many alternatives in securing a computing environment. You can build the security environment that's right for you and your organization and that reflects your security philosophy.

The Security/System Manager
Security managers require accounts with privileges. In many cases the security manager role and system manager role will be performed by the same individual, so the same account can serve both purposes. The *Guide To System Management and Daily Operations* describes the necessary characteristics of a system management account. The same principles apply to the security management role. It is important that strong cooperation and open communication exist when the security management and system management roles are performed by separate individuals. One way to do this is to share an account. If there are reasons why this is not desirable at your site, set up separate accounts. The security manager will require the additional security privileges.

▪ Your Security Environment

Your security environment is composed of three elements:

- Personnel
- Facilities
- Operating System

Personnel and Facilities
Although personnel and facility factors are outside of the operating system security domain, their safeguarding still requires a great deal of attention by the system manager. Managers must make personnel understand the high priority of security. Security managers must also focus their attention on keeping the facility secure from intrusion by unwanted outsiders as well as from access by insiders.

PERSONNEL - Security breaches on most computer systems are categorized into three areas:

- User irresponsibility
- User probing
- User penetration.

USER IRRESPONSIBILITY - User irresponsibility refers to situations in which the user does the wrong thing and causes some noticeable damage. Perhaps the user accidentally deletes another user's files, misplaces storage media, or fails to adequately protect important files. The user may be unaware of proper procedures or just negligent through laziness or sloppiness. Typically, the action is not malicious, but the results can be nonetheless damaging.

Unfortunately, there is little that an operating system such as VMS can do to protect sites from this source of security breach. The problem typically lies either in the application design or in sloppy or inconsistent use of available controls by the user.

USER PROBING - User probing refers to situations in which a user exploits insufficiently protected parts of the system. The user who succeeds in probing is usually somewhat knowledgeable, and is generally driven by intellectual challenge, or just plain curiosity. These crimes often start as larks; users just want to test their prowess against the system to possibly gain some forbidden access or machine time.

VMS provides many security features that can be implemented to combat user probing. The features are applied according to the current needs at the site as judged by the security manager. Some of the features are best implemented on a temporary basis, while others, if needed at all, are needed permanently.

USER PENETRATION - Penetration refers to situations in which the a skilled user succeeds in going through all controls and gains access to files which may contain proprietary, confidential or even secret information. When this type of penetration occurs, either the controls are insufficient, or they have been insufficiently applied. While considerable effort has been expended to make VMS operating system resistant to user penetration, some chance of this intrusion exists.

There is little doubt that the skilled user who succeeds in penetration is malicious. Thus, this is the most serious and potentially dangerous type of security breach, However, in a well-run operation, it is also the rarest form, because the skill levels and perseverance required from the perpetrator.

SITES - Sites should have an explicit security program that motivates people to protect information and also limits and controls the exposure of sensitive information. Sites must recognize that physical security requires both that management provides locks where appropriate and that all users actually keep items locked up as requested. The two aspects are clearly intertwined. You will never achieve any appreciable level of environmental security unless you address both aspects of the problem.

The security and system managers should be aware that a chain is broken at its weakest link. Typically, environmental weaknesses in security are the ones most readily found and exploited by would-be intruders. What's easier? Stealing a floppy disk or magtape from a work area that someone had carelessly left unattended, or going through the many time-consuming and elaborate procedures needed to break into system and obtain the same information?

It is fair to say that environmental weaknesses loom as the greatest source of security breaches.

Operating System

Just as the security and system manager maintain visual surveillance of the physical aspect of a computing environment, the VMS operating system provides security surveillance measures in certain key areas. It is designed to intercept certain information as it is processed and determine if it affects system security. For example, an insecure system might allow any user to login, while the most secure type of security system might require a very elaborate series of checks before permitting the user to login. An elaborate series of checks, in fact, typically involves the identification and authentication that associates a VMS user with a process.

There are three basic areas in which the operating system can control security. These are:

- Authorization
- Data processing
- Audit Of Operating System Activity

AUTHORIZATION - The authorization of powers and extent of user access is a central point at which the operating system can control security. Various schemes exist ranging from very straightforward automatic passing of innocuous powers to all users to very elaborate schemes that grant powers to the user on a need-to-know basis.

When granting authorization to the user, the operating system can control a user's access to certain objects in the system. For example, with sophisticated schemes, users are granted or denied access to files or devices, possibly even the way in which they can access each. (Include specific example here)

DATA PROCESSING - When processing data, the VMS operating system can automatically encode or decode data. VMS provides security managers with an optional encryption facility for this purpose.

The VMS operating systems can introduce security controls whenever it is called upon to access a file, volume, or any other system resource. VMS does, in fact, do just this through its standard protection system and its optional access control lists.

AUDIT OF OPERATING SYSTEM ACTIVITY - The operating system also monitors actions as it performs them. For elaborate security systems, VMS gives the security manager the option to request these events be audited and provide an alarm when the event is detected. Thus, irresponsible behavior, attacks, or other incidents can be detected while they are in progress.

▪ VMS Security Features For The System Manager

The actual implementation of an organization's security environment is the responsibility of the system/security manager. The VMS operating system and software products offer a wide range of security tools and facilities that help the manager fine-tune a security system and make it the best possible system for a particular organization.

The following section offers a general overview of some of the VMS security features and necessary steps the security/system manager must take to implement a successful security environment.

Topics covered here are:

- Establishing user accounts
- Authorizing usage
- Protecting information

Establishing User Accounts

As a security manager, some of the functions you will be performing most frequently are interfacing with the users, training them, assigning their initial passwords, and modifying their accounts to add or remove restrictions.

The primary tool for all these activities is the AUTHORIZE utility. Each security manager must look carefully at the site's operation to decide how to set up user accounts. While you probably can develop one or more of your own templates for creating for many of your users, don't over simplify the process creating accounts to the point where you simply apply a template. The danger in relying solely on templates is overlooking special considerations that apply to individual users, so that you forfeit the kinds of control that only you can exercise.

If you do use templates, plan to reexamine them often to ensure they are validity and reflect the way you want your operation to run.

INTRODUCTION TO GROUP DESIGN - As you contemplate the placement of users in groups, remember that the groups you compose will have impact on file protection and will influence the assignment of group privilege. Sometimes it is helpful to first map out functions you expect users to perform. Look for similar groups of users involved in a common function.

As you perform this exercise, think ahead, too, to what you may know about future plans in your company. Try to incorporate these ideas into your strategy. This is not an easy job, but it is worth the effort you put into it. You can fine-tune the groups at any time, but the most important step you will take is to gain perspective on the logical groupings according to the functions users perform.

The following example will illustrate many of the principles of group design. The Rainbows-4-U Paint Company is a distribution company with five different departments: executive, accounting, marketing, shipping, and secretarial. Some of the employees in those departments who need computer resources have the following job responsibilities.

Table 3-1 ▪ Principles Of Group Design

Department	Employee	Function
Executive	Sunny Gold	President
	Olive Green	Treasurer (head of computer operations)
Accounting	Lily White	Payroll
	Rick Silver	Bookkeeping
	Violet Indigo	Clerk
	Ruby Crimson	Clerk
Marketing	Rusty Brown	Forecasting
	Teddy Copper	Sales Reporting
Shipping	Bill Blue	Inventory Control
Secretarial	Misty Gray	Correspondence Management, Pay-check printing

As the system manager of Rainbow Paint's computer resources, Olive Green will set up UIC groups based on the existing organizational structure. For example, the employees in the accounting department (L. White, R. Silver, V. Indigo, and R. Crimson) will be members of the Accounting UIC, Accounting. Setting up the UIC group in this way ensures user L. White easy access to data from user R. Silver, and so forth.

At Rainbow Paint, as in any organization, everyone in the company should not have access to all the company's data. For example, one of the functions of the accounting department is maintaining the payroll. Because payroll information is confidential, the employees in the shipping and marketing areas should not have access to that information.

So, as the system manager, Olive must also consider how to protect confidential information. By setting up the UIC groups according to the departmental organization, she can also protect the data according to those groups. The UIC groups Marketing and Shipping can be denied access to specific data and users in the accounting department UIC, Accounting, for example.

In placing users into UIC groups, Olive will use these two guidelines:

- Users who typically share data and/or control of one another's processes should be arranged in the same group.
- Users who should not have access to each other's data or who should control each other's processes should be assigned to different groups.

As the system/security manager of Rainbow Paints' computer resources, Olive decides to set up the UIC groups Accounting, Executive, Marketing, Shipping, and Secretarial. However, when she considers how payroll is done at the company, she foresees certain problems. As the Company Treasurer, Olive herself must access to accounting data. The president of the company, Sunny Gold, also wants access to that information. If Olive allows world access to the accounting data on the system, then all the users can access it, effectively destroying the protection. Thus, in this particular case UIC-based protection does not represent the best possible solution.

INTRODUCTION TO ACL DESIGN - While considering this security problem further, Olive realizes that Misty Gray in the payroll group needs access to payroll information so she can prepare the checks. A further complication is that the clerks in the accounting department, Violet and Ruby, should not have access to the more confidential pieces of payroll data. Because of all these considerations, UIC-bases protection does not present the best possible protection.

To summarize, the following table defines the access requirements for one object involved in the payroll process, the file PAYROLL.DAT. PAYROLL.DAT needs to be accessed by the following users in the following ways:

Table 3-2 ▪ Payroll Application

User	UIC Group	Access Need
SGOLD	EXEC	R
OGREEN	EXEC	RWED
LWHITE	ACCOUNT	RWED
RSILVER	ACCOUNT	RWED
VINDIGO	ACCOUNT	NONE
RCRIMSON	ACCOUNT	NONE
RBROWN	MARKETING	NONE
TCOPPER	MARKETING	NONE
CBLACK	SHIPPING	NONE
MGRAY	SECRETARIAL	R

If we extend this problem to a few other objects such as the order processing database and the accounts receivable program, you quickly realize it is impossible to structure UIC groups to afford the protection needed.

INTRODUCTION TO IDENTIFIER DESIGN - Rather than attempt to restructure the UIC groups around this situation, Olive decides to achieve her goals by using access control lists on the files. For example, consider the ACL that Olive might construct for just PAYROLL.DAT:

```
(IDENTIFIER=OGREEN,ACCESS=READ_WRITE_EXECUTE_DELETE)
(IDENTIFIER=LWHITE,ACCESS=READ_WRITE_EXECUTE_DELETE)
(IDENTIFIER=RSILVER,ACCESS=READ_WRITE_EXECUTE_DELETE)
(IDENTIFIER=MGRAY,ACCESS=READ)
(IDENTIFIER=SGOLD,ACCESS=READ)
```

Notice that many of the users share the same access needs. There is an alternative that will shorten the ACL. Olive could use the Authorize Utility to define a general identifier PAYROLL in the rights database. The holders of the identifier could be all the users who need RWED access to PAYROLL.DAT. Once the identifier and its holders are defined, Olive could then use the following simpler access control list (ACL) to specify the same type of access she previously allowed for PAYROLL.DAT.

```
(IDENTIFIER=PAYROLL,ACCESS=READ_WRITE=EXECUTE=DELETE)
(IDENTIFIER=MGRAY,ACCESS=READ)
(IDENTIFIER=SGOLD,ACCESS=READ)
```

Notice that another advantage of this approach is that even if an employee leaves a position, Olive does not have to change every ACL across the system she removes the users UAF record; as a consequence, that user also no longer holds the identifier.

The larger the number of users, the greater the benefit of using this type of solution. Consider the typical situation where there are many more users and more objects to protect. Then, the flexibility and the advantages of using identifiers becomes more apparent. The most important aspect of ACLs is that they help you establish an overall scheme that considers the types of files on your system and the protection needs of each.

If you have done a good job of designating groups and identifiers, you should be able to design ACLs and define standard protection with minimal pain. Time spent clarifying the common access needs of your users pays dividends in simplifying the design of identifiers and ACLs. You will also provide a framework that users will appreciate for their own designs of ACLs for the files they control.

There are some disadvantages to ACLs. They can consume amounts of paged pool when files are open. The also require additional processing time. Thus, you should not use them indiscriminately. They are best applied where protection is really needed.

Authorizing Usage

As you grant system privileges to users, you must determine each user's needs and grant privileges commensurate to those needs. Typically, you must decide to what extent you should:

- Restrict devices
- Restrict work times
- Restrict modes of operation

RESTRICTING DEVICES - There are various ways you can restrict the number of devices at a user's disposal. You may want to limit the number of devices a particular user has access to, or you may just want to limit amount of usage.

RESTRICTING TERMINAL USAGE - Terminals are normally set up to be accessible to SYSTEM only, via the SYSGEN parameters TTY_DEFPROT and TTY_OWNER. To make terminals accessible to certain users as applications terminals, you change their protection and/or their owner UIC.

The imposing of system passwords will limit the use of those terminals to users who know the system password. You can also incorporate SET PROTECTION/DEVICE commands for specific terminals in the command procedure SYS$SYLOGIN.

RESTRICTING DISK VOLUMES - Another important method of restricting a user's ability to use the system is through limiting the number of blocks of a user has access to a specific disk. This is accomplished by limiting device and directory quotas in the UAF.

OTHER RESTRICTIONS - You can use the DCL command SET PROTECTION/DEVICE to limit the access to any non-file-structured device. You might also apply an access control list on the device to refine the limits on the user's access.

RESTRICTING WORK TIMES - If you decide you cannot allow users to login during certain times of the day, you can limit login times by using AUTHORIZE. You can define primary and secondary days of the week with the /PRIMEDAYS qualifier, unless you conform to the default, which is primary days defined as Saturday and Sunday. For example, if a user works Tuesday through Saturday, you would specify the /PRIMEDAYS qualifier as follows:

```
/PRIMEDAYS=(NOMON,TUES,WED,THUR,FRI,SAT,NOSUN)
```

Generally, you must first decide which types of access you want to restrict to certain hours, because you specify the work times with the qualifier that describes the mode of operation. Your choices are /LOCAL, /REMOTE, /DIALUP, /INTERACTIVE, /BATCH, and /NETWORK. However, if your site will apply one set of primary and secondary hours for all types of access, your problems become simpler. You can just specify one qualifier, /ACCESS; the hours you include will apply to all modes.

RESTRICTING THE MODE OF OPERATION - Sometimes it is very helpful to restrict users to certain modes of operations. For example, the following concerns might prompt you to prohibit access via the network for some users:

- The user has data that should only be accessed through the local node.
- Penetration attempts are more likely to occur via a network due to the increased anonymity of the connection. (This concern is equally relevant to the dialup connections as well.)

Similarly, you may want to limit the submission of patch jobs. Whatever your reason, use the AUTHORIZE qualifiers pertaining to access modes to impose the restrictions.

Other Forms of Restriction

RESTRICTING COMMAND USAGE - There are several methods you can apply to affect the command usage by your users.

RESTRICTING ACCOUNT DURATION - It is a good practice to set an expiration time for an account that matches the maximum length of time you expect the user to require the access.

GRANTING USER PRIVILEGES - Privileges restrict the performance of certain system activities to certain users. These restrictions protect the integrity of the operating system's performance and thus the integrity of service provided to users. You should use the following two factors to grant privileges to each user:

- Whether the user has the skill and experience to use the privilege without disrupting the system and ,
- Whether the user has a legitimate need for the privilege.

Privileges fall into seven categories:

- None — no privileges
- Normal — Minimum privileges to effectively use the system
- Group — Potential to interfere with members of the same group
- Devour — Potential to consume noncritical system-wide resources
- System — Potential to interfere with normal system operations
- File — Potential to compromise file security
- All — Potential to control the system

A user cannot execute an image that requires a privilege the user does not possess unless the image is installed as a known image with the privilege in question. You should install user images with amplified privileges only after ensuring that the user needs the access and is unlikely to misuse it.

Table 3-3 ▪ VMS Privileges

Category	Privilege	Activity Permitted
None	None	None requiring privileges
Normal	MOUNT	Execute mount volume QIO
	NETBMX	Create network connection
	TMPMBX	Create temp. mailbox
Group	GROUP	Control processes in the same group
	GRPPRV	Group Access via SYSTEM protection field
Devour	ACNT	Disable accounting
	ALLSPOOL	Allocate spooled devices
	BUGCHK	Make bugcheck error log entries
	EXQUOTA	Exceed disk quotas
	GRPNAM	Insert group logical names in the name table
	PRMCEB	Create/delete permanent common event flag clusters
	PRMGBL	Create permanent global sections
	PRMMBX	Create permanent mailboxes
	SHMEM	Create/delete structures in shared memory

(continued on next page)

Table 3-3 ▪ VMS Privileges (Cont.)

Category	Privilege	Activity Permitted
System	ALTPRI	Set base priority higher than allotment
	OPER	Perform operator functions
	PSWAPM	Change process swap mode
	WORLD	Control any process
	SECURITY	Perform security-related functions
	SHARE	Access devices allocated to other users
	SYSLCK	Lock system-wide resources
Files	DIAGNOSE	Diagnose devices
	SYSGBL	Create system-wide global sections
	VOLPRO	Override protection
All	BYPASS	Disregard protection
	CMEXEC	Change to executive mode
	CMKRNL	Change to kernel mode
	DETACH	Create detached processes to arbitrary UIC
	LOGIO	Issue logical I/O requests
	PFNMAP	Map to specific physical pages
	PHY_IO	Issue physical I/O requests
	READALL	Possess Read Access to everything
	SETPRV	Enable any privilege
	SYSNAM	Insert system logical names in the name table
	SYSPRV	Access objects via SYSTEM protection field

Protecting Information

Once you have some confidence in your ability to create a workable security system that allows only desirable users access to the system commands, you must then address just which files you want these users to access. On every system there are files that require protection.

Your tools for protecting information include the DCL commands SET PROTECTION, SET UIC, SET OWNER, SET FILE, EDIT/ACL, SET ACL/ACL, SHOW ACL, and SET DEVICE/ACL.

FILE ENCRYPTION - File Encryption is the process of applying an algorithm to data to concealing its content. Decryption is the procedure that reverses the operation and converts encoded information back to its original content. A good application for such a feature is if you need to copy proprietary software onto some media for removal to another site, If the copied encrypted software fell into the wrong hands in transit, it could not be used unless the new owner possessed the correct key to decrypt it.

▪ Security Considerations For The System User

System users are the first line of defense against unauthorized access to your computing environment. It is your job to make users aware of the security threat and to instill a sense of responsibility for safeguarding the various aspects of the security environment they have control over.

