SOFTWARE PRODUCT DESCRIPTION

PRODUCT NAME: VAX/VMS Operating System, Version 4.2

DESCRIPTION

System Overview

The VAX/VMS Operating System supports all VAX series computers, working reliably and efficiently in both time-sharing and production environments. Time-sharing users can work interactively with the system and can submit batch jobs. Jobs of all types, including processor intensive, I/O intensive, large memory, and real-time, in any mix, perform well on VAX/VMS. The system permits an absolute limit of 8192 concurrent process. However, the actual amount of work supported at one time with good performance depends on the types of processing performed as well as on the physical memory and secondary storage avaliable. The practical limit, with moderate per-user workloads, is in excess of 100 concurrent users for a large scale system.

System Configuration

The user receives with VAX/VMS a standard system configuration file and a site-independent start-up procedure. The user can modify the site-independent start-up procedure or create a site-specific start-up procedure. To start the system, the user performs a bootstrap operation. Optionally, the parameter values of the selected file can be modified at bootstrap time. The system configures itself by executing the start-up procedures. After system start-up, the user can reset system parameter values on-line for the next bootstrap operation and can dynamically change a subset of the parameter values, i.e., change the configuration of the running system.

To aid the user in setting up system parameters, a procedure is provided that analyzes the system configuration and generates a file containing parameter settings. This file can be modified by the user and a system parameter file generated. In addition, the system incorporates self-tuning mechanisms. For example, the system monitors job performance and increases the amount of physical memory for jobs that need it and decreases the amount

for jobs that do not. However, the user is always free to modify system parameters to suit special situations or to fine-tune. For example, the user has the ability to govern the real memory limits for each job. Services are provided for monitoring system performance and for adjusting system parameters.

The user starts the system from a console and a storage device that contains the bootstrap routine. After the system starts, the user can operate the system from any terminal. An automated procedure for shutting down the system in an orderly manner (telling interactive users to log off, spinning down disks, etc.) is provided with the system.

General Access

User access to VAX/VMS is by means of a command language. The DIGITAL Command Language (DCL), the standard command language for VAX/VMS, is supplied with the system. DCL commands take the form of a command name followed by qualifiers. The commands can be typed on the terminal, included in command procedures, and submitted as batch jobs (either from the terminal or the card reader). The commands provide basic system services, initiate system utility programs, and initiate user programs.

Many qualifiers exist to provide detailed levels of control where needed. However, most qualifiers have default values so that the typical command consists only of the command name and a few qualifiers. In addition, command names and keywords can be abbreviated. Another facility that saves keystrokes is the ability to substitute symbolic names for commands. For example, the user might equate a lengthy command to a short symbolic name. Thereafter, only the short name need be typed to execute the command.

On erroneous input, the user receives a message. The user can obtain on-line documentation by typing HELP followed by the name of a command and (optionally) a qualifier. The help facility can be used in an interactive mode and also is available within many of the interactive utility programs.

In addition to the usual services of an operating system in support of program development and operations, VAX/VMS provides the following services that enable nonprogrammers to process data:

* **Text processing** - The general user requires a few days to learn how to use one of the text editors supplied with the system.

* **Mail facility** - A personal mail facility permits a user to send messages to any other user by typing the recipient's name, the subject of the message, and the text of the message. If the recipient is logged in, a notification that new mail has arrived appears on the recipient's terminal. Otherwise, the recipient is notified of the new mail at log-in time. Mail recipients can reply to messages and forward them to other users. Messages can be examined, printed, filed, and edited at the terminal. A directory and search capability is incorporated to permit rapid scanning of mail files. Multinode operation is available if DECnet-VAX is installed.

* **Phone facility** - An interactive communication utility with networking capability (optional DECnet-VAX license is required for multinode operation).

* **Disk I/O** - The general user can create and process sequential character files with a few simple DCL commands. Processing includes editing, expanding, copying, sorting, merging, and deleting files.

* **Command-level programming** - The general user can create special files called command procedures, that contain a series of DCL commands. When a user invokes a command procedure, the system processes all the commands. Command procedures save many keystrokes by permitting the user to catalog series of often- entered commands. In addition, the user can take advantage of special DCL commands to perform programming-type functions such as assigning symbolic names, evaluating numerical and logical expressions, accepting parameters, communicating with the interactive user invoking the command procedure, performing conditional (IF) and branching (GO TO) logic, and handling conditions.

VAX/VMS provides a number of line-mode editors and a line-mode/character-mode editor called EDT. In character mode (also known as keypad mode), EDT permits the user to quickly insert and delete any amount of text within the existing text. Some of the capabilities are as follows:

* **Position** - The user can move a cursor up, down, right, left, to the next word, to the preceding word, or to the end of the line to be in position for an editing operation.

* **Insertions** - To insert text, the user simply types the characters.

* **Deletions** - The user can delete the current character or word, the preceding character or word, or blocks of text.

* **Cut and paste** - The user can move blocks of text from one position to another.

* **Searches** - The user can search for the next or preceding occurence of a particular string of characters.

EDT logs changes and replays them on demand should a failure occur during an editing session.

Finaly, the general user has access to any facilities that the applications programming user makes available through application systems built on top of VAX/VMS. Programming users, for example, might construct an inventory system that permits general users to query and update indexed files that hold the inventory data.

Program Development

VAX/VMS provides a comprehensive set of tools for developing programs including editors (such as EDT for producing and editing source programs), a file differences utility, programming languages, a linker, a librarian, a symbolic debugger, and a patch utility. The assembly-level VAX MACRO language is supplied with all systems.

The file differences utility contrasts two files by automatically aligning matching text and optionally ignoring comments, empty records, trailing blanks, and multiple blanks. Output can consist of a side-by-side display of differences, a file-by-file list of differences, an interleaved list of differences, a list with change bars, or an input file for the batch editor.

The linker binds programs written in the various languages into executable and shareable images. Shareable programs permit the storage of code and data used by many executable images as a single disk file and permit this code and data to be shared in physical memory at image run time. The linker encourages the modular development of application systems by permitting images to be written as small modules and then bound into a large image, and by binding programs written in different languages. (Programs written in all native-mode languages can call one another.) The large per-process address space provided by the system removes any requirement for program overlays.

The VAX Common Run-Time Procedure Library provides geneal string manipulation, I/O and I/O conversion, terminal independent screen handling, date/time, mathematics, G_ and H_ floating point emulation, resource allocation, signaling and condition handling, syntax analysis routines and cross reference procedures that can be called from programes written in VAX MACRO or any optional, native-mode high-level languages.

The common run-time procedure library provides run-time support for the VAX native-mode, high-level languages BASIC, COBOL, FORTRAN, PL/I, RPG, and PASCAL. All routines in the common run-time library follow standard call and condition handling conventions and most are contained within a shareable image.

At a lower level, programs can call the system directly for event flag, asynchronous system trap, logical name, record and file I/O, process control, timer, conversion, condition handling, lock management, and memory management services. Again, the system uses standard call and condtiton handling conventions.

Programmers using native mode, high-level languages need not worry about the mechanics of calling the common run-time procedures or the base system services. The necessary functions are built into the VAX languages. However, the various levels of detail are available for those with special requirements, for example, programmers constructing real-time applications. In addition, the user can add procedures to the common run-time procedure library and can write new system services.

A system utylity, sort/merge, creates sorted files from unsorted binary or character input and merges several sorted files into one, in ascending or descending sequence based on one or more keys.

The sort/merge routines can be invoked as a utility through DCL commands or can be called as subroutines by user programs. In addition to the usual record sort, the sort/merge routines provide tag, address, and index sorts.