The aspects covered in this section are:

- Logging on and off the system
- The proper use of passwords
- The protection of user files

Logging On and Off the System

A user must follow a sequence of actions to gain access to a system. This sequence is known as Logging in. Login is the first opportunity the system has to check users who request system access to ensure they are legitimate users. VMS users must identify themselves with a user name and password to prove they are authorized to use the system.

If proper procedures are not followed when logging off the system, users can compromise system security. As security manager you must ensure that users are taught correct logout procedures.

LOGINS - Login is also the system's first chance to impose restrictions, if desired, on the use of the system. The greater the security requirements at a site, the more elaborate the login procedure must be, and the more restrictions the user may face.

MESSAGES - The messages that are displayed after a user has successfully logged in could be of use to a would-be penetrator. Therefore, you as security manager may prefer to suppress the announcement message and welcome message so that the details of the node or operating system are not immediately revealed. This makes it a little more difficult for an outsider to know the proper login procedure, since login procedures typically vary from operating system to operating system. This is an action taken more frequently at sites that have high-level security concerns.

The same site that suppresses the welcome and announcement messages for security reasons is likely to choose to display the last-login-success-and-failure messages. The messages are valuable because authorized users can regularly check for unexplained logins and unsuccessful login attempts. These messages have another benefit: the appearance of such messages is disquieting to an illegal user because they show logins that are being monitored.

LOGGING OFF THE SYSTEM - As system/security manager you must establish some criteria for users to log off the system. They should know when and how to log off different input devices they are using.

When users leave their terminals on line and their offices open, they have effectively given away their password, privileges, and left their files and those of the group they are in unprotected. This vulnerable data can be copied or destroyed in a matter of seconds.

The system/security manager must continually remind system users to safeguard data, even in the simplest of ways.

LOGGING OUT WITH VIDEO TERMINALS - Each time users plan to log off the system from a video terminal, they should consider how much information can you afford to leave on the screen.

At low-security sites, there is seldom concern with what has been left on the screen. However, at sites with medium-level security concerns it may be desirable to leave nothing but the logout message on the screen. At high-security sites it is common practice to turn off the video terminals every time you log out.

LOGGING OUT FROM HARD-COPY TERMINALS - When users log out from a hard-copy terminal, their primary concern is to promptly and properly remove, file, or dispose of hard-copy output that might reveal security measures to a probing insider. (Note: security-conscious sites will have very few hard-copy terminals due to the security breaches possible if the output from the terminal falls into the wrong hands.)

As security manager, you should provide direction on the preferred procedures. Paper shredders or locked receptacles are often employed for this purpose.

LOGGING OUT OF DISCONNECTED PROCESSES - To logout of any disconnected process, users must issue the DCL command SHOW USERS to determine if they have other disconnected jobs. For each of your disconnected jobs, issue the DCL command CONNECT/CONTINUE to logout of the current process and connect back through each of the associated virtual terminals until they have reached the last existing process. However, a message is also displayed upon login asking you if you wish to connect to your other disconnected process.

Passwords

Systems commonly request passwords at login. Passwords are typically strings of characters that users can specify when they login to prove that they have been authorized to use the account. To preserve the secrecy of the passwords, terminals do not echo the password as it is not entered from the keyboard. Proper administration of passwords is critical to the security of a system.

List of Password Guide Lines — You can best protect your password by observing the following guidelines:

- Select reasonably long passwords that cannot be easily guessed. Avoid using words that would appear in a dictionary. Consider including a few digits in you passwords.
- Never post you password in your office or near your terminal.
- Never give your password to other users except under very unusual circumstances, and then be sure to change it immediately after the need for sharing has passed.
- Take special care users don't login on a terminal that they did not turn on. If necessary, take advantage of the VMS's secure terminal feature and instruct the user to press the break key before login.
- You can also restrict users to a given set of passwords.
- Insist that users change their passwords immediately if there is any reason to suspect they may have been discovered.
- Users should avoid using the same password for accounts on more than one systems.
- Users should never: walk away from a terminal without logging out or log into a terminal that is on.

The user may encounter various types of passwords on a VMS system. Most users will need to provide a user password when they log in. Other users will also need to provide a system password to gain access to a particular terminal before logging in with their user password. And still other users on systems with high security requirements will find themselves concerned with primary passwords and secondary passwords.

Users should be made aware of each type of password and its proper use.

SYSTEM PASSWORDS - System passwords control access to particular terminals. You can require system passwords when needed. Sometimes they are desirable simply as another level of security, but most often they are selected as a means of controlling access to terminals that might be targets for unauthorized use.

USER PASSWORDS - Typically, when users learn an account has been created on the system for them they are told whether or not a user password is required. If user passwords are in effect, users will likely be told to use a specific password for their first login. This password has been placed in a record in the User Authorization File (UAF) along with other pertinent information about their account.

SELECTING SECURE PASSWORDS - When selecting a password, user should avoid any combination of letters that form a word in any language. In that way, the password won't be discovered by a program that enters in succession the words in a dictionary, searching for the one that produces a successful login.

The content of a password is more important than the length. Using digits as well as letters provides the most secure passwords. Using letters only, there are 300 million combinations of six-character strings, for example.

AVOIDING PROGRAMS THAT STEAL PASSWORDS - Users should beware of using terminals that are already on. They might be revealing their password to a program that is specially designed to steal passwords. This precaution is particularly important when logging on a terminal in a public terminal room.

There are three precautions that can be taken to avoid these programs.

- Users should never directly login on a terminal that is already turned on. Advise them to press the (BREAK) key before pressing (RETURN). Pressing the (BREAK) key invokes the secure terminal server feature for the terminal, if you choose to enable this feature. The secure server ensures that the VMS login program is the only program able to receive their login.

- Second, users should never walk away from a terminal after they have logged in. They may return and be misled into thinking the system failed and then came back up again. In their absence, an inside penetrator could have loaded the password-stealing program into their area. Even a terminal that displays an apparently valid LOGOUT message may not reflect a process in the logged-out state that you would normally expect.

- And finally, make sure users check their last-login messages routinely. Remember, the password-stealing program cannot increase the login failure counts, even though it may portray its exit as a login failure to you. So, users should be particularly alert for login-failure counts that either do not appear following your failure, or that are one less than the number you experience. If you observe this, or any other failure trying to login, change your password immediately and notify your security manager.

Protecting User Files

The VMS operating system has many elaborate mechanisms for protecting files. These are extremely important tools for enhancing system security. They require a little work on the part of all users to master and use consistently, but they are worth the effort. The two primary protection mechanism are the standard user identification code (UIC)-based protection that controls access according to the user categories of System, Owner, Group, and World and Access Control Lists (ACLs).

All users will encounter the standard file protection mechanisms. Some users will also find themselves using the optional access control lists. Access control lists are important tools at sites with medium to high requirements for system security. ACLs are also important in environments with complex patterns of data sharing. As security requirements increase, so will the use of ACLs.

For more on UICs, see Chapter 4.

ACCESS CONTROL LISTS (ACLs) - ACLs can be used in conjunction with or as an alternative to UICs when protecting files. Generally, this method is used in conjunction with the standard UIC based protection, described above and in Chapter 4, as a way to fine-tune that protection where it is needed. This alternative offers a way to match the specific access to specific users for each object.

The system provides a rights database, a file that associates users of the systems with special names they are allowed to hold, called identifiers. Some of the identifiers represent the user's name and UICs. Other identifiers are more general names that many users will hold. The system/security manager will maintain this rights database, adding and removing identifiers as needs change. By allowing groups of users to hold identifiers, the manager has now created a different kind of group designation than one used with the users UIC. This alternative grouping is more finely tailored to the uses the holders of the identifier are expected to make of objects and permits each user to be a member of multiple overlapping groups.

Chapter 4 ▪ The System Manager

▪ Overview

This chapter describes many of the powers and responsibilities of a VAX/VMS operating system manager, from the initial bootstrapping of the system to the assignment of privileges and quotas to individuals or classes of users. The VMS operating system gives you the ability to deny or limit access, or assign priorities to realtime and interactive processes, for example. The operating system also supplies tools and defaults to help you in implementing your system design.

Topics include:

- Getting the system running
- User accounts
- Monitoring system activity
- Protection and privilege
- Error handling
- User Environment Test Package
- System management utilities

▪ Introduction

In a VAX/VMS operating system installation, the system manager controls two main areas:

- Decisions that optimize the performance and efficiency of the system
- Procedures that affect the overall management of the system

Digital supplies many tools to assist the manager in controlling these areas so that what might be complicated in some operating systems is, in the VMS operating system, straightforward and easy. In fact, a full-time system manager is not necessarily needed; system management can be shared by several persons, some of whom may serve additionally as system operators. However arranged, the management of a system has as its ultimate goal delivering efficient, economical service to all users. The VMS operating system helps by providing such features as self-installation of layered products (for example, higher-level language compilers), automatic configuration, a User Environment Test Package, and easy adjustment of parameter files.

Practically speaking, the job of the system manager is best defined in terms of the following six categories of tasks a manager typically oversees.

- Getting the system running
- Setting up users' accounts
- Managing public files and volumes
- Controlling the overall performance of the system
- Monitoring system activity
- Recognizing and dealing with errors and failures

▪ Getting the System Running

Unlike some operating systems, VMS makes it easy for the manager to get the system running. The operating system comes built. It is self-installing and automatically configuring. That means that any valid VAX hardware configuration can be supported by the VMS operating system without special configuration considerations. Many of the parameters can be adjusted to suit specific needs. For example, the system manager can increase or decrease the working set size from the default working set size with a simple instruction. In addition, tailoring of the parameter file to satisfy a specific need can go on while the system is running, so that there is no downtime or lag in productivity. The time needed for this task is, therefore, reduced, while the degree of expertise required by the manager is lessened.

Updating the system is simple: Digital supplies a command procedure to apply the update. The system manager merely runs the command procedure. The same is true for the installation of optional software, such as Digital layered products. Even the installation of customer-supplied application and system software — including user-written device drivers — is quite easy, because the VMS operating system provides a friendly environment.

User Environment Test Package (UETP)

When a VAX/VMS system is first installed and bootstrapped, an installation verification package can be used to supplement the Digital Field Service diagnostics. This package, the User Environment Test Package, is part of the VMS operating system. When run, it adapts to any VAX configuration and assures the manager that hardware and the operating system are working properly together. Errors are reported to the console terminal from which UETP was run and stored in a log file. In addition, the UETP serves as a quick check to help determine the cause when programs stop working or when any condition arises that gives the manager reason to doubt the functional integrity of the system.

The UETP performs three functions. It:

- Exercises major peripherals
- Validates VMS operating system services
- Tests major VMS software components (for example, VAX Record Management Services (RMS) or the VAX SORT/MERGE utility)

The UETP is used to exercise devices and functions that are common to all VAX/VMS operating systems; with the exception of optional features such as high-level language compilers. The system components tested include:

- Most standard peripheral devices
- The system's multiuser capability
- DECnet-VAX
- Clusterwide file access and locks

The UETP is fully automatic and requires no user interaction once started. Errors indicated during one test will not affect another test, although the same problem might occur in different parts of the UETP.

While the UETP is thorough, it is not exhaustive and should not be construed as fully diagnostic or as replacing a diagnostic test. It does not, for example, test layered products such as optional language compilers. Such products may have their own installation verification test packages in their distribution kits. The UETP tests system function, and the system manager can employ it to get a quick check of the system's condition.

Setting Up and Using a System of Accounts

Some of the main reasons for setting up a meaningful system of users' accounts are:

- To identify the users of the system.
- To define important relationships among the users of the system. For example, groups of users may share data and other files. These relationships are the basis of a system of file protection, interprocess communication, and system accounting.
- To grant to some users the privileges necessary to perform sensitive system functions, and thus to restrict other users from performing those functions.
- To set limits on the use of system resources.
- To give users priorities in using the system.

Many of the account parameters can be assigned by default, as can a large number of other values in a VAX/VMS operating system; or the manager can want to assign particular values to particular users. In either case, a record within the User Authorization File is set up for each user and contains critical accounting information.

The User Authorization File (UAF)

The User Authorization File (UAF) is one of the most important data structures with which the system manager must be concerned. The UAF contains one record for each user of the system; in effect, it defines the user to the system.

Besides the users' records, the UAF also contains a default value record and a system manager's record. In most cases, the manager will simply allow the default values for various parameters to stand. Thus, the manager may elect to choose characteristics only when warranted by a special case. The passwords on the default and system accounts, however, should be changed. These passwords are provided with all VMS systems, and intruders would try these passwords first.

Why is the UAF so important in controlling the performance of the VMS operating system? Simply stated, each process in a VMS operating system is associated with a user. Each user is allotted system resources and is given a priority and privileges, and all such attributes are specified in the user's record in the UAF. When a user logs onto the system, a process is created on behalf of that user. The process acquires the characteristics of the user. These are the same characteristics as the system manager put into or defaulted into the user's record in the UAF.

Each user's record in the UAF contains the following types of information:

- User's identification
  - User name
  - Password
  - User identification code (UIC)
  - Account name

  User's default directory name and default device name
- User's default command interpreter name
- User allotment of system resources
- User's privileges
- User's base priority

Through the User Authorization Program (Authorize), the system manager can add, delete, modify, or display records in the UAF.

The User Authorization Program

The User Authorization Program (Authorize) is a system utility required to maintain the User Authorization File (UAF). The AUTHORIZE program lets the manager:

- Create the UAF if one does not exist. A newly created UAF contains only the default value record and the system management account record; no users are yet known to the system.
- Define a new user to the system by creating a record for that user in the UAF and thus granting privileges and specifying limits and priority.
- Take away a user's right to the system by deleting that user's record from the UAF.
- Change the default record of the UAF.
- Change a user's privileges, limits, or priority by modifying that user's record in the UAF.
- Display all information about a user's account, with the exception of the user's password.
- Make a listing of all records in the UAF.

User Groups

A group is a collection of users whose processes normally have access to each others' files, file-structured volumes, mailboxes, shared pages of memory, common event flags, and the group logical name table. In addition, such processes may have special privileges to exercise control over each other. Therefore, the establishment of groups principally concerns interprocess communication and control.

In setting up a group, the system manager aims towards achieving two goals:

- To facilitate sharing of data and cooperation between users and their processes.
- To protect users from unauthorized access to their processes and data.

The importance of properly setting up groups should not be underestimated. As the system is increasingly used and as more and more files and protected data structures are created, relationships among group members, processes, devices, and data structures grow inevitably more complex. In time, it becomes harder to redefine the basic relationships among the users.

A user's membership in a particular group is defined by the User Identification Code (UIC). The UIC consists of two numbers, each ranging from 0 to 377. The first is a group number; the second is a member number.

The UIC is the basis of the VMS data protection scheme, and it is one of the factors (along with privilege) that govern the ways in which processes can interact with one another. The system manager's assignment of UICs, therefore, should involve two important considerations:

- Which users should be allowed to share data and file access, and which should not?
- Which processes should be allowed to cooperate, and which should not?

User Data Protection

For purposes of data protection, four different categories of users are defined. They are:

- Owner — users whose UICs are identical with the UIC of the owner of the data structure or device. The owner of a file is usually the creator of that file.
- Group — users of the system whose group numbers are the same.
- System — users of the system with group numbers of octal 10 or less. Certain privileges appertain to system users.
- World — all users.

All users potentially enjoy four types of access to protected data structures and devices: read (R), write (W), execute (E), and delete (D). Generally speaking, any category of user can be permitted or denied any type of access to data structures and devices. There are, however, exceptions, because not all types of access apply to all protected items. For example, execute access applies only to files that contain executable program images.

The scheme for protecting file-structured volumes is similar to that for protecting files, except that execute (E) access to a volume gives the user the right to create files on that volume.

For more information on defining user groups and assigning limits, priorities, and privileges to the user, see Chapter 3, System Security.

Limits, Priority, and Privilege

The attributes that the system manager can assign or can merely default to when creating the user's account record are:

- Limits on the use of reusable system resources
- The base priority used in scheduling the processes that the system creates for that user
- Privileges of using restricted and sensitive system functions

LIMITS - Limits are set on system resources that can be reused. An example is the amount of memory that a process can have in use for queued I/O requests. Most limit restrictions actually are placed on the use of system dynamic memory.

Usually the system manager simply assigns the default values of limits. However, the defaults can easily be overridden.

PRIORITY - A user's priority is the base priority that is used in scheduling any process the system creates for that user. There are 32 levels of software priority in the VMS operating system. For normal processes, the priority range is 0 to 15; for realtime processes, it is 16 through 31.

Processes with realtime priority are scheduled strictly according to base priority. But processes with normal priority are scheduled according to a slightly different principle, one that promotes overlapping of computation and I/O activities. This scheduling is all done transparently to the programmer and manager.

PRIVILEGES - Many system services are protected by privileges that restrict their availability to certain users. These restrictions are intended to protect the integrity of performance of the operating system, and thus the integrity of service provided to all users. The manager grants privileges to each user depending upon two factors: whether the user has the skill and experience to use the system service without disrupting the whole system, and whether the user has a legitimate need for the privilege.

Accounting For the Use of System Resources

For accounting purposes, the VMS operating system itself creates and maintains records of the use of system resources. These records are kept in an accounting log file.

Using the detailed accounting log records provided by the system, the system manager or a system programmer can establish programs for reporting on the use of system resources and for billing.

Because the users of system resources are identified in two ways, reports on the use of system resources and bills for the use of system resources can easily be generated in either of two ways: by user name or by account name.

VMS includes the accounting utility, Accounting, which uses the data in one or more previously recorded copies of the system's accounting log file to produce new files or reports. These accounting reports can be used as system management tools to learn more about the ways the system is used, how it performs, and in some cases, how particular users use the system. The reports can also be used to bill users for system resources.

The Accounting Utility gives the system manager information on operating system usage. It processes data by selection and/or sorting to produce four forms of optional output:

- A brief listing of selected records
- A full listing of selected records
- A binary copy of selected and/or rejected records
- A summary report of selected items from selected records

These forms of output can be directed to a terminal for display or to a disk or tape file.

▪ Managing Public Files and Volumes

Typically, overall planning and management of a system of public files and volumes are among the most important responsibilities of the system manager. The aspects of public files and volumes management that the system manager is most concerned with are:

- Initializing and mounting public volumes
- Regularly backing-up public files and volumes
- Installing frequently used or privileged executable images as known images or images that may be shared at runtime
- Installing frequently used sharable images as permanent global sections or images that can be shared at runtime
- Establishing systemwide logical names needed for running the executable images provided by Digital and for running other images available to all users at an installation
- Establishing disk quotas

Initializing and Mounting Public Volumes

Public volumes contain public files, which normally must be available to most users of a system. Public volumes can also contain files that users create for their own private use or for general use.

Public volumes contain the following kinds of public files supplied by Digital.

- The operating system itself in executable form and files related to the operating system.
- Utility programs in executable form. Utilities available from Digital are self-installing.
- Diagnostic and test programs in executable form and files related to these programs. Such packages as the User Environment Test Package are bundled in the system and installed along with it.
- Various system libraries: macro libraries, object module libraries, and shared runtime libraries.
- Text files; for example, the system error message file and help files, installed with the system.
- Optional software in executable form, plus related libraries and other files. Some, such as language processors, are self-installing

In addition, the system manager can include on public volumes files that are unique to an installation. Typically, these are files that must be accessible to many, if not all, users of the installation. The system manager can also permit any user to create, catalog, and store files on a public volume.

Mounting a disk volume establishes a relationship among the volume, the device on which it is physically mounted, and one or more processes that may gain access to it.

- BACKING UP PUBLIC FILES AND VOLUMES

To prevent the inadvertent loss or destruction of valuable information stored on disk file volumes, the system manager usually establishes a policy and a schedule for regularly backing up files on public volumes.

The Backup utility allows users to create back-up copies of files and directories and to restore them. It can back up entire volume sets in one operation or perform selective back-ups by file or date. Wildcarding is available, as well as several file selection qualifiers.

The Backup utility is intended primarily for use by system managers and operators; however it can be used by individual users to make personal back-up copies and to transport files.

There are two kinds of back-ups of public disk files and volumes: 1, selective, or partial, back-ups, and 2, system, or all-inclusive, back-ups. Either type of back-up can be done either to disk or magnetic tape.

▪ Installing Known Images and Creating Permanent Global Sections

The system manager can improve system performance significantly by installing certain executable and sharable images as known images and by creating permanent global sections.

There are two reasons for installing known images: to permit systemwide sharing of images that are frequently used by more than one user at a time, and to make image files more quickly accessible.

Typically, the kinds of executable images that are installed as known images are:

- Images that need more privileges than are commonly granted to users who need to execute them.
- Images that are executed frequently
- Images that are executed by more than one user at a time

A number of images supplied by Digital are ordinarily installed as known images in a site-independent startup procedure.

Sharable image sections produced by the linker are almost identical with executable image sections, except that they cannot be executed by use of the Digital Command Language command RUN. They can, however, be linked with object modules to create executable images.

Sharing common procedures leads to three significant improvements in system performance:

- Reduction of disk storage requirements
- Reduction of physical memory requirements
- Reduction of the amount of paging I/O needed

Assigning System Logical Names

A logical name is a user-specified name that may be equivalent to a file specification or to some portion of a file specification, such as a device name. A systemwide logical name is simply a logical name that can be referred to by all users of the system and by all processes created for those users.

Making sure that all needed system logical names have been assigned to equivalence names is an important part of the manager's role.

Except for default logical names, system logical names that are needed by nearly all or by all VAX/VMS installations are assigned in the start-up command procedure file, STARTUP.COM. Digital provides this as part of all software release distribution kits.

Usually the system manager is responsible for establishing the system logical names that are unique to an installation. As a rule, these names are assigned using Assign commands in the site-specific start-up command procedure.

▪ Overall Control of the System

Two important ways in which the manager exerts control over the behavior of a VMS operating system are: by maintaining command procedures of initialization commands that are essential to the proper operation of the system, and establishing output spooling and setting up and controlling batch queues, print queues, and terminal queues.

▪ STARTUP.COM AND SYSTARTUP.COM PROCEDURES

The command procedure STARTUP.COM is a start-up file that executes automatically immediately after the VMS operating system has been booted. This start-up file is supplied by Digital and contains commands for performing site-independent operations that must occur if the system is to run properly. It is not recommended that this file be edited. The operations include assigning system logical names, installing images as known images, building the I/O database, and loading I/O drivers.

The command procedure SYSTARTUP.COM is a command file that the manager can tailor to the needs of a specific installation. Typically, this file contains commands for performing such operations as setting the characteristics of terminals and other devices, purging the operator's log file, and announcing that the system is up and running.

Spooling and Batch, and Print, Terminal Queues

Usually the manager performs the following four closely related functions, which establish spooled devices and control queues:

- Establishing input and output spooling. The VMS operating system supports input spooling of batch job files and transparent spooling of output files for line-printers and terminals. Using DIGITAL Command Language commands, the system manager can easily specify which output devices are to be spooled.
- Creating and controlling batch queues.
- Creating and controlling print queues.
- Creating and controlling terminal queues.

A system manager need not learn the inner workings of spooling and queuing, but a pragmatic knowledge of how to establish spooled devices and how to create control queues is useful for efficient management of the system.

SPOOLING - Spooling is the technique of using a high-speed storage device to buffer data passing between low-speed I/O devices and high-speed main memory. The low-speed devices, which can be either the ultimate sources or the ultimate destinations of buffered I/O data, are called spooled devices; the high-speed mass storage devices are called intermediate devices.

Typically, the system manager chooses low-speed peripheral devices to include in the system's basic complement of spooled devices. At a minimum, the system manager should see that at least one lineprinter is set spooled when the system is started up. In a system with only one lineprinter, this is the default system printer. The system manager need not set a cardreader spooled, because cardreaders are spooled by default.

BATCH QUEUES - A batch queue is an execution queue that controls the execution of batch jobs. Batch jobs can enter the VMS system and be queued for initiation in two ways:

- As command procedure disk files submitted by use of the SUBMIT command. These files are placed in a batch queue and selected for execution according to their priority. By default, the name of this batch queue is SYS$BATCH.
- As batch files submitted by use of the JOB command from a card reader. These batch job files are spooled onto disk and placed in a batch queue. Unless the JOB command specifies otherwise, the name of this batch queue is SYS$BATCH. From the batch queue, batch jobs are selected for execution.

PRINT QUEUES - A print queue controls the execution of print jobs. Print queues can be one of the following types:

- Printer queue — A queue assigned to a specific print device
- Terminal queue — A printer queue assigned to a terminal device (being used as a noninteractive printer).
- Generic queue — A queue assigned to more than one printer queue
- Logical queue — A queue that is not assigned to a print device when assigned

Print jobs are queued for processing by an output symbiont in one of two ways: without the direct intervention of a user (implicitly) or with the direct intervention of a user (explicitly).

When an implicitly spooled print file destined for a spooled printer is closed, the file is placed in a print queue. Both the spooling of the output file to an intermediate device and the subsequent queuing of a job consisting of this file occur without the direct intervention of a user.

Through the PRINT command a user can explicitly queue a disk file or several files for printing. The disk file or files specified by the PRINT command are queued as a print job; if several files make up a print job, they will be printed together.

The Monitor Utility Program

The VMS operating system collects data on how the system is being used and how it responds to users' requests. The Monitor utility program can be used to examine the collected data at a specified interval and produce three forms of output:

- Statistical display on any Digital terminal
- A statistical summary file in ASCII format
- A binary recording file

Displays take the form of bar graphs and tables. Monitor can be used to observe the statistics of a running system or to read collected data and play it back.

These statistics are useful for two purposes: to aid system developers in understanding how the system operates, and to aid system managers in improving system performance.

The types of information collected and displayed are:

- Network activity
- File system statistics
- I/O system activity
- Use of the distributed lock management services
- Use of processor modes
- Page management statistics
- Nonpaged pool statistics
- Activity in the scheduler state queues
- Principal users of CPU paging and I/O resources
- System process activity

Each time Monitor is run, it starts accumulating a new set of performance measurement statistics.

The processor modes monitored are:

- Interrupt stack — percent of time processor was executing on interrupt stack
- Kernel mode — percent of time processor was executing in kernel mode (does not include time on interrupt stack)
- Executive mode — percent of time processor was executing in executive mode
- Supervisor mode — percent of time processor was executing in supervisor mode
- User mode — percent of time processor was executing in user mode (does not include time in compatibility mode)
- Compatibility mode — percent of time processor was executing compatibility mode user images
- Idle time — percent of time processor was executing the null process

The Operator's Log File

The operator's log file is a system management tool that is useful in anticipating and preventing failures of both the hardware and the software. By regularly examining it, the manager can often detect tendencies or trends toward failures and can take corrective action before such failures occur.

The system operator should, therefore, print out copies of the operator's log file regularly, and the system manager should retain copies for reference.

- RECOGNIZING AND DEALING WITH ERRORS

The error logging facility gathers and maintains information on system errors and events as they occur; this information provides a detailed record of system errors. By running the report generator program SYE, the manager or a Digital Field Service Representative can get a report of the errors and events that have occurred within a specified period of time.

USING ERROR REPORTS - The error reports generated by SYE are useful tools in two basic ways: reports aid preventive maintenance by identifying areas within the system that show potential for failure, and reports speed the diagnosis of a failure by documenting the errors and events that led up to them.

The detailed contents of the reports are most meaningful to Digital Field Service personnel. The system manager, however, can use the reports as an important indicator of the system's reliability. For example, when a report shows that a particular device is producing a relatively high number of errors, the system manager can consult Digital Field Service. By running a diagnostic program to investigate the device, field service can attempt to isolate the source of the errors. Once identified, the source of the errors can possibly be eliminated and a failure averted.

▪ System Management Utilities

At the time a VMS operating system is installed, several utilities are provided to tailor the system for a particular application environment. In addition, once the system is operational, facilities are provided to modify the environment and to upgrade/update the system with new software versions or optional software products.

VAX systems can be fine tuned, optimized, and maintained with the following utility programs

- BACKUP
- SYSGEN
- ACCOUNTING
- AUTHORIZE
- EDIT/ACL
- MOUNT
- SYSTEM DUMP ANALYZER (SDA)
- VERIFY
- INSTALL
- SHOW CLUSTER
- Other VMS Utilities

THE BACKUP UTILITY (BACKUP) - The VMS Backup Utility allows you create backup copies of files and directories from Files-11 structure Level 1 and 2 volumes and to restore them. BACKUP can be used to back up entire volume sets in one operation or to perform selective backups by file or date.

The BACKUP Utility can

- Copy disk files.
- Save disk files to a backup save set of files.
- Restore files to disk from a backup save set of files.
- Compare disk files or files in a backup save set of files with other disk files.
- List information about files in a backup save set to an output device of file.

SYSTEM GENERATION UTILITY (SYSGEN) - When VMS is installed for the first time or is upgraded from a previous version, system parameters must be adjusted. This is done with the System Generation Utility (SYSGEN). With SYSGEN, you can

- Create and modify system parameters.
- Connect devices and load their drivers.
- Create additional paging and swapping files.
- Identify the current "sit_independent" startup command procedure.
- Initialize multiport memory units.

THE ACCOUNTING UTILITY (ACCOUNTING) - The Accounting Utility enables you to obtain information on operating system usage. The accounting utility uses the data in one or more previously recorded copies of the system accounting log file to produce new files or reports. You can use the accounting reports as system management tools to learn more about how the system is being used, its performance, and determine its use by individual users.

It processes data by selection and/or sorting to produce four forms of optional output:

- A brief listing of selected records.
- A full listing of selected records.
- A binary copy of selected and/or rejected records.
- A summary report of selected items from selected records.

These forms of output can be directed to a terminal for display or tape file.

THE AUTHORIZATION UTILITY (AUTHORIZE) - The VMS Authorization Utility is a system management tool used to setup user accounts and assign privileges to those accounts. Two functions of the AUTHORIZE Utility are to

- Create and modify records in the system and network user authorization files.
- Create and maintain the rights database.

THE EDIT/ACL UTILITY - The ACL editor is a screen-oriented editor used to create and maintain access control lists (ACLs). Through the use of ACLs, you can define the type of user access to a system object, such as a file, device, or directory.

THE MOUNT UTILITY - The Mount Utility (MOUNT) allows you to enable a disk or tape volume and make it available for processing. When you issue the command, MOUNT, the Mount Utility ensures that

- The device has not been allocated by another user.
- A volume is physically loaded on the device specified.

THE SYSTEM DUMP ANALYZER UTILITY (SDA) - The System Dump Analyzer Utility helps determine the cause of system failures. When an error within the system interferes with normal operations, the VMS operating system writes information concerning its status to a system dump file. The SDA reads, formats, and displays the contents of this file. You can use the SDA to display information on a video display terminal or create hard-copy listings.

THE VERIFICATION UTILITY (VERIFY). - The VMS File Structure Verification utility is used by system managers and operators to detect inconsistencies and errors in file structures. You can reclaim a significant amount of disk storage by identifying lost files and files marked for deletion.

VERIFY will operate in any of three modes:

- Error reporting with no repairs
- Error reporting with errors
- User-controlled selective repairs

THE INSTALLATION UTILITY (INSTALL) - The Installation Utility is a system management tool that makes images known to the system in order to

- Speed activation of frequently used images
- Allow users with standard privileges to run images that require additional privileges
- Support user-written system services

The utility also provides information that helps system managers perform routine maintenance operations on images that require additional privileges.

THE SHOW CLUSTER UTILITY (SHOW CLUSTER) - The VMS Show Cluster Utility monitors VAXcluster activity and performance. Information taken from the System Communications Services (SCS) database, the connection management database, and the CI port database, is output to your terminal or other specified device or file.

You can use the SHOW CLUSTER utility to produce two basic types of reports — cluster reports and local_ports reports.

Cluster reports contain information about the entire cluster. By default, the cluster report includes the names of the nodes in the cluster, the operating system and version each node is executing, and an indication of whether each node is recognized by the local node as a cluster member.

Local_ports reports contain information about the ports on the local system.

THE NETWORK CONTROL PROGRAM UTILITY (NCP) - The Network Control Program Utility is used by system managers and operators to configure and control DECnet-VAX networks. System managers can also use this utility to monitor network resources and test network components.

THE VMS FILE DEFINITION LANGUAGE (FDL) - The VMS File Definition Language (FDL) is a special-purpose language used to write specifications for data files. These specifications are written in text files called FDL files that are then used by the RMS Utilities and library routines to create the actual data files.

RMS UTILITIES - RMS utilities include

THE ANALYZE/RMS_FILE UTILITY - The Analyze/RMS_FILE Utility (ANALYZE/RMS_FILE) allows you to examine the internal structure of a VAX RMS file. ANALYZE/RMS_FILE can perform five functions. It can

- Check the structure of a file for errors.
- Generate a statistical report on the file's structure and use.
- Enter an interactive mode through which you can explore a file's structure. This analysis can determine if the file is properly designed for its application and can point out improvements to make in the file's FDL specification.
- Generate an FDL file from a data file.
- Generate a summary report on the file's structure and usage.

THE CONVERT UTILITY (CONVERT) - The Convert Utility copies records from one or more files to an output file, changing the record format and file organization to those of the output file.

You can also use CONVERT to reformat an indexed file in which you have deleted or inserted many records. The file's specification is used as both the input and output file specification.

The Convert/Reclaim Utility (CONVERT/RECLAIM). The Convert/Reclaim Utility reclaims empty buckets in Prolog 3 indexed files so that new records can be written into those buckets.

A set of library routines can be used to perform the same function of both utilities from within a program.

THE EDIT/FDL UTILITY - This utility allows programmers to interactively create or modify an FDL file. EDIT/FDL was developed especially to manipulate FDL files and has many special features that simplify the processing of creating FDL files.

THE CREATE/FDL UTILITY - This utility uses the specifications in an existing FDL file to create a new, empty data file. You can either supply CREATE/FDL with the file specification of the new data file, or CREATE/FDL can use the specification given in the FDL file itself.

THE MAIL UTILITY (MAIL) - The VMS Mail Utility allows you to send messages to other users on your system or on any other computer connected to your system. You can also read, file, forward, delete, print, and reply to messages that other users send to you.

THE PHONE UTILITY (PHONE) - The VMS Phone Utility lets two VAX users communicate with each other, interactively. PHONE is designed to simulate some of the features of real telephone conversations.

Chapter 5 • VAXcluster Software

• Introduction

VAXcluster systems are the heart of a new concept in VAX/VMS computing. Clustering of VAX computers places greater computing power, a larger computer system, and more resources and data at your fingertips. Whether you are a VAX owner looking for ways to extend your computer system or are just considering how VAX computing could help your organization, the VAXcluster approach could be the perfect answer to your computing needs. It enables you to provide computer resources and applications to your entire organization as well as to the individual user. What's more, you can deal with expansion and change in a productive and controllable manner. Without needing to acquire new skills or change your present work patterns, you can gain greater access to information stored in databases throughout your company and improve your ability to share this information over a larger network.

This chapter explains what a VAXcluster system is, how it operates, and how your applications can benefit from it. The first two sections describe a VAXcluster system and the benefits it provides. Subsequent sections discuss the major software pieces that were engineered in VMS V4.0 for VAXclusters and system management of a VAXcluster system from the system manager's viewpoint. Chapter 5 is a description of the major software pieces that were engineered in VMS V4.0 for VAXclusters.

• What is a VAXcluster?

A VAXcluster configuration is a system that combines two or more VAX processors, and mass storage servers if desired, in a loosely coupled manner. Clustering is possible with VAX-11/750, VAX-11/780, VAX-11/782, and VAX-11/785 processors. The mass storage server is a free-standing, high-speed, intelligent device designed to the specifications of the Digital Storage Architecture and known as the Hierarchical Storage Controller, or HSC50. Each VAX processor or HSC50 in a cluster is called a node. Up to 16 nodes may be connected in a VAXcluster via a high-speed bus known as the Computer Interconnect, or CI.

One way to look at multiprocessing systems is to consider a spectrum of systems with two extremes. At one end of the spectrum would be a very tightly coupled multiprocessing system. Within the VAX family, it would be the VAX-11/782 system — a tightly coupled multiprocessor characterized by one copy of the VMS operating system residing in shared memory between two processors. At the other end of the spectrum is the computer network. The network is a very loosely coupled multiprocessing system in which each processor has its own copy of the operating system. The network systems can be tied together by DECnet software using point-to-point, X.25, or Ethernet connections.

▪ Highlights of a VAXcluster

VAXcluster configurations may grow system-by-system to meet the increased demand of users. As an organization's needs grow, more computing or I/O resources can be plugged into the VAXcluster without interrupting its operation.