A librarian utility permits the programmer to efficiently store object modules, macros, help text, or any general record-oriented information in central, easily accassible files. Object modules are automatically incorporated in images by the linker as they are referenced by calling programs. Macros are automatically included in the source code of VAX MACRO programs as the programs are assembled. Programs can call librarian routines to access help text or general information stored in a library file.

A symbolic debugger permits programmers to trace program execution and to display and modify program contents using the same symbols that are in the source code. Working interactively with DCL-like commands, the programmer can:

* **Set breakpoints** - Specify locations in the program where execution will temporarily halt

* **Set tracepoints** - Specify locations or operation codes which, if used, will cause a message to be displayed during program execution

* **Set watchpoints** - Specify locations, that, if modified, will cause a message to be displayed during program execution

* **Step through the program** - Execute the program on a line-by-line basis.

* **Examine locations** - Display the contents of a location

* **Modify locations** - Change data and instructions at a location

* **Call routines** - Call routines in the program from the interactive command level

* **Log the debugger session** - Place a record of the debugger session in a log file

* **Run from a command procedure** - Invoke a command procedure that contains debugger commands (created from a log file and/or an editing session) to return a previous session, or to check a series if items in the program

Expressions and data formats are generally similar to those of the language in which the program being debugged is written.

A patch utility permits the modification of programs without recompilation and linking. Fixes are applied directly to the executable image files. Locations in programes, however, can be addressed using the same symbols that are in the source code (i.e., the programmer does not have to interpolate from listings and maps). In addition, the utility automatically relocates patches that are larger than the program content being replaced.

The definition, creation, loading, and maintenance of efficient VAX RMS Indexed Sequential (ISAM) files is facilitated by a set of utilities that are integrated by a common File Definition Language (FDL). Analysis of the current state of any RMS file provides information on the file's internal efficiency and validity. The space efficiency of an RMS ISAM file can be improved with a reclamation utility. The most appropriate set of RMS file parameters can be found with an RMS file tuning tool. The utilities that create, efficiently load and reclaim space in RMS files can be called by user programs as well as invoked from the DCL command language. They reside in shareable images.

Operations

The basic unit of execution in VAX/VMS is the process, which consists of context and executable images. The context indentifies the process and describes its current state. The executable images consist of system and/or user programs that have been compiled and linked.

Processes receive processor time for the execution of their images on an event-driven, pre-emptive priority basis. Each time an event such as an I/O interrupt occurs, the system services the event, then passes control to the highest priority process ready to execute (even if the process that had control before the event could continue to execute). Process can be assigned base priorities in the range of 0 to 31, with 4 (as a typical default) the norm time-sharing users and applications that are not time critical. The system automatically adjust priorities whose bases are in the range of 0 to 15 to favor I/O-bound process. Interactive users do not have to wait long periods while computation bound processes tie up the processor. Real-time process can be assigned higher priorities to ensure that they receive processor time whenever they are ready to execute. The system does not adjust priorities whose bases are in the range of 16 to 31, nor does it raise above 15, priorities whose bases are in the range of 0 to 15.

Processes can be started in the following ways:

* **User log-in** - When an interactive use logs in, the system creates a process for the user. The process provides an environment in which the user can communicate with the system to name images to be executed and to perform other activities.

* **Exsplicit creation** - A user can specify the creation of a new process. In creating a process, the user names an image that will be executed when the process starts.

* **Batch job** - A user can name images to be executed and other activities to be performed and can submit this information to the system as a batch job either from a terminal or a card reader.

The system queues batch jobs for execution. The user can regulate the number of queues and the number of streams per queue (that is, the number of batch jobs in the queue that can execute concurrently). Time-sharing users can enhance performance by restricting interactive use of the system to work that requires immediate responses (such as editing), by submitting as batch jobs, work that requires lengthy execution times, and by limiting batch streams to a number that is reasonable for the system configuration.

Different batch job queues may have different attributes such as the maximum CPU time permitted, working set size, and priority. Facilities are provided for started and stopping queues, as well as the jobs in a queue. Jobs in a given queue may be removed or held until a particular time.

Peripheral devices can be controlled by the system or allocated by individual process. At least one disk must be a system disk. Users can designate other disks as system or group disks for the general use of all users logging in to the system or for a subset of users known as a "group". Interactive terminals are controlled by the system, and the system normally controls one or more printers.

Print queues, both generic and specific (with forms recognition) together with queue management facilities give the user versatile print capabilities.

The user can designate system-controlled printers as spooled devices for the output of queued print jobs, so that individual process can specify printed output without actually having to allocate a printer. Process submit printed output as jobs to print queues. Within each queue, print jobs are processed serially by priority; jobs with the same priority are processed on a first-in/first-out basis. A print queue can be associated with one or more printers. The next job in the queue goes on the first free printer associated with the queue. Special forms and printer characteristics can be specified, and print jobs can be started, stopped, restarted, held, and aborted.

Process can allocate devices that are not for general use. For example, an interactive user might allocate a tipe drive on which to back up some disk files. An application system might allocate 12 terminals, two disk drives, and a printer to be used excusively for certain production work.

Each process is permitted an address space as large as four gigabytes (one of which is reserved) with an upper limit of one gigabyte on program size. In practice, processes use more modest address spaces; but even so, many processes, some requiring extensive space, can be running concurrently. VAX/VMS uses the following two mechanisms to provide sufficient physical memory for the many processes using the system:

* **Paging** - The system allots physical memory to a process as needed up to a user-specified limit (typically in the ranges of 50 to 100 kilobytes for interactive users and 200 to 300 kilobytes for batch jobs). If memory in excess of the limit is needed, the system allows the physical memory requirements to expand to a quota and then releases some of the process code and data already in memory (taking care to preserve modifications) to make room for the new code and data. The released code and data are initially cached in memory in case they are needed again, but later they may be removed from physical memory (modifications being written to disk) depending on process directives and total system operations.

* **Swapping** - If too many processes exist to fit in physical memory, the system stores some of the processes on disk. The system monitors process status changes and ensures that the highest priority processes ready to execute are in physical memory. In addition, the system will attempt not to swap an executable process out of memory, until the process has accomplished work for a minimum (configuration-specified) period of time.

Paging and swapping should be high overhead activities as long as concurrent processes do not exceed a reasonable number for physical memory, secondary storage, and the types of processing being performed. The in-memory cache for released code and data, the technique of paging each process against itself, and the assurance that ready-to execute processes are in memory combine to minimize overhead.

If desired, processes can exercise control over memory management. A real-time process, for example, could inhibit the paging and/or swapping of critical code and data.

The system maintains structures in physical memory for sharing code and data among processes, for synchronizing image execution among processes, and for communicating (sending messages) among processes. User processes create and control the structures by requesting appropriate services of the system. Nonshared structures and shared structures can be protected against access by other processes.

The system minimizes operating system overhead by taking advantage of the following VAX hardware features:

* **Context switching and queue instructions** - To schedule processes quickly and efficiently

* **Software interrupt delivery mechanism** - To minimize the processor time required to return from performing a service

* **Asynchronous system trap delivery mechanism and queue instructions** - To minimize the processor time required for I/O processing

* **Interrupt priority levels** - To improve I/O response time

Reliability

The system handles errors as transparently as possible while maintaining data integrity and providing sufficient information to diagnose the errors. The system limits the effects of an error by first attempting to recover from the error, then if recovery fails, by reporting the error to the current process for action, and finaly (if the error cannot be clearly bounded), shutting down and restarting the system. VAX/VMS will shut itself down rather than continue operating with a condition that could propagate undetected bad data. The types of errors possible are as follows:

* **Processor errors (machine checks)** - The system retries the instruction on which the processor failed, as long as no internal state has been modified. If no retry is possible or the retry fails, the system reports the error to the current process as an exception. However, if the executive is currently executing, the system shuts down and performs a cold restart by bootstrapping a fresh copy of VAX/VMS from the system disk.