All VAXcluster disks, whether connected to a VAX processor directly or through an HSC50 controller, may be made available to a user logged on to any VAX node. Data on any VAXcluster disk may be shared at the volume, file, and record levels by users on either the same VAX processor or different ones.

Batch and print jobs may be load-balanced throughout the VAXcluster.

Redundancy features built into the hardware combined with failover capabilities of the software provide greater system and data availability.

These features and others are discussed in greater detail in subsequent sections.

▪ VAXcluster Benefits

VAXcluster systems offer a range of computing benefits. Some users may use all of them while others, because of the nature of their needs, will take advantage of only some. In either case, VAX clustering provides an effective answer for many of the problems facing managers of computer systems.

The following are some of the high-level benefits that a VAXcluster system can bring to users:

- Increased growth potential
- Increased data sharing
- Greater data and system availability
- Preservation of investment
- Lower costs and less expensive upgrades

Incremental Growth Capability

In the past, if a system became overloaded because a particular element could not meet the increased demand, the crucial question was whether the system could be expanded to handle the load. It could if, for example, the system lacked memory but space remained for memory expansion. However, if the CPU were overloaded, the problem was more difficult to solve.

The VAXcluster system solves the dilemma. Any type of resource — memory, CPU, mass storage controllers (HSC50s), or disks and tapes — may be added to keep pace with user demand. More processors, HSC50 controllers, and disks or tapes may be added to an operating VAXcluster without interrupting normal operations.

It is possible to connect up to 16 nodes by clustering VAX processors, alone or in combination with HSC50 devices. Moreover, each HSC50 can support up to 24 disks or tape formatters. That means you can create very large systems, either all at once or system by system.

Because the demands on most computer systems increase with time, the VAXcluster system is an elegant response to that growing demand. You can start at an early stage because as few as two connected VAX systems constitute a VAXcluster. But it ultimately can grow into a very large system containing many VAX systems and mass storage controllers.

Increased Data-Sharing

At the system manager's discretion, a VAXcluster user can have access to disks connected locally to any VAX processor or any HSC50 in the cluster. Files on those disks may be shared at the record level by all VAXcluster users, retaining full read-and-write capability.

This new capability is a major step forward in operating-system software technology and allows for increased data-sharing. Applications may be spread across multiple processors, or users of one application may operate across multiple processors. The system manager may choose to set up a homogeneous VAXcluster environment, in which users need not be concerned about which VAX processor they are using because each processor presents an equivalent environment.

Greater Data and System Availability

Data in a VAXcluster system is more accessible and more secure as a result of built-in hardware redundancy, the new features in VMS V4.0, and the presence of several VAX processors in the VAXcluster.

Hardware redundancy is inherent in the computer-interconnect (CI) data path; in the Star Coupler, which is the central connecting point of all nodes; and in the ability to connect two HSC50s to each Digital Storage Architecture (DSA) disk or tape in the cluster.

One VMS V4.0 software feature that enhances data availability is automatic failover for a dual-ported disk drive if its HSC50 fails. The failover is transparent to the VAXcluster users.

If a processor in a VAXcluster fails, VMS V4.0 releases all the locks that the failed processor has on resources so the remaining VAX nodes can continue to work.

These hardware and software features permit configuration of systems that provide greater system and data availability because processing can continue despite certain system failures.

Preserved Investment

The VAXcluster concept, as a state-of-the-art method of configuring computers, is designed to support the newest hardware and software, such as the CI bus, HSC50 mass storage controllers, and Digital Storage Architecture (DSA) disks and tapes. However, customers who have invested in UNIBUS and MASSBUS disks may desire to continue using them in a VAXcluster environment, thereby preserving their investment. They can, thanks to software in VMS V4.0.

A VAXcluster may be configured without HSC50s. Any UNIBUS or MASSBUS disk (which, by definition, is connected locally to a VAX processor) may be shared by all users in a VAXcluster. The system manager determines which local disks will be accessible to all cluster nodes. VAXcluster systems also permit full read-and-write access to dual-ported MASSBUS disks along both access paths simultaneously. Other nodes in the cluster may access a dual-ported MASSBUS disk using any available path, with fully automatic failover, should either of the directly connected processors fail.

In this way, the state-of-the-art software allows VAXcluster configurations that contain UNIBUS or MASSBUS disks, preserving and enhancing customers' investment in earlier disk technology.

Lower Expansion Costs

The VAXcluster approach can reduce the cost of expanding your system when a hardware component no longer meets user demands. In the past, if the limiting component was the VAX processor itself, for example, then another CPU had to be purchased — along with its own system disk, I/O controllers, and possibly more terminals. With the VAXcluster, the only necessary purchases are another processor, CI interface, and memory. No longer is there any need to buy a separate system disk, data disks, terminals, or printers for the new processor.

The new system may be booted from an existing disk on an HSC50 in the VAXcluster, and programs and data already existing may continue to be used in the cluster. The new processor can be accessed from existing terminals if they are connected to the VAXcluster through a terminal server. Lineprinters already in the VAXcluster also may be used. The result is lower cost because additional hardware previously required to expand a system often is not needed when a VAXcluster is enlarged.

▪ System Availability

The following diagram shows a fully configured VAXcluster. Each numeric label is described below to indicate the added availability the particular feature brings to the VAXcluster.

The Computer Interconnect (1) has built-in redundancy. It is a dual-path bus. Under normal operations both paths can be used; however, if one of the paths fails, communication can still occur over the remaining path. The VMS operating system periodically tests a failed path and puts it back online once the failure has been repaired.

The Star Coupler (2) also has built-in redundancy. A CI consists of four coaxial cables, two to transmit and two to receive. Inside the Star Coupler are two transformers for every eight nodes. One transmit cable and one receive cable are connected to the first transformer, and the remaining transmit cable and receive cable are connected to the second transformer. Every HSC50 or VAX processor node connects its CI in this way. In the very unlikely event that a transformer inside the Star Coupler fails, communication can still occur via the remaining transformer.

Digital Storage Architecture (DSA) disks and tapes may be connected between two HSC50 controllers (3). If one of the HSC50s fails, the VMS operating system will failover to the second HSC50, transparently to the user. Under normal operation, disks and tapes may be manually load-balanced between both HSC50s. If one of the HSC50s fails, the disks that were online to the failing HSC50 then become online through the other HSC50. Even though DSA devices can be physically connected to two controllers, only one of the controllers has an active path to the device for data transfer. On a failover, the active path may change to a different controller. But since a device can have only one active path at a time, dynamic dual-porting, where I/O occurs simultaneously from both controllers, is not available with DSA controllers. However, this is available with MASSBUS disks.

Dual-ported MASSBUS disks (4) are supported in VMS V4.0, with full read-and-write capability from both VAX processors simultaneously. However, the dual-ported MASSBUS disk cannot be a VMS system disk, and a CI must be present so that the distributed file system and distributed lock manager may coordinate file and record access on the dual-ported MASSBUS drive.

The terminal server software (5) allows a user to have two or more simultaneous sessions on different nodes. The automatic login failover switches a user's terminal to an alternate node in the cluster if the user's current node fails. The terminal server software also provides load balancing at login time by connecting the user to the least-loaded VAX processor in the cluster.

A DSA disk or tape may be dual-ported between two UDA50s or a UDA50 and an HSC50 (6). However, in either of the configurations, if one of the active paths fails, failover must be performed manually. The system manager must dismount the drive, press the port buttons on the drive, and remount the drive to connect it to the alternate controller.

VAX Processor Failures

In the event that a VAX processor in the cluster fails, all the resources that the failed processor had locked are released. This is a function of the distributed lock manager. When the system detects that a node in the cluster has failed, the remaining VAX nodes temporarily suspend normal operation and the distributed lock manager releases all the locks held by the failed VAX processor. Once this is complete, processing will continue normally on each remaining VAX node as long as the cluster still has enough VOTES to satisfy the QUORUM algorithm requirements.

Failure of a VAX processor can be detected in two ways. If the reason for the failure of the VAX processor is a software or hardware condition that results in the BUGCHK code being executed, or the system is shutdown by an operator, a datagram is sent out telling the remaining nodes of the failure. If the CPU is halted or, for some reason, the datagram is unsuccessfully transmitted (the CI port has ceased to function), the failed VAX node is detected by software polling, which occurs at an interval selected by the system manager. There are also parameters that control the number of nodes polled and, on detecting a failure, determine how long to wait in case it is merely a transient failure.

▪ VMS V4.0 And The VAXcluster

A range of new VMS components, from class and port drivers to the cluster server process, were engineered for VMS Version 4.0 in order to support the VAXcluster. This section describes new components along with some of the lower-level components that have been available since VMS V3.3. It also includes discussions of DECnet in a VAXcluster, the QUORUM algorithm and its impact on system configurations, device naming, and VMS features that are not supported across the VAXcluster.

The following VMS V4.0 components will be highlighted:

- Digital Storage Architecture (DSA) Class and Port Drivers
- System Communication Services
- MSCP (Mass Storage Control Protocol) Server
- Connection Manager
- Distributed Lock Manager
- Distributed File System and RMS
- Distributed Job Controller
- Cluster Server Process

DSA Class and Port Drivers

The Digital Storage Architecture (DSA), which is Digital's latest disk-and-tape technology, is called an architecture because the DSA disk/tape controller has clearly defined functions. A protocol known as the Mass Storage Control Protocol, or MSCP, has been defined between the host processor and the controller. Because of its architectural framework, it is very easy to add new devices and controllers to a VAXcluster. Two of Digital's controllers that adhere to the architecture are the UDA50 and the HSC50, both of which understand the MSCP protocol.

To support the new architecture, VMS includes device drivers for the DSA devices. Unlike a traditional device driver, the driver for the DSA device has two parts, the class and port drivers.

The class driver can be considered to be the generic driver handling all QIO (queue input/output) requests. The disk class driver accepts all disk QIOs for DSA disks; however, it doesn't "know" the physical port through which the I/O will ultimately pass. This is the job of the port driver.

With VMS V4.0, there are three class drivers: the disk class driver (already mentioned in connection with DSA disks), a tape class driver for DSA tapes, and a DECnet class driver for DECnet I/O over the CI. In the case of the tape and disk class drivers, the QIO received by the class driver is converted into an MSCP message. The DECnet class driver converts its QIOs into messages that adhere to the DECnet protocol.

The port driver understands the physical port and nothing else. There is no direct interface to the physical port other than through a class driver. It ensures that data passes through the port successfully or returns an error status. With VMS V4.0, there are two port drivers, one for the UDA50 and one for the CI780 or CI750, the CI controllers.

An I/O request from a user process to a disk connected to the UDA50 passes through the disk class driver and the UDA50 port driver. An I/O request to a disk connected to an HSC50 passes through the disk class driver and then the CI780/CI750 port driver. Therefore, any future hardware controllers that Digital engineers for DSA devices will require, at most, only a new port driver, not a new class driver. In addition, the same port driver, the CI port driver, may be used by a number of class drivers to handle different classes of I/O requests. For example, the CI port driver would be used by the disk, tape, and DECnet class drivers.

System Communication Services

System Communication Services, or SCS, is a layer of software between the class driver and port driver. The CI supports three types of data transfers: datagram, sequenced message, and block data transfer. The datagram is used by DECnet, the sequenced message is used by the VMS system for short messages between nodes — for example, distributed-lock-manager requests — and the block data transfer is used for data movement between the cluster nodes, such as input and output by disks and tapes. Consequently, it can be seen that the CI hardware was designed with consideration of the kind of data transfers required by VAXclusters. Therefore, the different modes of data transfer available to the VMS operating system provide efficient data transfers over the CI.

SCS provides a software interface to the three different transfer modes. An example is a disk I/O. An I/O request is made from a user's program. This is interpreted by the disk class driver, which produces the correct MSCP message. The MSCP message is then passed to the SCS level, which packages it as a sequenced message. The sequenced message is then transferred by the port driver through the CI port. As long as the disk to which the I/O was requested is on an HSC50, the HSC50 receives the MSCP message. After it has been decoded by the HSC50, the HSC50 then satisfies the I/O by transferring the data in a series of block data transfers, giving each one to the SCS level for correct packaging and transfer over the CI. When the I/O is completed, an MSCP message is sent via SCS from the HSC50 back over the CI to the disk class driver, thereby completing the I/O.

MSCP Server

Software called the MSCP server is responsible for one of the attractive features of the VAXcluster concept: preserving customers' investment in MASSBUS and UNIBUS disks. It does this by allowing MASSBUS and UNIBUS disks to be shared throughout the cluster.

MSCP is the protocol used to communicate between a VAX host and a DSA controller. If a user process running on VAX A in a VAXcluster has an I/O operation to perform to a MASSBUS or UNIBUS disk connected to VAX B in the same VAXcluster, the process queues its I/O to the disk class driver, as described above. The processing on the VAX node making the I/O request is the same, whether the I/O request is made to a MASSBUS or UNIBUS disk connected to another VAX processor or to a DSA device connected to an HSC50. The I/O request is transmitted in the form of an MSCP message communicated via SCS over the CI. When it arrives at the VAX processor (VAX B) that is connected to the MASSBUS or UNIBUS disk, the MSCP message is interpreted and translated into a MASSBUS or UNIBUS disk I/O request. This is the job of the MSCP server, which is also responsible for transferring the data over the CI.

The MSCP server enables a VAX processor to make locally connected MASSBUS and UNIBUS disks available to all other users of the VAXcluster. The system manager, however, decides which disks should be that widely available.

Connection Manager

The chief function of the connection manager is to determine VAXcluster membership and to handle VAXcluster state changes. As new nodes are connected to the VAXcluster, or are removed or fail, the connection manager tracks the state changes and, in conjunction with connection managers running on each of the VAX processor nodes, ensures a consistent view of the cluster. The connection manager also prevents cluster partitioning via the QUORUM algorithm, which is described later in this section.

Another function of the connection manager is to provide an acknowledged-message delivery service. This means the connection manager is responsible for sending messages on the behalf of upper software levels via SCS and the CI to other VAX processors in the VAXcluster. The connection manager either successfully delivers the message or returns an error indicating that the target node has left or is about to leave the cluster. Transient failures are invisible to the upper layers of software. By adopting this approach, the higher-level system layers don't have to deal directly with SCS or transient failures.

Distributed Lock Manager

The distributed lock manager is a natural extension of the lock-management capabilities that were introduced for a single node in VMS Version 3. The distributed lock manager provides a name space for resource names; many types of locks may be taken out against the resources. Processes requiring access to a locked resource are queued. A resource could be a device, file, record in a file, or a fictitious resource used as a signaling mechanism by processes on different VAX nodes. The distributed lock manager allows synchronized use of resources throughout a VAXcluster. Automatic deadlock detection also has been expanded to work in a VAXcluster system.

The distributed lock manager is used by the file system, RMS software, job controller, device allocation, and other utilities for cluster synchronization and is available to users to develop cluster applications.

If a VAX node fails, all locks held by the failing node are released, thereby allowing the remaining VAX nodes to continue processing instead of stalling while waiting for a lock that may never be released.

The lock manager has been designed so that the cost of lock operations is fixed, regardless of the number of VAX processors in the cluster. As a result, the cost of synchronization is not affected by the size of the cluster. In other words, adding additional VAX processors does not in itself increase the synchronization cost.

Distributed File System and RMS

As with the distributed lock manager, the distributed file system is an extension of the file system that existed on VMS V3. The file system processes virtual I/O functions for mounted volumes. For disk volumes, this involves functions such as creating, deleting, and extending and truncating files, as well as maintaining file directories, mapping virtual to logical I/O, arbitrating runtime access to files, enforcing file protection, and enforcing and maintaining disk usage quotas. Most of these activities can interact with each other and therefore must be coordinated. Put another way, processing of various functions from different processes may have to be serialized.

In VMS V3, serialization was achieved by performing all of the above functions in the context of a detached process known as an ancillary control process (ACP). The ACP processes one request from start to finish before starting another, thus eliminating all possible interactions between different functions. Attempting to extend this design to a cluster would create a potentially serious bottleneck for virtual I/O activity. Instead, the VMS V4.0 distributed file system handles virtual I/O requests as an extended QIO procedure, or XQP.

XQP is a new technique developed to allow vastly greater concurrence for file system activity in the cluster. In addition, it benefits all VMS systems. The XQP uses the distributed lock manager to serialize virtual I/O functions for the specific file or directory in use. The code is mapped into the P1 area of each process. If the function modifies disk storage (for example, the creation, deletion, extension, or truncation of a file), the function is serialized with respect to other concurrent requests that modify disk storage on the same volume.

In practice, the XQP technique means that most file system functions may proceed in parallel with file system requests issued by other processes, whether the processes are executing on the same node, or on another node in the cluster. In addition, the various caches used in the VMS V3 ACP to reduce disk I/O have been reimplemented to perform similarly in the XQP environment. Various lock manager features are used to validate specific elements in the processor-specific caches within the cluster.

The VMS Record Management Services (RMS) have been extended on VMS V4.0 to take advantage of the VAXcluster environment. The RMS utility also has been interfaced to the distributed lock manager, allowing RMS files to be used on any disk in the cluster by any user on any VAX processor node. RMS files may be shared clusterwide at the record level, using standard RMS file-sharing and record-locking features.

Distributed Job Controller

The distributed job controller allows the batch-and-print-queues database to be shared clusterwide. The distributed lock manager is used as a signaling mechanism to cause other VAX processor nodes in the cluster to examine the batch and print queues for jobs to be processed. Consequently, it is possible to create generic clusterwide batch and print queues, as follows:

- CLUSTERWIDE BATCH QUEUES.
  In the case of a clusterwide batch queue, all batch jobs submitted to this generic queue are spread across the cluster. The algorithm attempts to keep each VAX processor equally loaded.

  A system manager can control how batch jobs are distributed as described above, where each node is a peer and all batch jobs are evenly distributed in number across the cluster. For example, if a VAXcluster consists of five VAX nodes and a user on the first VAX processor submits five batch jobs, one of the batch jobs may be run on each of the VAX processor nodes in parallel. At the other extreme, just one of the VAX processor nodes in the VAXcluster is the "batch" machine; in this case all batch jobs will run on the "batch" machine, regardless of the VAX node on which they were submitted. Those are the two extremes available to a system manager. But an intermediate situation is also possible. The load-balancing algorithm looks at the number of available batch slots and running jobs on each machine and attempts to keep their ratio even on each VAX node. Therefore, certain machines can be set to accept more batch work than others by setting up the batch queue parameters accordingly.