* **Operating system errors (system errors or undetected hardware failures)** - The system checks its internal data structures for consistency to provide early detection of a system error. If the error effects only a single process, the system reports the error of the process. If the error effects more than one process, the system shuts down and performs a cold restart by bootstrapping a fresh copy of VAX/VMS from the system disk.

* **User errors (user bugs)** - The system uses hardware and software protection mechanisms to prevent processes from damaging the system. As with a system error, the system detects a user error through internal consistency checks and reports the error to the single affected process.

* **Memory errors** - The system examines memory at start-up time and does not use any pages found to be bad. During system operation, the hardware transparently corrects all single-bit memory errors. An unrecoverable error causes the memory page on which the error occurred to be added to the bad page list; if the page has not been modified, system operation continues with a new copy of the page. (Unrecoverable memory errors that occur during a read by an I/O device are reported to the device so that the I/O operation can be reported as failed.)

* **I/O errors** - The system automatically retries failed I/O operations to recover from transient errors and marginal media conditions. Disk I/O errors are retried using the hardware recovery mechanisms. Optionally, the system validates disk I/O transfers with read-check and write-check functions. If an I/O request cannot be completed, all data that was correctly transferred is returned to the initiator.

The system logs all processor errors, all operating system errors detected through internal consistency checks, all double-bit memory errors (and a summary of corrected single-bit errors), and all I/O errors. The log can be printed or examined interactively to locate failed and troublesome components.

If the effects of an unrecoverable error cannot be limited to a process, the system shuts down and restarts without operator intervention by bootstrapping a fresh copy of VAX/VMS from the system disk (although the user can inhibit the automatic restart feature). Whenever the system fails, a dump of physical memory is taken; the dump includes the contents of the processor registers. Additionally, an entry is made in the error log indicating the hardware and software states of the machine at failure. A utility that can access system data structures symbolically is provided for analyzing dumps.

On a power failure, the system shuts down automatically. On power restoration, the system restarts automatically and resumes processing at the point of interruption if the system has a time-of-day clock and a memory battery back-up unit, and if the contents of memory are still valid. The system restarts devices and communications lines. All I/O operations in progress, including magnetic tape, are restarted. On request, programs can be notified of power restoration. An optional battery operated hardware clock resets the date and time of day when the system restarts. If the system does not have a battery back-up unit, or if the memory contents are not valid on power restoration, the system will perform a cold start (reboot).

If for any reason the system disk does not come back on line after power failure within a specific time after the CPU regains power the system shuts down.

Diagnostics can be run on individual devices during normal system operation. The system need not be shut down and run stand-alone in order to format disks, to position disk and tape heads, and to diagnose hardware faults on device drives.

Certain critical components can operate in degraded mode. For example, the memory cache can be disabled. The system places a component in degraded mode when errors pass a treshold.

VAX/VMS includes a User Environment Test Package, which verifies that the major hardware components of the system are complete, properly installed, and ready for use. This package can be executed as part of system installation, or it can be run in stand-alone mode at any time.

One binary version of the operating system is created and distributed for each major release of VAX/VMS. Patches are applied throuhg an automatic, machine-readable, maintenance update procedure. Each patch checks for the proper version number and revision level, and updates these figures as appropriate.

Security and Control

VAX/VMS provides privilege, protection, and quota mechanisms to limit user access to system-controlled structures in physical memory, system-structured files and volumes, and certain devices. Typically, one or a few users (system managers and/or operators) who maintain the system on a day-to-day basis have unlimited access to the system, while the access rights of the remaining users are strictly limited.

User accounts maintained in a user authorization file constitute the basis for privilege and quota assignment. The system manager can modify this file through services provided by the system. Each user account identifies a user by name and password. To login and gain access to the system, the user must supply this name and password, password can never be displayed. (Even users submitting batch jobs from card readers must supply a name and password.) The password is encoded and does not appear on displays; once logged in, the users can change their passwords. In each user's account, the system manager assigns privileges and sets quotas on activities and structures that consume system resources, e.g., physical memory, CPU time, disk space, etc.

Login security includes breakin detection which allows terminals to be disabled when a breakin attempt is detected. The user is also informed at login as to the last time login was done for the account. Secure serving provides a means to prevent user programs from initating the login process to obtain user-names and passwords.

The System Manager also has the ability to limit each user to specific time of day access, such as allowing a user to only have access to the system from 9 AM to 5 PM.

Access control lists exist and allow more flexible protection for files and devices. These lists also allow for the logging of successful and unsuccessful attempts to access files.

Each user receives a user identification code (UIC), on which the protection mechanism is based. The UIC is associated with each structure, file, volume, and device that the user owns. For example, when the user creates a file, the system associates the user's UIC with the file. The system also attaches a user-specified protection mask that permits and/or prohibits categories of access to the protected entity. Typically, a protection mask might permit all users to read a data file, while permitting only the owner to modify or delete it. The protection mechanism also permits users to associate in groups, such that access can be permitted to all members of the group and denied to all others.

Scavenge protection is also provided in three forms. File high-water marking prevents users from reading beyond the end of a file mark. Erase on delete insures that information in a file is zeroed before being returned to general use. Erase on extend prevents a user from reading information that may have been previously allocated to another file.

Security alarms are provided on both a file and a system-wide basis. A class of operations may be defined to receive security related messages.

Input/Output

I/O directives can be specified on the following levels:

* **DCL commands** - Commands such as EDIT, CREATE, APPEND, COPY, PRINT, TYPE, SORT, and DELETE provide the nonprogramming user with the ability to manipulate files.

* **High-level programs** - Programs written in COBOL, FORTRAN, BASIC, PASCAL, C, PL/I and other high-level languages can perform I/O using standard language elements. Application programmers will find this level the simplest and most efficient in most cases.

* **VAX Record Management Services (RMS)** - VAX RMS consists of a set of routines that MACRO and high-level language programs can call for device-independent I/O. VAX MACRO and VAX BLISS-32 programmers will find VAX RMS the most efficient level in most cases.

* **Queue I/O (QIO) system services** - The QIO system services are direct calls to the operating system. Programmers can use this level of I/O to perform special device dependent functions and to eliminate the system overhead involved in RMS; for example, to respond to interrupts from real-time devices as rapidly as possible.

A special program called a device driver executes I/O instructions to actually effect the data transfers. Each different type of I/O device requires its own driver. DIGITAL supplies drivers for all devices supported by VAX/VMS, and users with special needs or nonstandard devices can write their own drivers. (The VAX/VMS documentation set includes the VAX/VMS Guide to Writting a Device Driver.) Additionally, the system provides services for programs to bypass the driver mechanism and handle device interrupts directly for certain UNIBUS devices.

DIGITAL supplies a set of programs called ancillary control processes (ACPs), which maintain standard structures associated with disk, tape, network, and remote terminal I/O. When a new file is created on a structured disk volume, for example, the disk ACP will update the volume's directory.

The I/O routines of the native mode, high-level languages and native-mode utilities are built on top of VAX RMS, which uses the QIO services; thus, all I/O within the system is standard, unless the user elects to bypass the DIGITAL-supplied mechanisms. Supported devices include disks, tapes, unit record equipment, terminals, networks, and mailboxes (virtual devices for interprocess comunication). VAX RMS provides both record access to supported file organizations and an option to bypass record processing; thus, a program can address blocks within a file directly.

VAX RMS supports sequential, relative, stream, and multiple-key indexed disk files. Relative and indexed files can be shared for reading, writing, updating, and deleting records. The system automatically locks and releases records as they are processed. In addition, the programmer can lock and release records explicitly. VAX RMS will control the interlocking for shared, sequential files if the files are formatted as 512-byte, fixed-lenght records. For shared files, the user can optionally request that RMS allocate I/O buffers in a shared global section. Depending on the application, this could reduce the total amount of physical memory used for I/O buffers and/or reduce the amount of disk I/O necessary for processing a file. For other record formats, the user must assume responsibility for an interlocking mechanism. (Compatibility mode programs can also use RMS, but can not use record locking or file sharing.)

Disk and Tape Volumes

Disk volumes can be organized into volume sets. Files of any organization type can span any number of volumes within a volume set. They can be allocated to the set as a whole (the default) or to specific volumes within the set. Volume sets can contain a mix of disk device types and can be extended by adding volumes. Optionally, specified portions of indexed files can be allocated to specified areas of a single disk volume or to specified volumes in a volume set.

Quotas can be placed on the amount of space individual users can allocate. Quota assignment is by UIC and may be controlled individually for each volume set in the system (or volume if the volume is not part of a set).

Structure information can be cached in memory to reduce the I/O overhead required for file management services. Although not required to do so, users can preallocate space and control automatic allocation. For example, when a file is extended, it can be extended by a given number of blocks, contiguously or noncontiguously for optional file system performance in specific cases.

The system applies software validity checks and check-sums to critical disk structure information. The critical information is duplicated. If a volume is improperly dismounted because of user error or system failure, its structure information is automatically rebuilt the next time it is mounted. The system detects bad blocks dynamically and prevents their reuse once the files to which the blocks were allocated are deleted.

The system provides eight levels of named directories and subdirectories, whose contents are alphabetically ordered. Device and file specifications follow standard conventions. Logical names can be used to abbreviate the specifications and to make application programs device and file-name independent. A logical name can be assigned to an entire specification, to a portion of a specification, or to another logical name.

VAX/VMS supports multivolume magnetic tape files with transparent volume switching. Access positioning is either by filename or by relative file position.

System utilities that aid in file maintenance include:

* **File transfer** - Transfers files from one volume to another, files can be in any of several formats.

* **Backup/restore** - Provides comprehensive, online and standalone full volume, and incremental file backup for file structured mounted volumes and volume sets. Files can be backed up to magnetic tape or another disk. Individual files, selected directory structures, or all files on a

volume set can be backed up and restored using standard file naming conventions with selection by date. Backup/restore takes place onto previously initialized and mounted volumes or on unitialized media. The N+1 redundant recording technique permits recovery of backed up data even if one of N blocks has been corrupted. In the case of incremental backups, detection that a file has not been modified since the last backup eliminates the unnecessary movement of data. It is also possible to completely restore a volume or volume set from a series of (for example) daily incremental backups. Backup and restoration of disk files can be selectively performed on line or a per-volume basis off line. Per-volume operations require exclusive access to the volume.

* **Bad block locator** - Locates and records bad blocks on a disk.

* **Analyze disk structure** - Validates structure information on a disk volume against the actual contents, prints structure information, and permits changes to structure information.

Installation

VAX/VMS systems are distributed as binary kits on tape or disk. Procedures for setting up the system disk from a kit and for readying the system for day-to-day operations are simple and straightforward. The binary kit includes the following facilities:

* System installation package
* System configuration procedures
* User Environment Test Package
* Operating system kernel, including virtual memory manager, swapper, system services, and drivers for VAX/VMS supported I/O devices
* System generation utility (for tayloring system parameter files)
* User authorization control program
* Interactive and batch job controller and symbiont manager
* Card reader input symbiont
* Line printer output symbiont
* Bootstrap utility
* Start-up procedure
* Shut-down procedure
* Accounting manager
* Accounting report utility
* Operator communication process
* Message utility

* MAIL utility (requires the optional DECnet-VAX on each node for remote mail to another VAX/VMS system)
* Error logging and print utility
* Help facility
* DCL command interpreter
* DCL command definition utility
* Monitor utility (for displaying system activity)
* Various interactive and batch editors
* Text formatter (DIGITAL standard runoff)
* Log-in facility
* Network command terminal facility (allows remote log-in to another VAX-11 system if the optional DECnet-VAX software license and kit is installed on each node)
* Phone facility - An interactive communications utility with networking capability (requires the optional DECnet-VAX software license and kit on each node for remote communication)
* VAX MACRO assembler with cross-reference capability
* Linker with cross-reference capability
* Install utility
* Librarian utility with cross-reference capability
* Common run-time procedure library
* Symbolic debugger (for MACRO and most optional native-mode languages)
* VAX Record Management Services
* RMS file analysis and validation utility
* RMS file conversion and loading utility
* RMS Indexed Sequential file space reclamation utility
* RMS file design and tuning utility
* Files-11 ancillary control process
* Disk quota utility
* Back-up and restore facilities
* Disk structure analysis
* Magnetic tape ancillary control process
* Sort/merge utility
* File management facilities
* File differences utility
* File search utility (for searching the content of files)
* Dump utility
* Patch utility
* System dump analyzer
* Bad block locator utility
* Software maintenance update procedure

VAX/VMS Distribution

The software installation guides contain a complete list of the modules included in the binary kit, as well as instructions for their installation.

A source kit is available for users who wish to retrieve and modify selected source modules. (Source modules are useful as templaters for user-written device drivers, although this same information can be obtained from the microfiche listing supplied with the binary kit.) Retrieval of selected source modules can occur once the source kit is copied to disk. The source kit can be used for a complete rebuilding of the system, except as noted below. The kit includes the tools (including source patches) that DIGITAL used to build the binary kit; however, any patches released after building the binary kit are not reflected in the source kit.

Although every attempt is made to include accurate source modules, corresponding compiler-version modules, and supporting command procedures, DIGITAL does not warrant the ability to build a complete binary kit of the system. DIGITAL will neither warrant nor supply updates and support services for the compiler-version binary modules. No supporting documentation is provided, and source modules for intermediate updates of VAX/VMS are not available. Source kits are available only on major releases, (i.e., Versions 1.0, 2.0, 3.0, etc.) and are not updated with any maintenance updates. Depending on how much of the source kit is needed, up to three RP06 disk drives or equivalent disk space can be required for processing the source modules after copying them from tape.

In addition, DIGITAL does not warrant the results of using the source kit to change selected portions of the system.

Source listings on microfiche for the system software components are provided on an "as is" basis without DIGITAL warranty expressed or implied.

VAX/VMS Disk Block Requirements

* Block space requirements (Block Cluster size = 1)

* Disk block size for VAX/VMS, Version 4.2 for installation is approximately 16,000 blocks.

  Disk block size for VAX/VMS, Version 4.2 after installation is approximately 34,000 blocks.

  This figure includes a 4600 block page-file. Most systems will require a larger page-file, and a swap-file.

This figure also idcludes library files which were in data-reduced format. Most installations will choose to data-expand these files. This would require approximately 3000 additional blocks.

This block size will be approximately the same for tailored, normal, and syscommon configurations. For tailored systems, the blocks are distributed on two disks. Approximately 6525 blocks will be placed on the system disk, while the remaining blocks will be placed on the library disk.

These sizes are approximations; actual sizes may vary depending on the user's system environment, configuration and options selected.

GROVTH CONSIDERATIONS

The minimum hardware requirements for any future version or update of VAX/VMS may be greater from the minimum hardware requirements for the current version.