- CLUSTERWIDE PRINT QUEUES.
  It is also possible to have a generic clusterwide print queue. Assuming again that you have a cluster of five VAX nodes, each with a printer, if print jobs are submitted to the clusterwide generic print queue, the job will be directed to the printer device with the shortest queue. This allows print jobs to be spread across the VAXcluster. However, users may also submit their print jobs to any printer connected in the cluster. This is particularly useful when a cluster includes equipment that is not duplicated on every node, such as a letter-quality printer or laser printer.

  Print jobs may be submitted to this printer from any VAXcluster processor node. Or a cluster may have only one printer connected to one of the VAX processors; this printer, too, may be used from any of the VAX processors in the cluster.

Cluster Server Process

The cluster server process exists on each VAX processor node in the cluster. It performs global cluster functions. For example, it is used by the $Brdcst system service to broadcast messages to any or all terminals in the cluster. It is also used by the $Mount command to mount a disk globally on each of the VAX nodes. (The mount command need be issued only once from one of the VAX processors.)

DECnet in the VAXcluster

DECnet software is required in a VAXcluster for two reasons. First, a system manager will want to be able to control the cluster from any terminal and therefore may require the virtual-terminal capability that DECnet provides. Secondly, if communication is desired between processes on different processor nodes in the cluster, it is achieved with DECnet. Because the DECnet package has its own class driver, DECnet can utilize the CI directly, using the datagram service provided by the CI hardware. Each VAX processor in a cluster is seen as an individual DECnet node with a unique node name and node number. Digital requires that the DECnet node name and number be the same as the cluster node name and cluster identification number on each VAX processor.

Digital also recommends that every DECnet node in a VAXcluster be a full-routing node when the CI is being used. DECnet connections utilizing the CI are point-to-point; therefore, to prevent one node from routing all DECnet traffic, each node should be a full-routing node. VAX nodes could also be connected to Ethernet, which does not require the nodes to be full-routing nodes but allows them to be end nodes only. This permits the option of DECnet using the Ethernet connection to connect to other VAXclusters or DECnet nodes in a local area network (LAN). The Terminal Servers described in the hardware section can also be placed on the LAN. Of course, each VAX processor node in the VAXcluster can have traditional point-to-point connections for long-distance wide-area connections or can use X.25 connections.

▪ VMS V4.0 QUORUM Algorithm

A VAXcluster can share disks and their data. Consequently, the coordination of access to the data resources and the result it has on data integrity is very important. This is achieved in the cluster by each VAX processor node having a strong sense of cluster membership, which prevents two clusters from sharing the same data resource. A VAX processor node may only participate in one cluster. If sharing were allowed, between two or more clusters, the data resource could be written to and read in an uncoordinated manner, and data integrity could be lost.

The situation where two clusters share the same data resource in an uncoordinated manner is known as a partitioned cluster. Partitioning occurs when two or more nodes operate as if they belong to separate clusters although they are intended to be part of the same cluster.

With VMS V4.0, partitioning is prevented by a scheme known as the QUORUM algorithm. Each VAX node in the cluster has a SYSGEN parameter known as VOTES and a parameter known as QUORUM. Typically, each VAX node sets its VOTES parameter to 1 and the QUORUM parameter to the sum of VAX nodes divided by 2, plus 1, rounded down to the next integer value. (In this calculation, HSC50 devices are not counted as nodes.) For example, a cluster consisting of five VAX nodes and two HSC50 nodes would have a QUORUM of 3.

Processing may proceed only if the sum of the VOTES of the processors is at least equal to the QUORUM. In this example, if the cluster were to be partitioned into two clusters of three VAX nodes and two VAX processor nodes, then the three-node cluster would continue to function because it would still have 3 VOTES, enough to meet QUORUM. However, the second cluster would cease to function, since it has only 2 VOTES, or below QUORUM. Using this scheme, it is impossible for a properly configured cluster to become partitioned.

The QUORUM algorithm also affects the number of VAX processor nodes that may fail or leave the cluster before the remaining VAX processor nodes are affected. In the case of five VAX nodes with a QUORUM of 3, two VAX nodes may fail without affecting the remaining nodes. However, on the failure of a third node, the total number of votes falls below the QUORUM parameter and all processing in the cluster will stop until QUORUM is regained. If, however, after the first two nodes fail, it is known that the VAX processors are going to be down for some time, then the QUORUM value can be adjusted on each of the remaining VAX nodes by an operator or system manager. This means, for the example, that after the two nodes fail, there would be three remaining nodes with a QUORUM of 3. Consequently, the cluster could not tolerate a further failure because it would reduce the total number of VOTES below the QUORUM value. However, if the QUORUM value were adjusted to 2, then another VAX processor failure could be tolerated and the remaining nodes could continue running.

Therefore, it is apparent that the QUORUM algorithm has an effect on the configuration of VAXclusters. The bigger the cluster, the greater the number of VAX processor nodes that can fail before the number of VOTES drops below QUORUM and all processing is suspended in the cluster. The QUORUM algorithm works well for three or more VAX nodes but causes a problem in the case of two VAX nodes. With a cluster of only two VAX nodes, the QUORUM for the cluster is 2. Then, if either processor fails, the remaining processor would wait for QUORUM to be restored. This may be satisfactory for some VAXcluster users.

Another possibility is that a master-slave relationship is established. This can be achieved by giving the master processor 1 VOTE and the slave processor 0 VOTES, assuming a QUORUM of 1. In this case, if the slave VAX processor node were to fail, the master VAX processor node would continue to operate normally. However, if the master processor node were to fail, then both nodes would be unavailable until QUORUM were regained.

The third solution introduces one more concept to the QUORUM algorithm and is known as a quorum disk. A quorum disk is a disk that can be written to and read by all VAX nodes in a cluster. A quorum disk may be a disk on an HSC50 or for a two node cluster a dual-ported MASSBUS disk. The quorum disk is assigned a number of votes. If a cluster contains a quorum disk and it meets the criteria of being able to be written to and read by all VAX nodes, then the votes assigned to the quorum disk are counted towards the cluster QUORUM.

Therefore, in the case of a two-VAX-node cluster with a quorum of 2, if a quorum disk assigns one vote, then the failure of one of the processor nodes can be tolerated because there is still a quorum of 2 VOTES. In a normal running situation — both VAX nodes operating with a quorum disk — 3 VOTES would be counted. A quorum disk may be used by a cluster of any size to introduce extra VOTES, thereby tolerating the failure of extra VAX nodes.

Device-naming in a VAXcluster

Each node in a VAXcluster must have a unique name and SCS node number. To avoid confusion, Digital requires that the name and number be the same as the name and address of the DECnet node. The names consist of six characters and are assigned both to VAX processor nodes and to HSC50 nodes.

Each device in a cluster must also be identified by a unique name. A device connected to a particular node therefore would adopt the node name as a prefix to its own name. For example, a terminal line on a VAX processor named STAR might be STAR$TTA5: or an RA81 disk on a HSC50 with a name of PRIDE might be PRIDE$DUA0:. (RA80s and RA81s have the designation DU, and RA60s have the designation DJ.) Another example would be a MASSBUS disk connected to a VAX processor that is made available to the rest of the cluster via the MSCP server. An RP06 device connected to the VAX processor STAR could have the device name STAR$DBA0:. This approach varies only in the case of dual-ported disks. If disks are dual-ported between two HSC50s, or a MASSBUS disk is dual-ported between two VAX processors, the VMS operating system needs a universal device name that is known and doesn't change, regardless of what paths are available.

The answer is a concept called an allocation class. Each pair of VAX processors or HSC50s attached to a dual-ported disk are assigned a unique allocation-class number ranging from 1 to 255. For example, if a disk is dual-ported between the HSC50s named PRIDE and GREED, a unique allocation-class number, perhaps 255, must be assigned to both HSC50s. An RA81 disk dual-ported between the two HSC50s therefore might have a device designation 255DUA0:. The name is valid as long as either path is known.

Each disk connected to both VAX processors within the same allocation class must have a unique device name. For example, take the case of two VAX processors each having an RP06 device designated DBA0: and also sharing a dual-ported MASSBUS disk, thereby being assigned the same allocation class. In that case, one of the disks named DBA0: would be required to change its unit number to make it unique. The dual-ported MASSBUS disk must have the same name on both systems; that is, the same controller letter and the same unit number.

VMS Features Not Supported Across the VAXcluster.

Not all features available on a single VAX processor node are available across the cluster. Some features do not lend themselves to VAXcluster use. Other features not currently available may well be implemented in future stages of the VAXcluster program.

Features that are not implemented include

- Process control system services ($CREPRC, $SUSPND, $SETPRI)
- Accounting data
- Hardware error log
- Event flags
- Logical names
- Mailboxes
- Writable global sections

Time is maintained independently by each node. However, when a new cluster node boots, an attempt is made to set time from another node that is already in the cluster.

All of the above features continue to work on each of the cluster nodes. But, for example, an event flag set on VAX processor A in a cluster would not be set on VAX processor B of the same cluster.

It also should be noted that not every application is suited to VAXclusters. Care must be taken when migrating time-dependent applications to VAXclusters that require realtime, time-critical responses. The coordination and communication activity among VAXcluster members may require the suspension of processing activity for periods of time which, while relatively short, would not meet critical response times demanded by some applications.

▪ Managing A VAXcluster

A system manager has a number of choices when setting up a VAXcluster system. Each processor may have a separate User Authorization File (UAF) or there may be one UAF for the entire cluster. The UAF determines which users may log onto the system and, when a user has logged on, defines the user's default disk, directory, user identification code and privileges, and other entities. If each system has its own UAF, then different users may be restricted to certain VAX processors. However, the system manager must coordinate the different UAFs on the different VAX processors. For example, the system manager must ensure that a user has the same UIC on every VAX processor that is available to the user and that no two users have the same UIC.

The alternative is to have a single UAF placed on a shared disk for the entire cluster. In that case, the same UAF will be accessed whenever a user logs onto any VAX processor in the cluster. With this method, the user's process always will be defined the same way, making it irrelevant to a user which VAX processor he actually is using. If the cluster is set up this way and is connected to Ethernet, then the load-balancing feature of the Terminal Server may be used, as described in Chapter 3.

Single or Multiple System Disks

VMS V4.0 supports two or more VAX processors booted from the same physical disk drive. VMS V3 introduced the concept of rooted directories, which allow two or more VMS systems to reside on one disk, each under its own root named SYS0 through SYSF. VMS V4.0 uses this feature to allow multiple VAX processors to boot from the same disk drive. VMS V4.0 allows the VAX nodes of a VAXcluster to boot from any combination of the following:

- One VAX processor per system disk
- Multiple VAX processors per system disk sharing no system files
- Multiple VAX processors per system disk sharing virtually all system files

VMS V4.0 also allows the sharing of VMS and most layered products. Portions of those products that must be separate for use by each VAX processor are placed in a separate rooted directory. Therefore, the majority of VMS and layered products are placed in a common directory to be shared by all nodes booting from the disk. This greatly decreases the complexity of managing VMS and layered products in a cluster, since any ongoing software updates have to be applied only once.

Plans to use shared system disks prompts the question of how many VAX processors should be booted from the same disk. The answer almost certainly will be determined by performance. If the shared VMS and layered-product option described above is implemented, then all of a system's software and layered products can be considered in three parts. They are the portions of the system that must reside in a unique rooted directory for each VAX processor, the common directory containing common VMS and layered-product components, and each VAX processor's page- and swapfiles.

How these sections of the system are spread over cluster disks will determine the system's ultimate performance. The decision should be based on performance rather than on space requirements. It also should be remembered that, with multiple VAX processors using one disk, it is relatively easy to reach the I/O capacity of the drive, resulting in a system bottleneck. Further more, should a system disk fail, all systems booting from it fail.

DECnet

With VMS V4.0, DECnet software must be available in a VAXcluster. This is convenient for the system manager because the cluster can be controlled from one terminal, thanks to the virtual terminal capability that DECnet provides. Each cluster node name and number must be the same as the DECnet node name and number. Each DECnet cluster node also should be a full-routing node because DECnet utilizes the CI as a point-to-point configuration.

Cluster Operations

All operator messages generated by the system or by users are dispatched to all operator terminals in the cluster, regardless of the VAX processor to which the operator terminal is connected.

- INSTALLATION OF A VAXcluster:

This section is an overview of how to install a VAXcluster system. Details may be found in the VMS V4.0 documentation. Digital recommends that each VAX processor node be set up individually. First, install VMS V4.0 and then DECnet on each of the VAX processor nodes. Second, boot each VAX processor node separately and run it in a single-node environment. When satisfied that everything is set up correctly on the VAX node, close it down and move on to the next one.

Once all the VAX processor nodes have been set up individually, they may be booted together. At that point, a VAXcluster will be formed. The system manager should then adjust the QUORUM parameter to the appropriate value for the cluster while booting the cluster.

- UPGRADING A RUNNING VAXcluster

  An HSC50 or VAX processor may be added to a running VAXcluster system without interrupting the cluster.

  A DSA disk or tape also may be added to an HSC50 while the cluster is running, assuming that enough SDI or STI interfaces are available. When the HSC50 detects a new device, it informs the VAX processors in the cluster that a new device is available. The VMS operating system on each VAX processor then dynamically adds to its I/O database. The device may then be mounted.

  More HSC50s may be added to a VAXcluster. The CI cables are attached to the Star Coupler, as long as enough connectors are available in the Star Coupler. The Star Coupler has an 8- and 16-node configuration. Adding another HSC50 or VAX processor may require upgrading the Star Coupler to a 16-node configuration. With adequate connections, once the HSC50 is added and booted, each VAX processor can detect the new HSC50 and any disks connected to it. The cluster will continue to function normally.

Chapter 6 ▪ VAX Networking and Communication Software

▪ Overview

Digital offers you a range of products to link processes (or tasks) running on a VMS operating systems, other Digital operating systems, or other manufacturer's operating systems. This capability allows many computers to operate together in data communications networks for distributed processing.

Specifically, VAX/VMS systems are connected to other systems in three ways: VAX to other Digital systems (DECnet-VAX software products), VAX to other manufacturer's systems (DECnet-VAX Internet software products), and VAX to public packet switched networks (VAX PSI software product).

Topics include:

- Digital Network Architecture
- Description of a DECnet-VAX network
- DECnet-VAX Phase IV features
- Internet products
- PACKNET products
- Procedures for programming over DECnet

▪ Introduction

DECnet is a family of software and hardware communications products that enable Digital operating systems to participate in a network environment. A network is an entity of two or more computer systems, called nodes, connected by physical links (cable, microwave, or satellites, for example) for the purpose of exchanging data and sharing resources.

Communication circuits operate over these physical lines. A circuit is a communications data path over which all input and output between nodes takes place. Communication between individual processes, on different systems, takes place through logical links. A logical link is a connection between two processes at the user level. A circuit can support many logical links, all communicating at the same time.

In a network of more than two nodes, the process of directing a data message is called routing. A very important feature of a DECnet network is that its nodes are permitted to route messages through the most cost-effective path, if a circuit is disabled, nodes can seek alternative routes.

A DECnet multinode network is decentralized; that is, many nodes connected to the network can communicate with each other without having to go through a central node. There is no hierarchy; all nodes are considered "peers", each of which do not depend on the availability of a node on a higher level. As a member of a multi-node network, your node can communicate with any other network node, not just the nodes that are physically adjacent to you, and gain access to software to software facilities that may not exist on your local node. Therefore, different applications running on separate nodes can share the facilities of any other node.

Phase IV DECnet supports large and small networks with single and multiple area networks. In a single area network, a maximum of 1023 nodes is possible. The optimum number, however, is much smaller, perhaps 200 to 300 nodes depending on physical relationship of the nodes. In a multiple area network, as many as 63 subnetworks (single area networks) of 1023 nodes each can be connected.

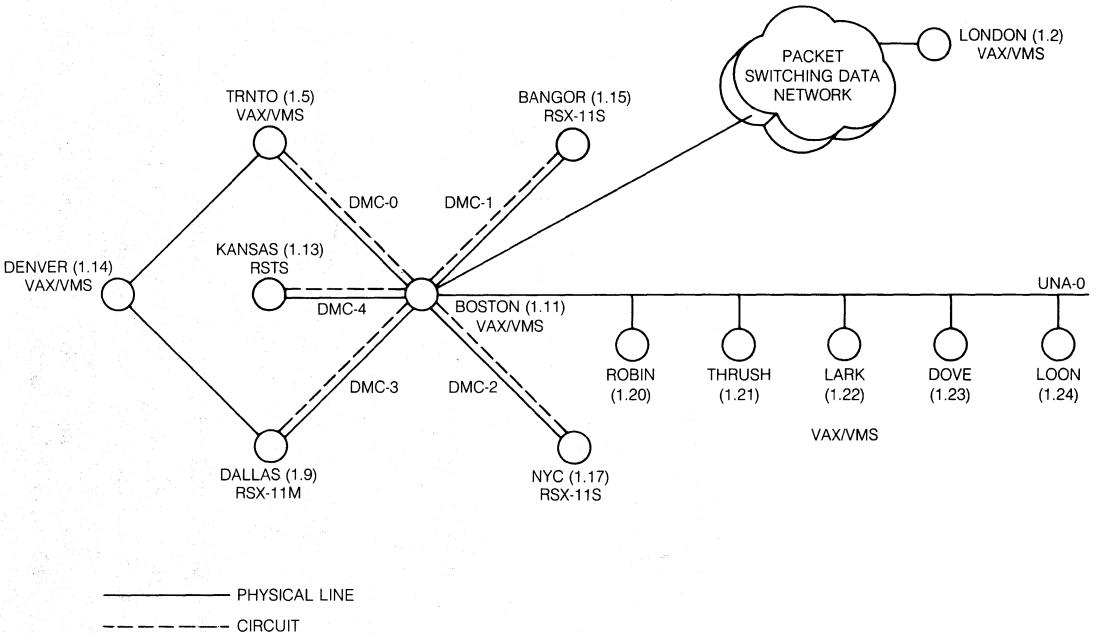


Figure 6-1 ▪ Typical Single-Area DECnet Network

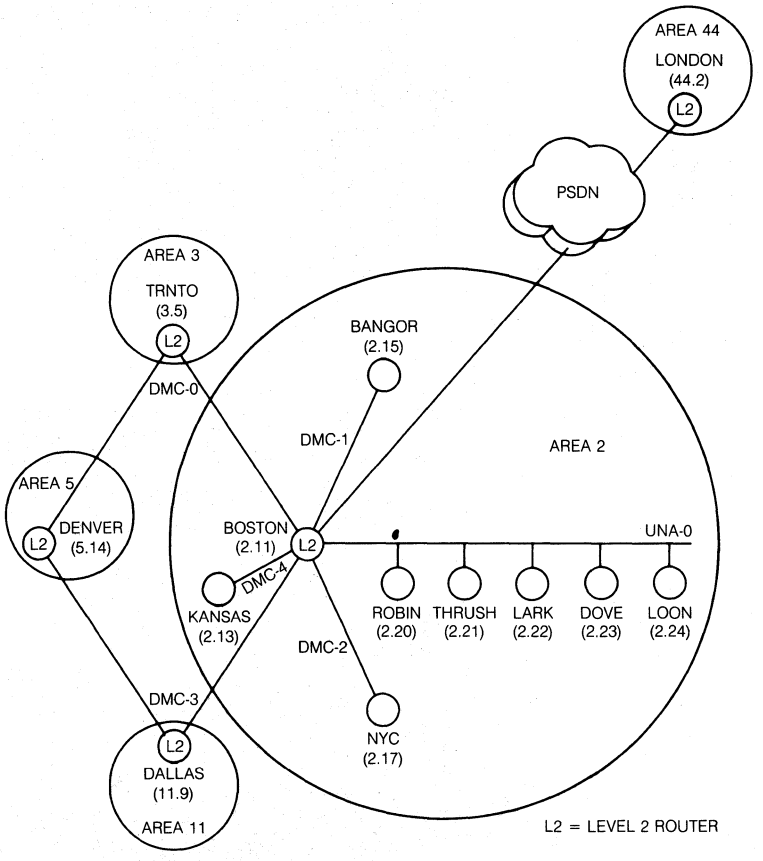


Figure 6-2 ▪ Multiple-Area DECnet Network

▪ Overview of Digital's Network Architecture

The Digital Network Architecture, DNA, is the framework for all Digital communications products. DECnet, DECnet/SNA Gateway, communications servers are among many of these products. The International Standards Organization (ISO) has created an architectural model for open systems interconnection. DNA is closely based on the ISO model.

Table 6-1 ▪ Architectural Layers of DNA and Their Associated Functions

DECnet Functions		DNA Layers	DNA Protocols
File Access Command Terminals Host Services Network Control Program-to-program	User N E T M A N A G	Network Application	User Protocols Data Access Protocol (DAP) and others
		Session Control	Session Control Protocol
		End Communication	Network Services Protocol (NSP)
Adaptive Routing	E	Routing	Routing Protocol
Host Services Packet Transmission/Reception	M E N T	Data Link	DDCMP; ETHERNET; CI; X.25
		Physical Link	sync; async; ETHERNET

A hierarchical, layered architecture like DNA offers flexibility by allowing for the substitution and addition of new technologies in future phases. This means that a customer's application level software investment is protected while Digital adds new products and capabilities from one phase of DNA to another.

NETWORKING ENVIRONMENTS - There are basically two types of networking environments: those involving communications strictly between Digital systems (Digital-Digital) and those involving communication between Digital systems and other vendor's systems (Digital-to-non-digital). Digital therefore offers products that allow communications between Digital Systems via DECnet, products to connect Digital systems to IBM and other vendors' systems (Internet), and products to connect Digital systems to other vendors' systems via X.25 packet-switched data networks (Packnet). Both local and remote communications are possible.

Table 6-2 defines the DECnet software products used to connect VAX systems to other systems as mentioned above.

Table 6-2 ▪ Digital's VAX Networking Product Set

Type Of Communication	VAX Software Communications Products	LAN Communications Subsystem Servers
Digital-to-Digital Host Communication		
Local Traditional	All DECnet Software	N/A
LAN	DECnet-VAX	DECnet Router/ DECnet Router/ X.25 Gateway
REMOTE	All DECnet Software	DECnet Router/ DECnet Router/ X.25 Gateway
Digital-to-non-Digital Host Communications	All Internets VAX PSI	DECnet/SNA Gateway, DECnet Router/ X.25 Gateway
Terminal Connection	CX/DX/AX poly-COM	Terminal Server

As shown in Table 6-2, Digital offers a broad range of networking products — from basic communication devices for our OEMs to full networks complete with network application programs for commercial business customers.

DNA Phase IV and Ethernet — The incorporation of the Ethernet local area network technology into DNA Phase IV demonstrates Digital's commitment to providing a complete set of products to address local area networking needs. The existing DECnet product line already offers comprehensive, wide-area support. Ethernet products improve resource sharing by increasing performance of data traffic in a local area and by connecting to other local and remote communications networks. Ethernet, a par of DECnet, offers remote or long-haul connections and local connections as one integral network. Ethernet is specifically for the types of communications and resource sharing that take place in a local work environment, most often a room, a building, or a complex or buildings.

▪ Types of Networks

Digital communications and networking software products make three types of networks available to VAX/VMS users.

- Communications networks
- Resource-sharing networks
- Distributed computing networks

Communications Networks

These networks exist to move data from one, often distant, physical location to another. The data may be file-oriented (as is often the case for remote job entry systems) or record-oriented (as occurs with the concentration of interactive terminal data). Interfaces to common carriers, using both switched and leased-line facilities, are normally part of such networks. Such networks are often characterized by the concentration of all user applications programs and databases on one or two large host systems in the network. Figure 6-3 illustrates such a network.

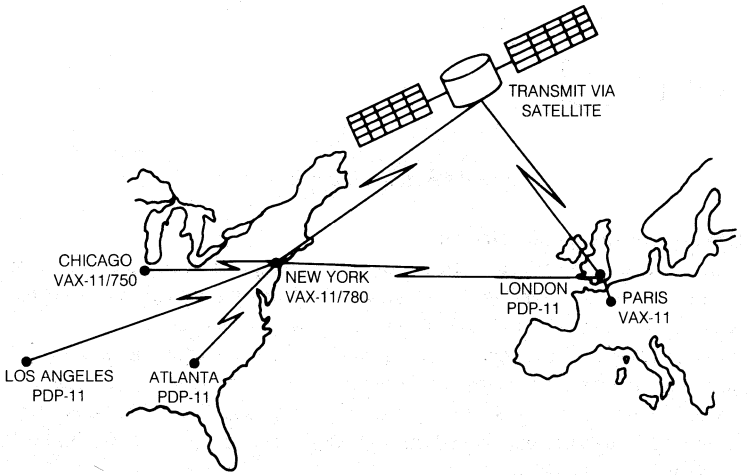


Figure 6-3 ▪ Communications Network

Resource-Sharing Networks

These networks exist to permit sharing expensive computer resources among several computer systems. Shared resources can include not only peripherals such as mass storage device, but also logical entities, such as centralized databases available to other systems in the network. These networks are often characterized by the concentration of high-performance peripherals, extensive databases, and large programs on one or two host systems in the network. Typically, the satellite system have less expensive peripherals and smaller programs. Figure 6-4 illustrates a resource-sharing network.

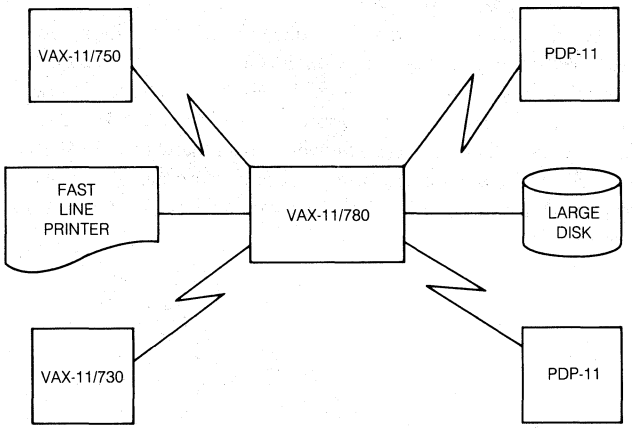


Figure 6-4 ▪ Resource Sharing Network

Distributed Computing Network

These networks coordinate activities of several independent computing system and exchange data among them. Networks of this nature can have specific geometries (star, ring, hierarchy), but often have no regular arrangement of links and nodes. These networks are usually configured so that the resources of a system are close to the users of those resources. Distributed computing networks are usually characterized by multiple computers with applications programs and databases distributed throughout the network. Figures 6-5 and 6-6 illustrate two such networks.

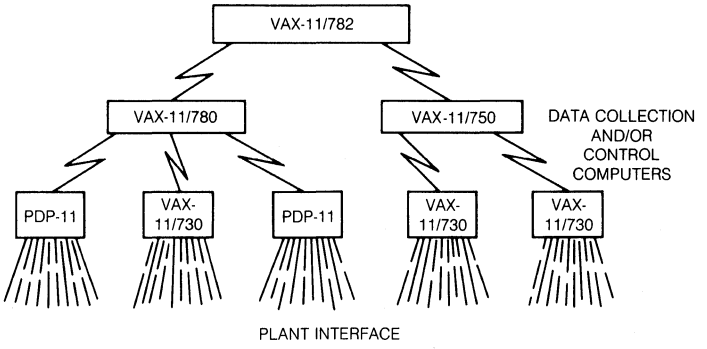


Figure 6-5 ▪ Typical Manufacturing Network

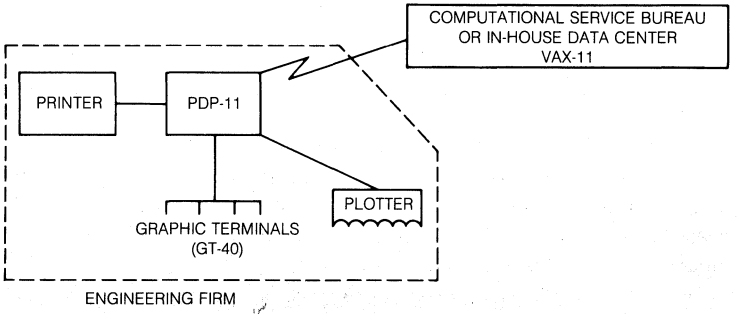


Figure 6-6 ▪ Computational Network

▪ Network Components and Characteristics

To establish a VMS operating system as part of a DECnet communications network, you must build and maintain a network configuration database, which consists of records that describe the specific network components your particular system requires. This section discusses various DECnet-VAX components and their characteristics:

- Nodes, DTEs, and X.25 Protocol Module
- Circuits
- Lines
- Routing
- Logical Links
- Objects
- Logging
- Network access control

NODES, DTEs, and the X.25 Protocol Module

A node is an operating system that uses DECnet software to communicate with other operating systems across a network. A VMS node uses DECnet-VAX software to communicate with other DECnet-VAX nodes and with other Digital operating systems that support DECnet.

A VMS operating system uses DECnet-VAX Internet products to communicate with other manufactures operating systems and VAX PSI software products to communicate with other Digital operating systems via a Packet Switching Data Network.

The equivalent of a node in a Packet Switching Data Network is a DTE (Data Terminal Equipment). A DTE is a computer or terminal that uses VAX PSI software to communicate with remote nodes across a PSDN (Packet Switching Data Network). You can configure your DECnet-VAX node as a DTE if VAX PSI software is installed.

NODES - Nodes can be generally classified by their physical relationship to the user (local or remote), or by their ability to receive and/or transmit data message throughout a network.

HOST NODE - A node provides services for another node.

LOCAL AND REMOTE NODES - A local node is the VMS operating system your user account resides on. Other nodes in the same network your local node is attached are called remote nodes.

ROUTING AND END NODES - Nodes can also be classified by the function they perform in a network. Nodes that accept and forward messages intended for other nodes in a network are called routing (full-routing) nodes. Nodes that only accept messages are called end nodes (non-routing).

CONNECTOR NODE - If you have VAX PSI software in multi-host configuration installed on your local node and you are on an Ethernet, you can configure your node to serve as a connector node. This is a gateway to a Packet Switching Data Network (PSDN) for host nodes on an Ethernet.

Types of DECnet Nodes - DECnet supports a variety of types of nodes developed during different phases of DNA implementation. Phase II, III, IV nodes can all exist on a network. The following types of nodes can be configured as being adjacent to each other on the network:

- Phase II/Phase II
- Phase II/Phase III
- Phase III/Phase III
- Phase III/Phase IV
- Phase IV/Phase IV

Phase II nodes can communicate with each other as long as there is a physical data link between. They support only point-to-point connections. There is no Phase II support for Ethernet.

With Phase III, DECnet introduced adaptive routing, which allows a reasonably large number of nodes to communicate conveniently. There is no Phase III support for Ethernet.

Phase IV DECnet permits the configuration of very large networks and expands the types of data links available for use. Phase IV supports routing, which allows configuration of a network of up to 63 areas. Phase IV software supports Ethernet circuits, as well as DDCMP, CI and X.25 circuits.

Phase IV nodes can be one of two kinds: routing nodes (routers) or end nodes (nonrouters).

- Phase IV routing nodes. These nodes deliver packets to and receive packets from other nodes, and route packets from other source nodes through to destination nodes. The use Ethernet, DDCMP, X.25, and CI circuits. In an area network configuration, Phase IV routers exist at two routing levels:
  - The level 1 router, which performs routing within a single area. The node type is ROUTING IV.
  - The level 2 router, which performs routing within its own area and to and from other areas. The node type is AREA.
- Phase IV nonrouting nodes (end nodes). These nodes deliver packets to other nodes and receive packets from other nodes, but do route packets through. They can be attached to an Ethernet, DDCMP, X.25, or CI circuit. The node type is NONROUTING IV.

Other routing and nonrouting nodes are:

- Phase III routers. These nodes deliver packets to and receive packets from other nodes, and route packets from other destination nodes whose addresses are less than 256. They use DDCMP, X.25, and CI circuits, but do not support Ethernet circuits.
- Phase II nodes. These nodes can send packets to adjacent PHASE III routers or to other adjacent Phase II nodes. However, Phase II nodes can send packets only in point-to-point configurations. Phase III nodes cannot communicate with a Phase II node through another Phase III node.

DTEs - A VMS operating system connected to a Packet Switching Data Network (PSDN), with VAX PSI software installed, can function as a DTE. To configure a local DTE, you must establish a X.25 protocol module entry in your configuration database.

You can use the connector node, gateway, if you have VAX PSI software in multi-host mode installed on your local node, and if you are on Ethernet. If your node is on an Ethernet with a multi-host connector on it, you can use the connector node to access a PSDN without being configured as a DTE, provided you have VAX PSI Access software installed.

X.25 PROTOCOL MODULE - The X.25 protocol module is a software component that controls the transmission of data packets over the PSDN. The configuration database for the X.25 module identifies the PSDN your DTE is connected to, and specifies user groups associated with the DTE.

Circuits

Circuits are high-level communications data paths between nodes or DTEs. Circuits operate over physical lines (see below), which are low-level communications paths. DECnet-VAX employs four classes of circuits:

- DDCMP circuits
- CI circuits
- Ethernet circuits
- X.25 circuits

DDCMP CIRCUITS - DDCMP circuits provide the logical, point-to-point, or multi-point connection between two or more nodes. A point-to-point circuit operates over a corresponding synchronous or asynchronous DDCMP point-to-point line. Multi-point control circuits operate over synchronous DDCMP control lines.

CI CIRCUITS - CI circuits are available only on those node which are attached to a CI-780 or CI-750 Interconnect. CI circuits are similar in many ways to DDCMP multi-point circuits, but use their own protocol.

ETHERNET CIRCUITS - Ethernet circuits provide for a multi-access connection between a number of nodes on the same circuit. An Ethernet circuit differs from other DECnet circuits in that there is not a single node at the other end. An Ethernet circuit is a path to many nodes. Each node on a single Ethernet circuit is considered as being adjacent to every other node on the circuit and equally accessible.

X.25 CIRCUITS - X.25 circuits use the X.25 level 3 protocol (the packet level), and provide for communication over the Packet Switching Data Network (PSDN). X.25 circuits are made available to PSI application programs (often called, "native X.25 user programs) through VAX PSI software, or DECnet data link mapping (DLM). DLM permits the use of X.25 as a data link through the mapping of data link information between the DECnet routing layer and the X.25 native circuits and X.25 DML circuits.

Lines

Lines provide physical communications and are the lowest level communications path. A line is a physical data path from a DECnet node to another node or from a gateway to an IBM SNA network or packet-switched data network.

DECnet-VAX supports four classes of lines:

- DDCMP lines
- CI lines
- Ethernet lines
- X.25 lines

DDCMP LINES - A DDCMP line provides the physical point-to-point or multi-point connection between two or more nodes.

CI LINES - A CI line provides a high-speed connection between two or more nodes.

ETHERNET LINES - An Ethernet line is a multi-access connection between two or more nodes.

X.25 LINES - An X.25 line is the physical link between your DTE and the PSDN.

Network Routing

Routing is the network function that determines the path or route along which data travels. The Routing layer of DECnet handles routing functions.

ROUTING NODES - Routing nodes (routers) are nodes that can send and receive data from one node to another. Routers have two or more circuits. Routers regularly receive and maintain information about other nodes in the network. They perform the routing operation by associating a circuit with the destination node for the packet over that circuit with the destination node for the packet and transmitting that packet over that circuit. Routers can use DDCMP, CI, Ethernet, or X.25 circuits as their data links.

Logical Links

DECnet uses a mechanism called a logical link to allow communications between processes running on either the same node on or separate nodes in the network. A logical link carries a stream (consisting of regular and interrupt data) of full-duplex traffic between two user-level processes. Each logical link is a temporary data path that exists until one of the two processes terminates the connection.

Objects

Objects provide known general-purpose network services. An object is identified by object type, which is a discrete numeric identifier for either a user task or a DECnet program such as NML or FAL. The DECnet network software uses object type numbers to enable logical link communications with NSP.

There are two general types of DECnet-VAX objects:

- Objects with a 0 object value. These objects are usually user defined images for special-purpose applications. They are named when a user requests a connection.
- Nonzero objects. Nonzero objects are known objects that provide a specific network services such a file access (FAL) or network management (NML). They may also be user-defined tasks; these objects should be for user-supplied, known services.

The following Digital-supplied objects are defined inside the network configuration database.