SOFTWARE PRODUCT DESCRIPTION

PRODUCT NAME: VAX/VMS Operating System, Version 4.2

DESCRIPTION

System Overview

The VAX/VMS Operating System supports all VAX series computers, working reliably and efficiently in both time-sharing and production environments. Time-sharing users can work interactively with the system and can submit batch jobs. Jobs of all types, including processor intensive, I/O intensive, large memory, and real-time, in any mix, perform well on VAX/VMS. The system permits an absolute limit of 8192 concurrent process. However, the actual amount of work supported at one time with good performance depends on the types of processing performed as well as on the physical memory and secondary storage avaliable. The practical limit, with moderate per-user workloads, is in excess of 100 concurrent users for a large scale system.

System Configuration

The user receives with VAX/VMS a standard system configuration file and a site-independent start-up procedure. The user can modify the site-independent start-up procedure or create a site-specific start-up procedure. To start the system, the user performs a bootstrap operation. Optionally, the parameter values of the selected file can be modified at bootstrap time. The system configures itself by executing the start-up procedures. After system start-up, the user can reset system parameter values on-line for the next bootstrap operation and can dynamically change a subset of the parameter values, i.e., change the configuration of the running system.

To aid the user in setting up system parameters, a procedure is provided that analyzes the system configuration and generates a file containing parameter settings. This file can be modified by the user and a system parameter file generated. In addition, the system incorporates self-tuning mechanisms. For example, the system monitors job performance and increases the amount of physical memory for jobs that need it and decreases the amount

for jobs that do not. However, the user is always free to modify system parameters to suit special situations or to fine-tune. For example, the user has the ability to govern the real memory limits for each job. Services are provided for monitoring system performance and for adjusting system parameters.

The user starts the system from a console and a storage device that contains the bootstrap routine. After the system starts, the user can operate the system from any terminal. An automated procedure for shutting down the system in an orderly manner (telling interactive users to log off, spinning down disks, etc.) is provided with the system.

General Access

User access to VAX/VMS is by means of a command language. The DIGITAL Command Language (DCL), the standard command language for VAX/VMS, is supplied with the system. DCL commands take the form of a command name followed by qualifiers. The commands can be typed on the terminal, included in command procedures, and submitted as batch jobs (either from the terminal or the card reader). The commands provide basic system services, initiate system utility programs, and initiate user programs.

Many qualifiers exist to provide detailed levels of control where needed. However, most qualifiers have default values so that the typical command consists only of the command name and a few qualifiers. In addition, command names and keywords can be abbreviated. Another facility that saves keystrokes is the ability to substitute symbolic names for commands. For example, the user might equate a lengthy command to a short symbolic name. Thereafter, only the short name need be typed to execute the command.

On erroneous input, the user receives a message. The user can obtain on-line documentation by typing HELP followed by the name of a command and (optionally) a qualifier. The help facility can be used in an interactive mode and also is available within many of the interactive utility programs.

In addition to the usual services of an operating system in support of program development and operations, VAX/VMS provides the following services that enable nonprogrammers to process data:

* **Text processing** - The general user requires a few days to learn how to use one of the text editors supplied with the system.

* **Mail facility** - A personal mail facility permits a user to send messages to any other user by typing the recipient's name, the subject of the message, and the text of the message. If the recipient is logged in, a notification that new mail has arrived appears on the recipient's terminal. Otherwise, the recipient is notified of the new mail at log-in time. Mail recipients can reply to messages and forward them to other users. Messages can be examined, printed, filed, and edited at the terminal. A directory and search capability is incorporated to permit rapid scanning of mail files. Multinode operation is available if DECnet-VAX is installed.

* **Phone facility** - An interactive communication utility with networking capability (optional DECnet-VAX license is required for multinode operation).

* **Disk I/O** - The general user can create and process sequential character files with a few simple DCL commands. Processing includes editing, expanding, copying, sorting, merging, and deleting files.

* **Command-level programming** - The general user can create special files called command procedures, that contain a series of DCL commands. When a user invokes a command procedure, the system processes all the commands. Command procedures save many keystrokes by permitting the user to catalog series of often- entered commands. In addition, the user can take advantage of special DCL commands to perform programming-type functions such as assigning symbolic names, evaluating numerical and logical expressions, accepting parameters, communicating with the interactive user invoking the command procedure, performing conditional (IF) and branching (GO TO) logic, and handling conditions.

VAX/VMS provides a number of line-mode editors and a line-mode/character-mode editor called EDT. In character mode (also known as keypad mode), EDT permits the user to quickly insert and delete any amount of text within the existing text. Some of the capabilities are as follows:

* **Position** - The user can move a cursor up, down, right, left, to the next word, to the preceding word, or to the end of the line to be in position for an editing operation.

* **Insertions** - To insert text, the user simply types the characters.

* **Deletions** - The user can delete the current character or word, the preceding character or word, or blocks of text.

* **Cut and paste** - The user can move blocks of text from one position to another.

* **Searches** - The user can search for the next or preceding occurence of a particular string of characters.

EDT logs changes and replays them on demand should a failure occur during an editing session.

Finaly, the general user has access to any facilities that the applications programming user makes available through application systems built on top of VAX/VMS. Programming users, for example, might construct an inventory system that permits general users to query and update indexed files that hold the inventory data.

Program Development

VAX/VMS provides a comprehensive set of tools for developing programs including editors (such as EDT for producing and editing source programs), a file differences utility, programming languages, a linker, a librarian, a symbolic debugger, and a patch utility. The assembly-level VAX MACRO language is supplied with all systems.

The file differences utility contrasts two files by automatically aligning matching text and optionally ignoring comments, empty records, trailing blanks, and multiple blanks. Output can consist of a side-by-side display of differences, a file-by-file list of differences, an interleaved list of differences, a list with change bars, or an input file for the batch editor.

The linker binds programs written in the various languages into executable and shareable images. Shareable programs permit the storage of code and data used by many executable images as a single disk file and permit this code and data to be shared in physical memory at image run time. The linker encourages the modular development of application systems by permitting images to be written as small modules and then bound into a large image, and by binding programs written in different languages. (Programs written in all native-mode languages can call one another.) The large per-process address space provided by the system removes any requirement for program overlays.

The VAX Common Run-Time Procedure Library provides geneal string manipulation, I/O and I/O conversion, terminal independent screen handling, date/time, mathematics, G_ and H_ floating point emulation, resource allocation, signaling and condition handling, syntax analysis routines and cross reference procedures that can be called from programes written in VAX MACRO or any optional, native-mode high-level languages.

The common run-time procedure library provides run-time support for the VAX native-mode, high-level languages BASIC, COBOL, FORTRAN, PL/I, RPG, and PASCAL. All routines in the common run-time library follow standard call and condition handling conventions and most are contained within a shareable image.

At a lower level, programs can call the system directly for event flag, asynchronous system trap, logical name, record and file I/O, process control, timer, conversion, condition handling, lock management, and memory management services. Again, the system uses standard call and condtiton handling conventions.

Programmers using native mode, high-level languages need not worry about the mechanics of calling the common run-time procedures or the base system services. The necessary functions are built into the VAX languages. However, the various levels of detail are available for those with special requirements, for example, programmers constructing real-time applications. In addition, the user can add procedures to the common run-time procedure library and can write new system services.

A system utylity, sort/merge, creates sorted files from unsorted binary or character input and merges several sorted files into one, in ascending or descending sequence based on one or more keys.

The sort/merge routines can be invoked as a utility through DCL commands or can be called as subroutines by user programs. In addition to the usual record sort, the sort/merge routines provide tag, address, and index sorts.