- File Access Listener (FAL). FAL is an image that provides authorized access to the file system of a DECnet node on behalf of processes executing on any node in the network. FAL communicates with the initiating node by means of the Data Access Protocol (DAP).
- Network Management Listener (NML) NML is an image that provides services such as gathering and reporting information about network status, zeroing line and node counters, and loading a stand-alone system image to a remote node.
- Event Logger (EVL). EVL is an image that logs significant events (logically or remotely) for a give network component.
- Loopback Mirror (MIRROR) Mirror is an image that is used for particular forms of loopback testing.
- DECnet Test Receiver (DTR). DTR is a DECnet test program that is used with the DECnet Test Sender (DTS) to test the logical links.
- Mail. Mail is an image that provides personal mail service for VMS nodes.
- Phone. Phone is an image that allows you to have interactive conversations with users on VMS.
- Host Loader (HLD). HLD is an image that provides downline task-loading support for RSX-11S tasks.

VAX PSI OBJECTS - The object component of VAX PSI contains records that identify the object, specify a command procedure that is initiated when the incoming call arrives, and specify account information for the incoming call.

X.25 and X.29 Server Modules

To handle calls coming in over a PSDN from remote DTEs and terminals, X.25 and X.29 server modules (call handlers) are required. The X.25 server module handles incoming calls that originated at a remote DTE; the X.29 server module handles incoming calls that originated at a remote terminal. Your local DECnet-VAX node can receive X.25 and X.29 calls if it is configured as a:

- DTE connected directly to a PSDN,
- A multi-host connector node that forwards calls between a PSDN and host nodes on an Ethernet
- A host node on an Ethernet that uses a connector node to send and receive X.25 and X.29 calls

X.25 Access Module

The X.25 access module provides a means for user processes on VMS host nodes to access remote nodes or terminals connected to a PSDN through a VMS node serving as a multi-host connector node. Both the host node and the connector node must be on an Ethernet. The host node must be configured with VAX PSI Access software. The connector node must be configured with VAX PSI software in multi-host mode.

The X.25 access module identifies the connector node to which the local node is connected, the network accessed by the connector node and, and optionally, access control information. The DECnet-VAX host system with VAX PSI Access software uses a DECnet link to connect to the connector node. VAX PSI Access software uses the link to transmit X.25/X.29 messages between the host and the connector node.

Logging

Network software logs significant events that occur during network operation. Included among these events are:

- Circuit and node counter activity
- Changes in circuit, line, and node states
- Service requests (for example, when a circuit or line is put in an automatic service state).
- Passive loopback (this occurs when the executor node is looping back test messages)
- Routing performance and error counters (circuit, line, node, and data packet transmission)
- Data transmission performance and error counters
- Lost event reporting

This information can be useful for maintaining the network because it can be recorded continuously by the event logger.

Network Access Control

DECnet-VAX regulates access to the network at various levels, including:

- Routing initialization passwords for links connecting the local node to remote nodes
- System-level access control for inbound logical link connections that result in a process being created
- Node-level access control for inbound and outbound logical links
- Proxy login access control for individual accounts

ROUTING-INITIALIZATION PASSWORDS - Whenever a communications circuit is established, the local node in the circuit attempts to initialize with the DECnet software at the remote node connection for that circuit. As part of this initialization, the remote node may require a password to complete the operation. Passwords are an optional feature that the system manager can use when setting up the configuration database. Generally, password are used only when a system has dialup telephone lines used by the network.

SYSTEM-LEVEL ACCESS CONTROL - DECnet-VAX provides system-level access control over logical link connections. The network user on the initiating node may explicitly supply an access control string to control which account is used on the remote node. If, however, the initiating node does not supply explicit access control information, DECnet optionally provides default access control when sending the request to the remote node. It also optionally provides default access control for incoming logical links if the initiating node has not supplied access control information.

SETTING ACCESS CONTROL INFORMATION FOR OUTBOUND CONNECTS - The system manager can specify default access control information for outbound connections. This enables the local node to send outbound logical link requests with default access control information when that information is not explicitly provided. The remote node stores the access control information in its configuration database. The default access control information can include privileged and nonprivileged names and passwords to be used in connecting to a particular remote node.

The system manager at a node can specify a list of privileges required for connection to a particular object such as NML. When the local node requests connection to an object for which privileges have been specified, it sends the default privileged access control string to the remote node. If the system manager does not specify privileges for an object, such as FAL, the object is accessible to all users. When the local node requests connection to this object, it sends the nonprivileged access control string.

SOURCE OF ACCESS CONTROL INFORMATION FOR LOGICAL LINK CONNECTIONS - Whenever a local DECnet node attempts to connect to a remote DECnet-VAX node by means of a logical link, system-level access control information is sent to the LOGINOUT image running in the context of a process on the remote node. Access control information can come from a number of sources:

- Access control string explicitly supplied by network user
- Nonprivileged default access control string
- Privileged default access control string
- Proxy login access control
- Null access control string
- Default inbound access control string
- Executor node, nonprivileged access control string

Finally, if none of these sources supplies the information, the connection fails.

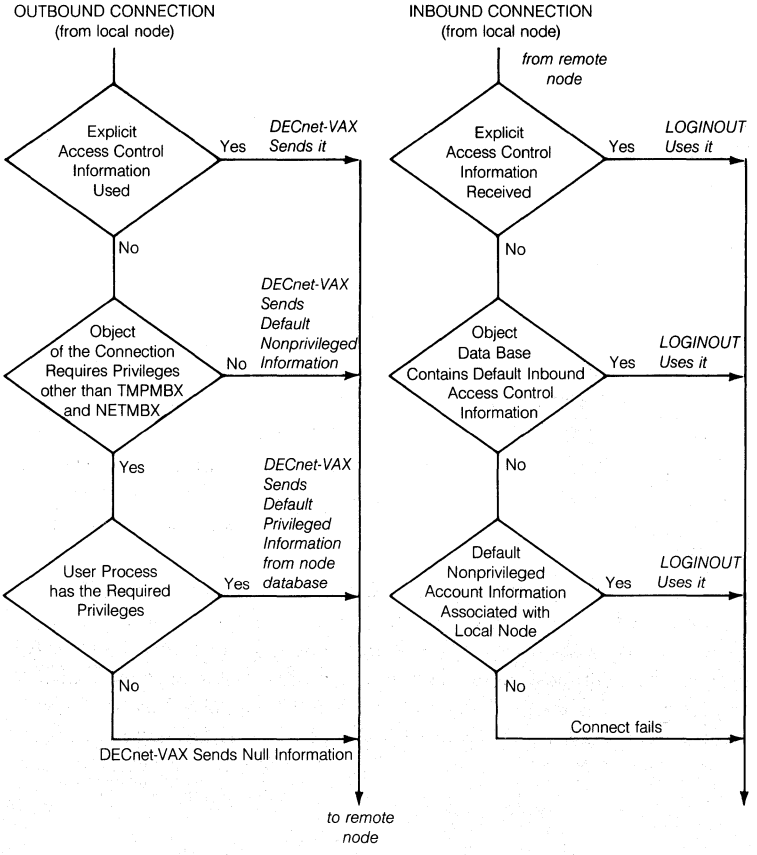


Figure 6-7 ▪ Access Control For Inbound Connections

▪ VAX/VMS Communication Software: DECnet-VAX and VAX PSI

DECnet-VAX and VAX Packetnet System Interface (PSI) networking software can be configured on all VMS operating systems and all MicroVMS operating systems.

DECnet-VAX is the implementation of DECnet which causes A VMS operating system to function as a network node. As the VMS network interface, DECnet-VAX supports both the protocols necessary for communicating over the network and the functions necessary for configuring, controlling, and monitoring the network.

Under DECnet-VAX, Digital's Internet family of products supports the interconnection of Digital's computers and networks to systems built by other manufacturers. Internet products are tools for distributed data processing in a multi-vendor environment. As parts of powerful VAX systems, they provide transparent communications with the equipment of other vendors, and at the same time, offer the flexibility of local file systems, many different programming language, and wide selection of computing power.

Digital's IBM Internet product line can be viewed from two different perspectives, that is, either as end-user functions or as networking technologies (System Network Architecture, SNA, and Bisynchronous Systems Communications, BSC).

VAX PSI is the software product that allows the VAX/VMS user to participate in a packet switching environment. VAX PSI implements the X.25 and X.29 recommendations providing a user interface to a Packet Switching Data Network (PSDN). A PSDN consists of switching nodes connected by high-speed links.

You can use VAX PSI to line DECnet nodes across a PSDN through data link mapping (DLM), which permits an X.25 virtual circuit to be used as a DECnet data link. You can also attach your computer or terminal directly to the PSDN and communicate over an X.25 virtual circuit with a remote node connected to the PSDN.

Another way in which you can communicate with a remote node over PSDN is through and X.25 multi-host connector mode. The host node that uses the services of the connector node is a VAX/VMS system configured with VAX PSI software in the multi-host mode. The host node that uses the services of the connector node is a VAX/VMS system configured with VAX Access software. Alternatively, your VAX/VMS node can use a server-based X.25 gateway node to communicate over a PSDN, provided VAX XEP software is installed on your node.

▪ DECnet-VAX Configurations

DECnet supports the following network connections:

- A connection to an Ethernet circuit in a local area network configuration
- point-to-point or multi-point connections to a node running DECnet using the Digital Data Communications Message Protocol (DDCMP)
- A connection over the Computer Interconnect (CI) to another node running DECnet.
- An X.25 connection to a node running DECnet (either a direct connection over a PSDN or a connection made by means of an X.25 multi-host connector node that accesses a PSDN).

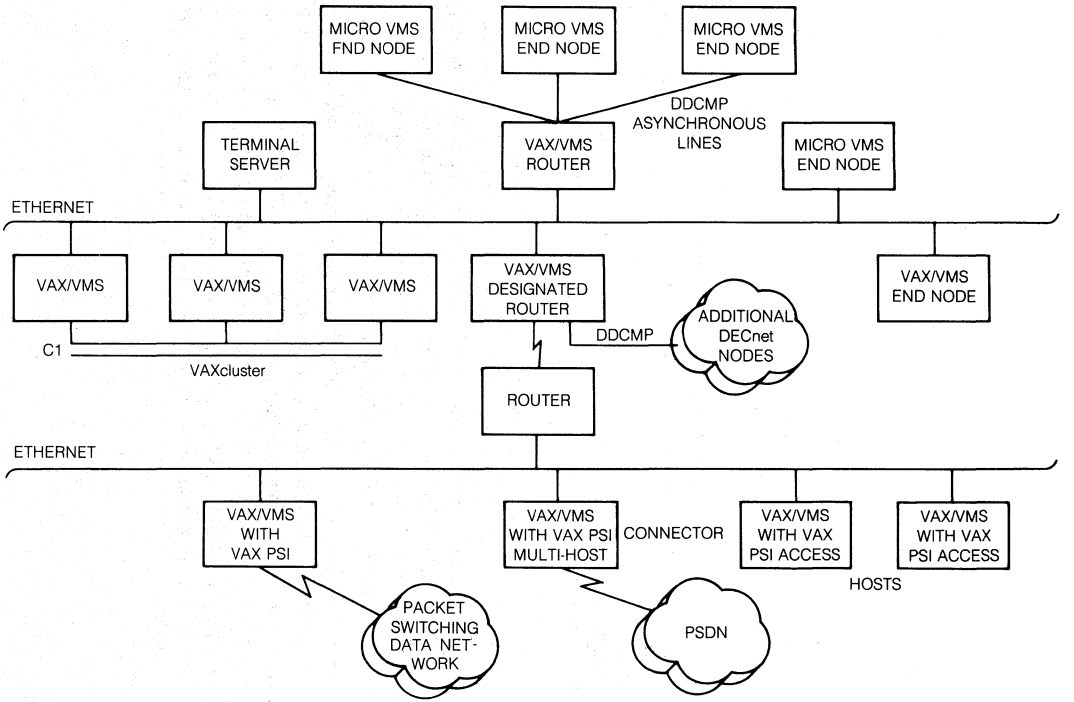


Figure 6-8 ▪ Sample DECnet Phase IV Configuration

Figure 6-8 is a sample DECnet-VAX Phase IV configuration illustrating various DECnet-VAX nodes (that is, MicroVMS end nodes, VMS routers, and VMS end nodes in a VAXcluster) connected to an Ethernet, and two Ethernets connected by routers. This figure also shows the use of a DDCMP synchronous line to connect a router to additional DECnet

DECnet-VAX Ethernet Local Area Network (LANS) Configuration

Ethernet is a local area network component that provides a reliable high-speed communications channel, optimized to connect information processing equipment in a limited geographic area, such as an office, a building, or a complex of buildings (a campus, for example).

Local area networks (LANS) are designed for a wide variety of technologies and arranged in many configurations. Digital Equipment Corporation, Intel Corporation, and Xerox Corporation collaborated in producing the Ethernet specification to develop a variety of LAN products. Digital's implementation of the Ethernet specification that was originated by the Xerox Corporation appears at the lowest two levels of the overall DNA specification: The Physical layer and Data Link Layer.

Ethernet circuit devices supported by DECnet are the DEUNA and DEQNA software products. VMS uses DEUNA and microVMS uses DEQNA.

DDCMP Point-To-Point And Multi-Point Network Configurations

Digital Data Communication Message Protocol (DDCMP) provides a low-level communications path between systems. DDCMP performs the basic communications function of moving information blocks over an unreliable communications channel. DDCMP is also used to manage the orderly transmission and reception of blocks on channels with one or more transmitters and receivers.

Figure 6-9 illustrates DDCMP point-to-point and multi-point configurations. A point-to-point and multi-point configuration. A point-to-point configuration consists of two systems connected by a single communication channel. A multipoint configuration consists of two or more systems connected by a communications channel, with one of the systems, called the control station, controlling the channel. All other systems on the communications channel are known as tributaries.

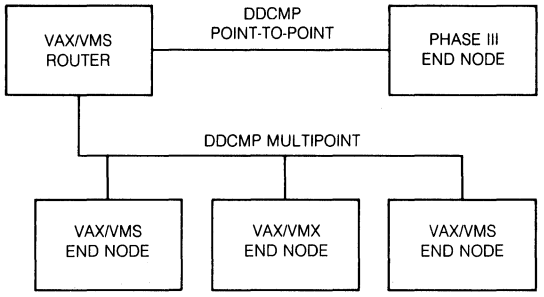


Figure 6-9 ▪ Typical DDCMP Point-To-Point Connections

DECnet-VAX Configurations For VAXclusters

A VAXcluster is an organization of VAX systems which communicate over a high-speed communications path, the CI, and share processor resources as well as disk storage. Figure 6-10 shows a typical VAXcluster. The CI is the physical link between the nodes in a VAXcluster. CI cables from the individual nodes in the cluster are connected to a Star Coupler.

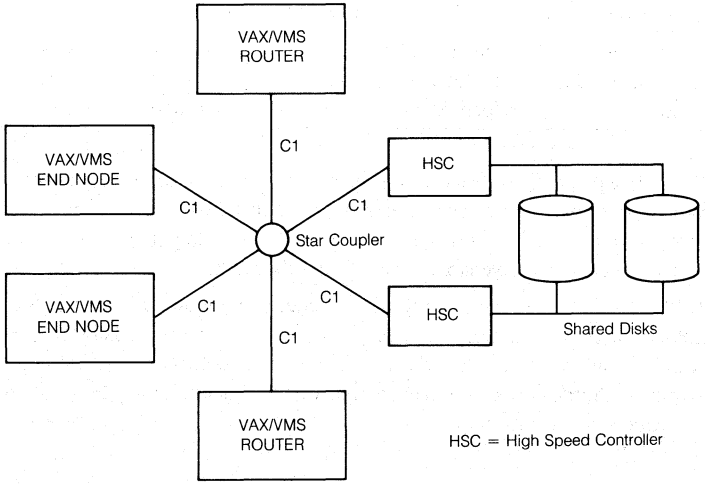


Figure 6-10 ▪ A Typical VAXcluster Configuration Using CI As A Data Link

DECnet-VAX connections are required for all VAX/VMS systems in the VAXcluster. Use of DECnet-VAX insures that VAXcluster system managers can access each node in the cluster from a single terminal, even if terminal switching facilities are not available. DECnet-VAX is also required by the User Environmental Test Package (UETP).

The choices for DECnet-VAX physical links for use in the VAXcluster are:

- Connecting each VAX/VMS node in the cluster to an Ethernet (as shown in Figure 6-10.
- Using the CI that connects the cluster nodes as the DECnet-VAX data link (as shown in Figure 6-3)

Connecting each VAXcluster node to an Ethernet provides distinct advantages:

- Each node in the cluster can be an end node, resulting in lower overhead for these nodes, decreased routing traffic throughout the network, and simpler installation procedures. Note their must be at least one router on the Ethernet to which the cluster end nodes are
- Ethernet provides for better performance in DECnet transmissions than the CI, because of the available communications protocols.
- Terminal servers can be used when nodes in a VAXcluster are connected to an Ethernet. Digital's terminal servers offer a number of benefits to the VAXcluster user, such as load balancing and easier cluster management.

If only one physical link is used to connect each cluster node to the network, the Ethernet link should be used instead of the CI data link, because DECnet performs better through Ethernet. In this case, the CI should perform the functions of a system and not be enabled as a DECnet data link.

A VAXcluster node connected to an Ethernet may require additional DECnet links in order to communicate with remote nodes not on the Ethernet, or to communicate with a MicroVMS system over an asynchronous DDCMP line. A VAXcluster node connected to more than one DECnet link must be configured as a router, not an end node.

If the nodes in a VAXcluster are not connected to an Ethernet, the CI should be used as the DECnet data link between the nodes. CI circuit devices are treated as though they were multi-point devices. but each node on the CI can talk directly to every other node and on polling is involved.

A two-node VAXcluster that uses the CI as the data link can be configured using end nodes, but at least one router is required if additional nodes are configured in the cluster. The CI does not have a needed so nodes in the cluster can identify each other. If the router in a three-node cluster fails, the cluster reverts to being a two-node cluster, which can consists of the end nodes only. For a cluster of four or more nodes, however, more than one router is required in order to prevent the loss of communications capability between the remaining nodes if the router fails.

X.25 Network Configurations

Packet switching data networks provide fast, dependable communications between geographically distributed nodes. Data transmitted over a PSDN is divided into packets each of which has a header containing control and destination information. The PSDN interleaves packets from many users over shared transmission lines and delivers the packets in the correct order to the proper destinations. The routing is invisible to the user.

X.25 AND X.29 PROTOCOL RECOMMENDATIONS FOR PACKNET PRODUCTS - In the 1970's the International Telephone and Telegraph Consultive Committee (CCITT) developed a series of recommendations for standard communication protocols that could be used by PSDNs and other common carriers to provide data communications products. Two of these recommendations, X.25 and X.29, define the interface between the computer and the network and the control of asynchronous terminals connected directly to a network. Together, these protocols define the Interactive Terminal Interface (ITI).