A librarian utility permits the programmer to efficiently store object modules, macros, help text, or any general record-oriented information in central, easily accassible files. Object modules are automatically incorporated in images by the linker as they are referenced by calling programs. Macros are automatically included in the source code of VAX MACRO programs as the programs are assembled. Programs can call librarian routines to access help text or general information stored in a library file.

A symbolic debugger permits programmers to trace program execution and to display and modify program contents using the same symbols that are in the source code. Working interactively with DCL-like commands, the programmer can:

* **Set breakpoints** - Specify locations in the program where execution will temporarily halt

* **Set tracepoints** - Specify locations or operation codes which, if used, will cause a message to be displayed during program execution

* **Set watchpoints** - Specify locations, that, if modified, will cause a message to be displayed during program execution

* **Step through the program** - Execute the program on a line-by-line basis

* **Examine locations** - Display the contents of a location

* **Modify locations** - Change data and instructions at a location

* **Call routines** - Call routines in the program from the interactive command level

* **Log the debugger session** - Place a record of the debugger session in a log file

* **Run from a command procedure** - Invoke a command procedure that contains debugger commands (created from a log file and/or an editing session) to return a previous session, or to check a series if items in the program

Expressions and data formats are generally similar to those of the language in which the program being debugged is written.

A patch utility permits the modification of programs without recompilation and linking. Fixes are applied directly to the executable image files. Locations in programes, however, can be addressed using the same symbols that are in the source code (i.e., the programmer does not have to interpolate from listings and maps). In addition, the utility automatically relocates patches that are larger than the program content being replaced.

The definition, creation, loading, and maintenance of efficient VAX RMS Indexed Sequential (ISAM) files is facilitated by a set of utilities that are integrated by a common File Definition Language (FDL). Analysis of the current state of any RMS file provides information on the file's internal efficiency and validity. The space efficiency of an RMS ISAM file can be improved with a reclamation utility. The most appropriate set of RMS file parameters can be found with an RMS file tuning tool. The utilities that create, efficiently load and reclaim space in RMS files can be called by user programs as well as invoked from the DCL command language. They reside in shareable images.

Operations

The basic unit of execution in VAX/VMS is the process, which consists of context and executable images. The context indentifies the process and describes its current state. The executable images consist of system and/or user programs that have been compiled and linked.

Processes receive processor time for the execution of their images on an event-driven, pre-emptive priority basis. Each time an event such as an I/O interrupt occurs, the system services the event, then passes control to the highest priority process ready to execute (even if the process that had control before the event could continue to execute). Process can be assigned base priorities in the range of 0 to 31, with 4 (as a typical default) the norm time-sharing users and applications that are not time critical. The system automatically adjust priorities whose bases are in the range of 0 to 15 to favor I/O-bound process. Interactive users do not have to wait long periods while computation bound processes tie up the processor. Real-time process can be assigned higher priorities to ensure that they receive processor time whenever they are ready to execute. The system does not adjust priorities whose bases are in the range of 16 to 31, nor does it raise above 15, priorities whose bases are in the range of 0 to 15.

Processes can be started in the following ways:

* **User log-in** - When an interactive use logs in, the system creates a process for the user. The process provides an environment in which the user can communicate with the system to name images to be executed and to perform other activities.

* **Exsplicit creation** - A user can specify the creation of a new process. In creating a process, the user names an image that will be executed when the process starts.

* **Batch job** - A user can name images to be executed and other activities to be performed and can submit this information to the system as a batch job either from a terminal or a card reader.

The system queues batch jobs for execution. The user can regulate the number of queues and the number of streams per queue (that is, the number of batch jobs in the queue that can execute concurrently). Time-sharing users can enhance performance by restricting interactive use of the system to work that requires immediate responses (such as editing), by submitting as batch jobs, work that requires lengthy execution times, and by limiting batch streams to a number that is reasonable for the system configuration.

Different batch job queues may have different attributes such as the maximum CPU time permitted, working set size, and priority. Facilities are provided for started and stopping queues, as well as the jobs in a queue. Jobs in a given queue may be removed or held until a particular time.

Peripheral devices can be controlled by the system or allocated by individual process. At least one disk must be a system disk. Users can designate other disks as system or group disks for the general use of all users logging in to the system or for a subset of users known as a "group". Interactive terminals are controlled by the system, and the system normally controls one or more printers.

Print queues, both generic and specific (with forms recognition) together with queue management facilities give the user versatile print capabilities.

The user can designate system-controlled printers as spooled devices for the output of queued print jobs, so that individual process can specify printed output without actually having to allocate a printer. Process submit printed output as jobs to print queues. Within each queue, print jobs are processed serially by priority; jobs with the same priority are processed on a first-in/first-out basis. A print queue can be associated with one or more printers. The next job in the queue goes on the first free printer associated with the queue. Special forms and printer characteristics can be specified, and print jobs can be started, stopped, restarted, held, and aborted.

Process can allocate devices that are not for general use. For example, an interactive user might allocate a tipe drive on which to back up some disk files. An application system might allocate 12 terminals, two disk drives, and a printer to be used excusively for certain production work.

Each process is permitted an address space as large as four gigabytes (one of which is reserved) with an upper limit of one gigabyte on program size. In practice, processes use more modest address spaces; but even so, many processes, some requiring extensive space, can be running concurrently. VAX/VMS uses the following two mechanisms to provide sufficient physical memory for the many processes using the system:

* **Paging** - The system allots physical memory to a process as needed up to a user-specified limit (typically in the ranges of 50 to 100 kilobytes for interactive users and 200 to 300 kilobytes for batch jobs). If memory in excess of the limit is needed, the system allows the physical memory requirements to expand to a quota and then releases some of the process code and data already in memory (taking care to preserve modifications) to make room for the new code and data. The released code and data are initially cached in memory in case they are needed again, but later they may be removed from physical memory (modifications being written to disk) depending on process directives and total system operations.

* **Swapping** - If too many processes exist to fit in physical memory, the system stores some of the processes on disk. The system monitors process status changes and ensures that the highest priority processes ready to execute are in physical memory. In addition, the system will attempt not to swap an executable process out of memory, until the process has accomplished work for a minimum (configuration-specified) period of time.

Paging and swapping should be high overhead activities as long as concurrent processes do not exceed a reasonable number for physical memory, secondary storage, and the types of processing being performed. The in-memory cache for released code and data, the technique of paging each process against itself, and the assurance that ready-to execute processes are in memory combine to minimize overhead.

If desired, processes can exercise control over memory management. A real-time process, for example, could inhibit the paging and/or swapping of critical code and data.

The system maintains structures in physical memory for sharing code and data among processes, for synchronizing image execution among processes, and for communicating (sending messages) among processes. User processes create and control the structures by requesting appropriate services of the system. Nonshared structures and shared structures can be protected against access by other processes.

The system minimizes operating system overhead by taking advantage of the following VAX hardware features:

* **Context switching and queue instructions** - To schedule processes quickly and efficiently

* **Software interrupt delivery mechanism** - To minimize the processor time required to return from performing a service

* **Asynchronous system trap delivery mechanism and queue instructions** - To minimize the processor time required for I/O processing

* **Interrupt priority levels** - To improve I/O response time

Reliability

The system handles errors as transparently as possible while maintaining data integrity and providing sufficient information to diagnose the errors. The system limits the effects of an error by first attempting to recover from the error, then if recovery fails, by reporting the error to the current process for action, and finaly (if the error cannot be clearly bounded), shutting down and restarting the system. VAX/VMS will shut itself down rather than continue operating with a condition that could propagate undetected bad data. The types of errors possible are as follows:

* **Processor errors (machine checks)** - The system retries the instruction on which the processor failed, as long as no internal state has been modified. If no retry is possible or the retry fails, the system reports the error to the current process as an exception. However, if the executive is currently executing, the system shuts down and performs a cold restart by bootstrapping a fresh copy of VAX/VMS from the system disk.