The X.25 recommendation defines three levels of protocol for the ITI interface: Level 1 covers physical and electrical characteristics; Level 2 pertains to link access procedures; and Level 3 pertains to packet procedures.

The X.29 recommendation defines the procedures for information exchange between an asynchronous terminal and the packet assembly/disassembly facility of the PSDN.

Rapidly, X.25 is becoming the standard international communications protocol. As another advantage, X.25 allows computers from different manufactures to work together: with appropriate security validation, any system on the network can send data to any other system on the network. X.25 ensures data integrity, while at the same time relieving users of any concern about input and output of the various processors in the network.

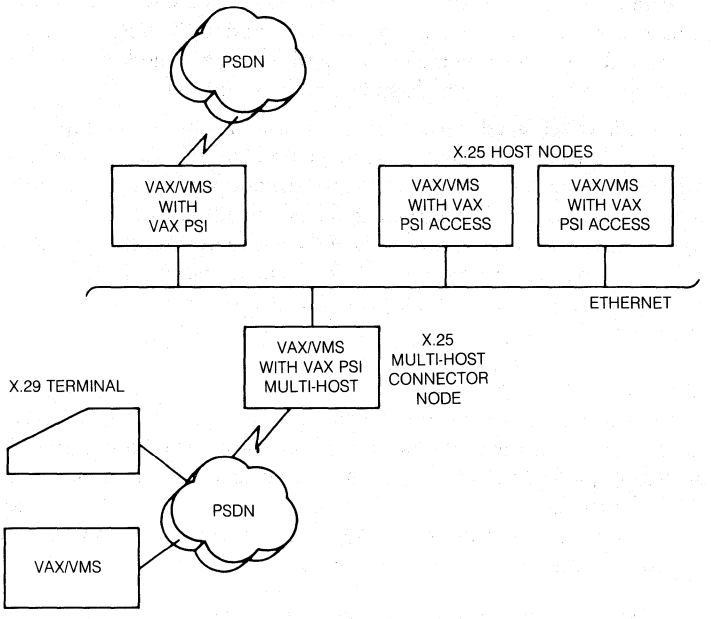


Figure 6-11 ▪ DECnet Network Configuration Showing Examples of X.25 Connections

▪ Performing Network Operations

DECnet-VAX software offers user a variety of operations that can be accomplished over the network. Three of which are:

- Manipulating files on remote nodes (that is, transferring, deleting, or renaming file operations)
- Access remote files at the record level
- Perform task-to-task (interprocess) communications

DECnet-VAX software allows you to access files on remote nodes as though they were on your local node. It also allows you to design applications that communicate with each other over the network. VMS operating system and DECnet-VAX communications software are integrated to provide a high degree of transparency for user operations. For some applications however, its is desirable to have more direct access to network-specific information and operations. For this purpose, DECnet-VAX provide nontransparent communication.

Designing User Applications For Network Operations

DECnet-VAX supports several programming languages for design of network applications with:

- DCL commands and command procedures
- Higher-level language programs
- MACRO programs using RMS service calls or system service calls

You can use the following higher-level languages to develop networking applications:

- VAX FORTRAN
- VAX BASIC
- VAX BLISS
- VAX PASCAL
- VAX C
- VAX PL/I
- VAX COBOL

With any of these languages, you can access remote files and create tasks that exchange information across the network.

Table 6-3 summarizes the normal use of the programming languages for specific network operations that you can perform with DECnet-VAX.

Table 6-3 ▪ Access Levels

User Lang.	Network Operation	Language Calls	Access Level
DCL	Network command terminals	DCL commands	Transparent network access via DCL
	Remote file manipulation		
	Task-to-task communication		
Higher-Level	Remote file access (file and records)	Higher-level language I/O statements	Transparent network access via RMS
MACRO or higher-level	Remote file access (files and records)	RMS service calls	
	Task-to-task communication		
MACRO or higher-level	Task-to-task communication	System service calls	Transparent and nontransparent network access via QIO

The method you choose to access the network is directly related to the language and network operation you wish to perform. For example, you may want to use standard VAX RMS calls in a VAX MACRO program to access remote files, and then use system service calls to communicate between MACRO programs in a task-to-task communication application.

Table 6-3 illustrates the three access levels and the corresponding network operations. The various levels of network access provide a convenient context in which to discuss typical user operations over the network.

The first two access levels, DCL and RMS, are entirely transparent to the network user. Because standard DCL commands and RMS service calls are used, no DECnet-specific calls are required. You need only to specify, in your file specification, the remote node on which the file resides. Likewise, higher-level language tasks can use a variation of the standard VMS file specification in conjunction with standard I/O statements to access remote tasks and exchange information, which also remains transparent to the user. Transparent network access, therefore, allows the user to move data across the network with little concern for the way this operation is performed.

The third level of access is system services. These provide a transparent and nontransparent user interface to the network. Transparent communication at the system service level provides all the basic functions necessary for two tasks to exchange messages over the network. These operations are transparent because they do not require DECnet-specific calls because standard system service calls are used.

This basic set of functions is extended to nontransparent communication by allowing a nontransparent task to receive multiple inbound connections and to use additional network protocol features such as optional user data and interrupt messages.

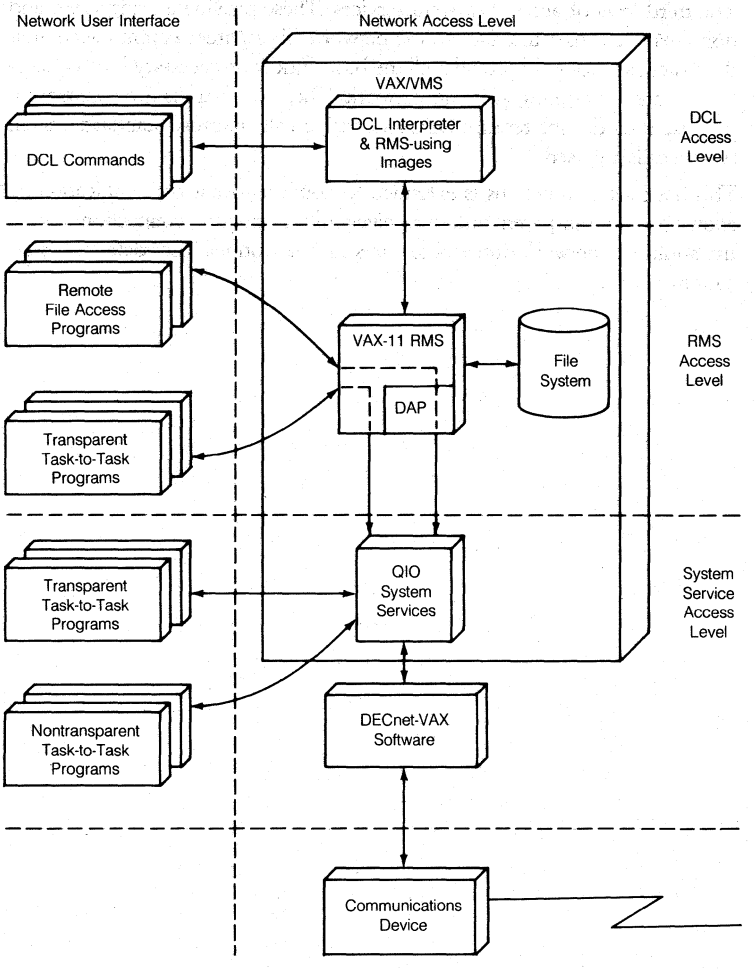


Figure 6-12 ▪ Network Access Levels and DECnet-VAX User Interface

Accessing The Network

Accessing a network through DECnet-VAX software products entails using correct:

- Network file and task specifications
- Access control parameters
- And logical names

USING FILE AND TASK SPECIFICATIONS IN NETWORK APPLICATIONS - When accessing remote files, DECnet-VAX users follow standard VMS file specification format. A node specification string that includes a node name must be present. You can also include an optional access control string in the node specification to specify explicitly the user name and password of specific account to use on the remote system.

For example, the file specification:

```
TRNTO"SMITH JOHN"::WORK.DISK:TEST.DAT;1
```

contains explicit access control information and can be used to access the file TEST.DAT, which resides in user Smith's top-level directory on the device WORK.DISK on node TRNTO.

The following file specification, which does not contain explicit access control information, can also be used to access the remote file TEST.DAT provided that a proxy or default DECnet account exists on the target node:

```
TRNTO::DBA1:[SMITH]TEST.DAT;1
```

Task-to-task communication requires the use of a task specification string enclosed in quotation marks. This string identifies the target task to which you want to connect on a remote node. For example, the following task specification string:

```
BOSTON::"TASK=TASK2"
```

identifies the task TEST 2 by means of the TASK= for of the task specification. You can also use the 0= form to specify a task. For example,

```
BOSTON"JONES KC"::"0=TEST2
```

also identifies the task TEST2. Note that in this case, explicit access control information is also included in the node specification string.

Using Access Control For Network Applications

Access control is the control that a node exercises over inbound and outbound logical link connections. The terms inbound and outbound refer to the direction of the logical link connection request. A node receives and processes inbound requests; it processes and send outbound requests. This distinction is useful for discussing access control as it relates to VAX/VMS nodes in a network.

When DECnet-VAX software sends an outbound connection request in response to either a remote file access or a task-to-task communication operation, certain access control information may be necessary to connect successfully to the remote node and log in. As in logging in at your local VAX/VMS node, you can supply specific access control information in the form of a user name and password that the remote node recognizes. The remote node processes inbound connection requests containing this information to verify that you are a valid user of the system.

Figure 6-13 illustrate the access control processing that takes place for a simple DCL command.

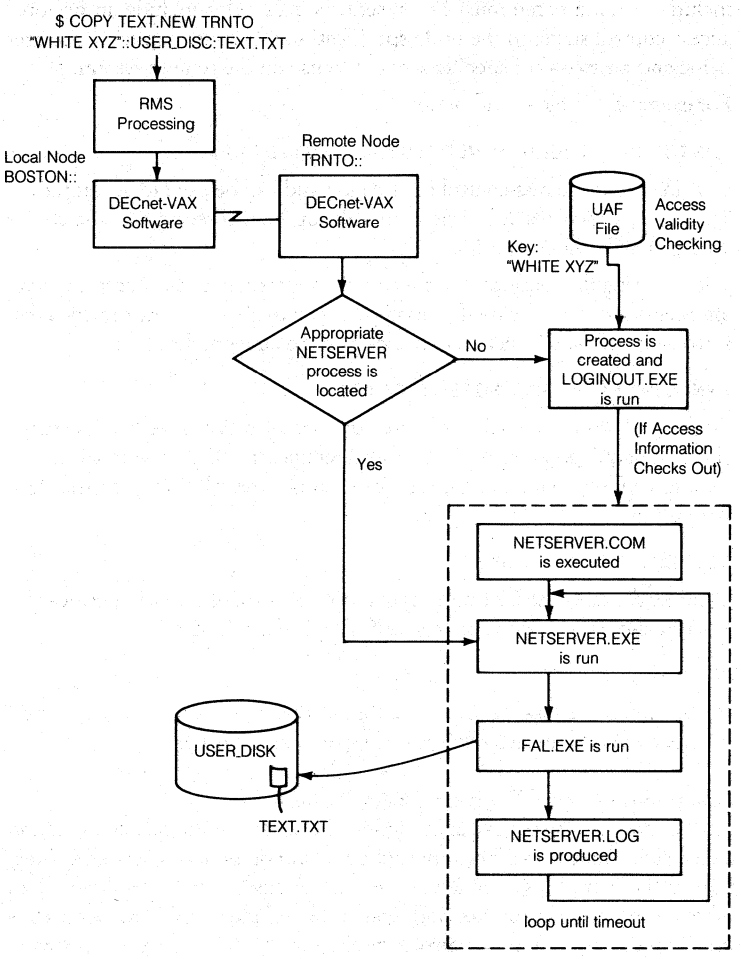


Figure 6-13 ▪ Remote File Access Using Access Control String Information

When explicit access control information is not provided in the connection request, DECnet-VAX software uses the remote node name specified in the connection request as a key to locate the appropriate record in the local configuration database. This record contains default assess control information applicable to the remote node.

Depending on the privileges required by the object to which you want to connect and those of the user process, one of the three possible sets of default access control information is sent to the remote node: default privileged, default nonprivileged, or null. Because these defaults are node parameters, all privileged operations requested with default access control for a given node run under the same default account. The same is true for nonprivileged operations requested with default access control.

If the target node is running DECnet-VAX, it can associate incoming connect request with specific accounts other that the default DECnet account. This type of access is known as proxy login and requires that the originator of the request have a proxy account on the target node and that proxy login access be enabled at that node.

Chapter 7 ▪ The MicroVMS Operating System

▪ Overview

MicroVMS is VMS. It is a modular version of the VMS operating system for use with MicroVAX systems. It has virtually all of the same features, files and utilities as the VAX/VMS operating system. Application programs that run under MicroVMS will run on any VAX/VMS system without modification and applications developed to run under VMS will run on MicroVMS systems.

As a flexible, general purpose operating system, MicroVMS is ideal for a broad range of applications. It can be used on MicroVAX systems in standalone, networked, or distributed systems. Users with such diverse application needs as image processing, numerical control, or small business computing will find only MicroVMS offers the advantage of being able to use VMS layered software. Program developers wishing to design automatic test equipment, expert systems, or telecommunications applications will find the wealth of familiar VMS utilities and design tools allows them to be more productive.

MicroVMS is designed for easy installation and use. It does not require expert assistance to install the operating system or establish accounts. With the installation instructions, just about anyone can install the MicroVMS system. Maintaining the system is also uncomplicated; typically, a programmer using MicroVMS for application development has the knowledge to maintain a system.

As the name MicroVMS implies, the operating system uses *virtual memory management* to permit users to write application programs much larger than physical memory and to allow any number of jobs share memory. To allow the user to get the most from the memory available, only a basic operating system is required to run applications. Additional modules can be added to give users additional utilities, multiuser security, application development, and system software development. For users who require powerful communications capabilities, DECnet-VAX software is also available.

▪ The MicroVMS Modules

MicroVMS is VMS made available in several packages. These packages, or modules, are easier to manage than larger operating systems. MicroVMS is an entirely native-mode system and supports only Q-bus devices.

The MicroVMS Operating System

The operating system contains three component modules: the Base System, Common Utilities, and Multiuser Security. Of these modules, the base system contains all functions necessary to run applications and is the only component that is prerequisite to all other components.

- THE BASE SYSTEM

  The base system contains the minimal components required to install and execute application software.

  The MicroVMS base system includes specific capabilities to facilitate:

  - System start-up; installation of system options, layered products and user applications; program execution.
  - File handling operations such as create, copy, back up, sort, and others; logical name definitions; help and text file creation and maintenance operations.
  - Diskette handling operations such as initialize, mount, and others.
  - Use of command procedures; assignment of values to symbols; manipulation of string data and integers; use of lexical functions to obtain system information, to convert data, and to perform other operations.
  - Multiple account set up; provision of standard UIC protection to files and other objects.
  - Support for additional data devices.

- COMMON UTILITIES

  The following common utilities are available and may be installed individually: EDT, MAIL, HELP, RUNOFF, DIFF, DUMP, and SEARCH. For more information about these utilities, refer to Chapters 3, 4, and 5.

- MULTIUSER SECURITY

  This module provides facilities for regulating and protecting a system containing multiple accounts. With it system managers can control disk quotas, batch and print queues, operator communications, and security classifications. It also includes an accounting utility.

- PROGRAM DEVELOPMENT

  Like VMS, MicroVMS offers programmers an extraordinary environment for application and system software development. The Common Language Environment and a host of programming utilities make MicroVAX systems running MicroVMS an exceptional programmer system.

- APPLICATIONS DEVELOPMENT

  This module is prerequisite to programming languages such as VAX FORTRAN or VAX COBOL. It is also a prerequisite for the System Software Development module.

  The Application Development module contains:

  - A linker for combining and processing object modules (produced by the language processors) to create executable programs.
  - A debugger for interactively examining a program as it executes.
  - Libraries of object modules to perform standard operations and to access system services.
  - A message utility to create properly formatted condition codes and error messages.
  - Utilities to analyze files and to create and maintain indexed files.

- SYSTEMS SOFTWARE DEVELOPMENT

  This module contains languages, utilities, and documentations that permit greater access to system structures. It also lets programmers attach unsupported devices to the system.

  The System Software Development module supports:

  - MACRO language programming.
  - Writing and debugging device drivers (through documentation).
  - Writing privileged shareable images (through documentation).
  - Connection to interrupt vectors (through documentation).
  - Mapping I/O space (through documentation).
  - Changing interrupt priority levels and access modes (through documentation).
  - Patching executable programs.
  - Analyzing object modules and executable programs.

  This module also provides:

  - A utility for monitoring system activity.
  - Ancillary control processes for magnetic tapes and Files-11 Level 1 disk structures.
  - A facility for collecting and reporting system hardware and software errors.

▪ MicroVMS Optional Software Products

Because MicroVMS is fully compatible with the VMS operating system, a wide variety of the same proven VMS software products are available for use with MicroVMS. The same system software products, languages and tools, and information management products that have made VMS the most popular general purpose operating system on the market give users of MicroVMS all the flexibility and productivity they have come to expect from a VAX system.

DECnet-VAX

These products provide Digital Network Architecture (DNA) capabilities to MicroVAX systems. DECnet-VAX for MicroVMS systems allows for Digital-to-Digital communications. DECnet-VAX permits users to transfer files among systems, to execute images on others systems, to use the mail utility to correspond with people on other systems, and to otherwise transfer information and affect processes among systems.

With appropriate optional layered software, DECnet-VAX supports Internet software for multivendor communications, standard Packetnet System Interface (PSI), and Ethernet local area networks. Because MicroVMS supports Phase IV DECnet, a MicroVAX system can be part of a network containing thousands of nodes. Both end-node and route-through support are available.

- END-NODE SUPPORT
  This module allows users to originate and receive DECnet messages.

- ROUTE-THROUGH SUPPORT
  This module allows users to originate, receive, and pass DECnet messages through to other systems.

▪ Media Availability

MicroVMS is available for all MicroVAX systems on RX50 5-1/4 inch floppy disks.

For MicroVAX II systems only, MicroVMS will be available on TK50 and RX50 tape.