* **Operating system errors (system errors or undetected hardware failures)** - The system checks its internal data structures for consistency to provide early detection of a system error. If the error effects only a single process, the system reports the error of the process. If the error effects more than one process, the system shuts down and performs a cold restart by bootstrapping a fresh copy of VAX/VMS from the system disk.

* **User errors (user bugs)** - The system uses hardware and software protection mechanisms to prevent processes from damaging the system. As with a system error, the system detects a user error through internal consistency checks and reports the error to the single affected process.

* **Memory errors** - The system examines memory at start-up time and does not use any pages found to be bad. During system operation, the hardware transparently corrects all single-bit memory errors. An unrecoverable error causes the memory page on which the error occurred to be added to the bad page list; if the page has not been modified, system operation continues with a new copy of the page. (Unrecoverable memory errors that occur during a read by an I/O device are reported to the device so that the I/O operation can be reported as failed.)

* **I/O errors** - The system automatically retries failed I/O operations to recover from transient errors and marginal media conditions. Disk I/O errors are retried using the hardware recovery mechanisms. Optionally, the system validates disk I/O transfers with read-check and write-check functions. If an I/O request cannot be completed, all data that was correctly transferred is returned to the initiator.

The system logs all processor errors, all operating system errors detected through internal consistency checks, all double-bit memory errors (and a summary of corrected single-bit errors), and all I/O errors. The log can be printed or examined interactively to locate failed and troublesome components.

If the effects of an unrecoverable error cannot be limited to a process, the system shuts down and restarts without operator intervention by bootstrapping a fresh copy of VAX/VMS from the system disk (although the user can inhibit the automatic restart feature). Whenever the system fails, a dump of physical memory is taken; the dump includes the contents of the processor registers. Additionally, an entry is made in the error log indicating the hardware and software states of the machine at failure. A utility that can access system data structures symbolically is provided for analyzing dumps.

On a power failure, the system shuts down automatically. On power restoration, the system restarts automatically and resumes processing at the point of interruption if the system has a time-of-day clock and a memory battery back-up unit, and if the contents of memory are still valid. The system restarts devices and communications lines. All I/O operations in progress, including magnetic tape, are restarted. On request, programs can be notified of power restoration. An optional battery operated hardware clock resets the date and time of day when the system restarts. If the system does not have a battery back-up unit, or if the memory contents are not valid on power restoration, the system will perform a cold start (reboot).

If for any reason the system disk does not come back on line after power failure within a specific time after the CPU regains power the system shuts down.

Diagnostics can be run on individual devices during normal system operation. The system need not be shut down and run stand-alone in order to format disks, to position disk and tape heads, and to diagnose hardware faults on device drives.

Certain critical components can operate in degraded mode. For example, the memory cache can be disabled. The system places a component in degraded mode when errors pass a treshold.

VAX/VMS includes a User Environment Test Package, which verifies that the major hardware components of the system are complete, properly installed, and ready for use. This package can be executed as part of system installation, or it can be run in stand-alone mode at any time.

One binary version of the operating system is created and distributed for each major release of VAX/VMS. Patches are applied throuhg an automatic, machine-readable, maintenance update procedure. Each patch checks for the proper version number and revision level, and updates these figures as appropriate.

Security and Control

VAX/VMS provides privilege, protection, and quota mechanisms to limit user access to system-controlled structures in physical memory, system-structured files and volumes, and certain devices. Typically, one or a few users (system managers and/or operators) who maintain the system on a day-to-day basis have unlimited access to the system, while the access rights of the remaining users are strictly limited.

User accounts maintained in a user authorization file constitute the basis for privilege and quota assignment. The system manager can modify this file through services provided by the system. Each user account identifies a user by name and password. To login and gain access to the system, the user must supply this name and password, password can never be displayed. (Even users submitting batch jobs from card readers must supply a name and password.) The password is encoded and does not appear on displays; once logged in, the users can change their passwords. In each user's account, the system manager assigns privileges and sets quotas on activities and structures that consume system resources, e.g., physical memory, CPU time, disk space, etc.

Login security includes breakin detection which allows terminals to be disabled when a breakin attempt is detected. The user is also informed at login as to the last time login was done for the account. Secure serving provides a means to prevent user programs from initating the login process to obtain user-names and passwords.

The System Manager also has the ability to limit each user to specific time of day access, such as allowing a user to only have access to the system from 9 AM to 5 PM.

Access control lists exist and allow more flexible protection for files and devices. These lists also allow for the logging of successful and unsuccessful attempts to access files.

Each user receives a user identification code (UIC), on which the protection mechanism is based. The UIC is associated with each structure, file, volume, and device that the user owns. For example, when the user creates a file, the system associates the user's UIC with the file. The system also attaches a user-specified protection mask that permits and/or prohibits categories of access to the protected entity. Typically, a protection mask might permit all users to read a data file, while permitting only the owner to modify or delete it. The protection mechanism also permits users to associate in groups, such that access can be permitted to all members of the group and denied to all others.

Scavenge protection is also provided in three forms. File high-water marking prevents users from reading beyond the end of a file mark. Erase on delete insures that information in a file is zeroed before being returned to general use. Erase on extend prevents a user from reading information that may have been previously allocated to another file.

Security alarms are provided on both a file and a system-wide basis. A class of operations may be defined to receive security related messages.

Input/Output

I/O directives can be specified on the following levels:

* **DCL commands** - Commands such as EDIT, CREATE, APPEND, COPY, PRINT, TYPE, SORT, and DELETE provide the nonprogramming user with the ability to manipulate files.

* **High-level programs** - Programs written in COBOL, FORTRAN, BASIC, PASCAL, C, PL/I and other high-level languages can perform I/O using standard language elements. Application programmers will find this level the simplest and most efficient in most cases.

* **VAX Record Management Services (RMS)** - VAX RMS consists of a set of routines that MACRO and high-level language programs can call for device-independent I/O. VAX MACRO and VAX BLISS-32 programmers will find VAX RMS the most efficient level in most cases.

* **Queue I/O (QIO) system services** - The QIO system services are direct calls to the operating system. Programmers can use this level of I/O to perform special device dependent functions and to eliminate the system overhead involved in RMS; for example, to respond to interrupts from real-time devices as rapidly as possible.

A special program called a device driver executes I/O instructions to actually effect the data transfers. Each different type of I/O device requires its own driver. DIGITAL supplies drivers for all devices supported by VAX/VMS, and users with special needs or nonstandard devices can write their own drivers. (The VAX/VMS documentation set includes the VAX/VMS Guide to Writting a Device Driver.) Additionally, the system provides services for programs to bypass the driver mechanism and handle device interrupts directly for certain UNIBUS devices.

DIGITAL supplies a set of programs called ancillary control processes (ACPs), which maintain standard structures associated with disk, tape, network, and remote terminal I/O. When a new file is created on a structured disk volume, for example, the disk ACP will update the volume's directory.

The I/O routines of the native mode, high-level languages and native-mode utilities are built on top of VAX RMS, which uses the QIO services; thus, all I/O within the system is standard, unless the user elects to bypass the DIGITAL-supplied mechanisms. Supported devices include disks, tapes, unit record equipment, terminals, networks, and mailboxes (virtual devices for interprocess comunication). VAX RMS provides both record access to supported file organizations and an option to bypass record processing; thus, a program can address blocks within a file directly.

VAX RMS supports sequential, relative, stream, and multiple-key indexed disk files. Relative and indexed files can be shared for reading, writing, updating, and deleting records. The system automatically locks and releases records as they are processed. In addition, the programmer can lock and release records explicitly. VAX RMS will control the interlocking for shared, sequential files if the files are formatted as 512-byte, fixed-lenght records. For shared files, the user can optionally request that RMS allocate I/O buffers in a shared global section. Depending on the application, this could reduce the total amount of physical memory used for I/O buffers and/or reduce the amount of disk I/O necessary for processing a file. For other record formats, the user must assume responsibility for an interlocking mechanism. (Compatibility mode programs can also use RMS, but can not use record locking or file sharing.)

Disk and Tape Volumes

Disk volumes can be organized into volume sets. Files of any organization type can span any number of volumes within a volume set. They can be allocated to the set as a whole (the default) or to specific volumes within the set. Volume sets can contain a mix of disk device types and can be extended by adding volumes. Optionally, specified portions of indexed files can be allocated to specified areas of a single disk volume or to specified volumes in a volume set.

Quotas can be placed on the amount of space individual users can allocate. Quota assignment is by UIC and may be controlled individually for each volume set in the system (or volume if the volume is not part of a set).

Structure information can be cached in memory to reduce the I/O overhead required for file management services. Although not required to do so, users can preallocate space and control automatic allocation. For example, when a file is extended, it can be extended by a given number of blocks, contiguously or noncontiguously for optional file system performance in specific cases.

The system applies software validity checks and check-sums to critical disk structure information. The critical information is duplicated. If a volume is improperly dismounted because of user error or system failure, its structure information is automatically rebuilt the next time it is mounted. The system detects bad blocks dynamically and prevents their reuse once the files to which the blocks were allocated are deleted.

The system provides eight levels of named directories and subdirectories, whose contents are alphabetically ordered. Device and file specifications follow standard conventions. Logical names can be used to abbreviate the specifications and to make application programs device and file-name independent. A logical name can be assigned to an entire specification, to a portion of a specification, or to another logical name.

VAX/VMS supports multivolume magnetic tape files with transparent volume switching. Access positioning is either by filename or by relative file position.

System utilities that aid in file maintenance include:

* **File transfer** - Transfers files from one volume to another, files can be in any of several formats.

* **Backup/restore** - Provides comprehensive, online and standalone full volume, and incremental file backup for file structured mounted volumes and volume sets. Files can be backed up to magnetic tape or another disk. Individual files, selected directory structures, or all files on a

volume set can be backed up and restored using standard file naming conventions with selection by date. Backup/restore takes place onto previously initialized and mounted volumes or on unitialized media. The N+1 redundant recording technique permits recovery of backed up data even if one of N blocks has been corrupted. In the case of incremental backups, detection that a file has not been modified since the last backup eliminates the unnecessary movement of data. It is also possible to completely restore a volume or volume set from a series of (for example) daily incremental backups. Backup and restoration of disk files can be selectively performed on line or a per-volume basis off line. Per-volume operations require exclusive access to the volume.

* **Bad block locator** - Locates and records bad blocks on a disk.

* **Analyze disk structure** - Validates structure information on a disk volume against the actual contents, prints structure information, and permits changes to structure information.

Installation

VAX/VMS systems are distributed as binary kits on tape or disk. Procedures for setting up the system disk from a kit and for readying the system for day-to-day operations are simple and straightforward. The binary kit includes the following facilities:

* System installation package
* System configuration procedures
* User Environment Test Package
* Operating system kernel, including virtual memory manager, swapper, system services, and drivers for VAX/VMS supported I/O devices
* System generation utility (for tayloring system parameter files)
* User authorization control program
* Interactive and batch job controller and symbiont manager
* Card reader input symbiont
* Line printer output symbiont
* Bootstrap utility
* Start-up procedure
* Shut-down procedure
* Accounting manager
* Accounting report utility
* Operator communication process
* Message utility

* MAIL utility (requires the optional DECnet-VAX on each node for remote mail to another VAX/VMS system)
* Error logging and print utility
* Help facility
* DCL command interpreter
* DCL command definition utility
* Monitor utility (for displaying system activity)
* Various interactive and batch editors
* Text formatter (DIGITAL standard runoff)
* Log-in facility
* Network command terminal facility (allows remote log-in to another VAX-11 system if the optional DECnet-VAX software license and kit is installed on each node)
* Phone facility - An interactive communications utility with networking capability (requires the optional DECnet-VAX software license and kit on each node for remote communication)
* VAX MACRO assembler with cross-reference capability
* Linker with cross-reference capability
* Install utility
* Librarian utility with cross-reference capability
* Common run-time procedure library
* Symbolic debugger (for MACRO and most optional native-mode languages)
* VAX Record Management Services
* RMS file analysis and validation utility
* RMS file conversion and loading utility
* RMS Indexed Sequential file space reclamation utility
* RMS file design and tuning utility
* Files-11 ancillary control process
* Disk quota utility
* Back-up and restore facilities
* Disk structure analysis
* Magnetic tape ancillary control process
* Sort/merge utility
* File management facilities
* File differences utility
* File search utility (for searching the content of files)
* Dump utility
* Patch utility
* System dump analyzer
* Bad block locator utility
* Software maintenance update procedure

VAX/VMS Distribution

The software installation guides contain a complete list of the modules included in the binary kit, as well as instructions for their installation.

A source kit is available for users who wish to retrieve and modify selected source modules. (Source modules are useful as templaters for user-written device drivers, although this same information can be obtained from the microfiche listing supplied with the binary kit.) Retrieval of selected source modules can occur once the source kit is copied to disk. The source kit can be used for a complete rebuilding of the system, except as noted below. The kit includes the tools (including source patches) that DIGITAL used to build the binary kit; however, any patches released after building the binary kit are not reflected in the source kit.

Although every attempt is made to include accurate source modules, corresponding compiler-version modules, and supporting command procedures, DIGITAL does not warrant the ability to build a complete binary kit of the system. DIGITAL will neither warrant nor supply updates and support services for the compiler-version binary modules. No supporting documentation is provided, and source modules for intermediate updates of VAX/VMS are not available. Source kits are available only on major releases, (i.e., Versions 1.0, 2.0, 3.0, etc.) and are not updated with any maintenance updates. Depending on how much of the source kit is needed, up to three RP06 disk drives or equivalent disk space can be required for processing the source modules after copying them from tape.

In addition, DIGITAL does not warrant the results of using the source kit to change selected portions of the system.

Source listings on microfiche for the system software components are provided on an "as is" basis without DIGITAL warranty expressed or implied.

VAX/VMS Disk Block Requirements

* Block space requirements (Block Cluster size = 1)

* Disk block size for VAX/VMS, Version 4.2 for installation is approximately 16,000 blocks.

  Disk block size for VAX/VMS, Version 4.2 after installation is approximately 34,000 blocks.

  This figure includes a 4600 block page-file. Most systems will require a larger page-file, and a swap-file.

This figure also idcludes library files which were in data-reduced format. Most installations will choose to data-expand these files. This would require approximately 3000 additional blocks.

This block size will be approximately the same for tailored, normal, and syscommon configurations. For tailored systems, the blocks are distributed on two disks. Approximately 6525 blocks will be placed on the system disk, while the remaining blocks will be placed on the library disk.

These sizes are approximations; actual sizes may vary depending on the user's system environment, configuration and options selected.

GROVTH CONSIDERATIONS

The minimum hardware requirements for any future version or update of VAX/VMS may be greater from the minimum hardware requirements for the current version.