digital

Introduction to
VAX–11
Record Management Services

Order No. AA-D024C-TE

VAX11

March 1980

This document contains a general description of the record management services of the VAX/VMS operating system. The information in this document is introductory in nature; hence, programming techniques are not presented in this document. Indexed sequential access support is included.

Introduction to
VAX–11
Record Management Services

Order No. AA-D024C-TE

SUPERSESSION/UPDATE INFORMATION: This revised document supersedes the Introduction to VAX-11 Record Management Services (Order No. AA-D024B-TE)

OPERATING SYSTEM AND VERSION: VAX/VMS V02

SOFTWARE VERSION: VAX/VMS V02

To order additional copies of this document, contact the Software Distribution Center, Digital Equipment Corporation, Maynard, Massachusetts 01754

digital equipment corporation · maynard, massachusetts

First Printing, August 1978
Revised, February 1979
Revised, March 1980

The postage-prepaid READER'S COMMENTS form on the last page of this document requests the user's critical evaluation to assist us in preparing future documentation.

The following are trademarks of Digital Equipment Corporation:

DIGITAL	DECsystem-10	MASSBUS
DEC	DECtape	OMNIBUS
PDP	DIBOL	OS/8
DECUS	EDUSYSTEM	PHA
UNIBUS	FLIP CHIP	RSTS
COMPUTER LABS	FOCAL	RSX
COMTEX	INDAC	TYPESET-8
DDT	LAB-8	TYPESET-11
DECCOMM	DECSYSTEM-20	TMS-11
ASSIST-11	RTS-8	ITPS-10
VAX	VMS	SBI
DECnet	IAS	PDT
DATATRIEVE	TRAX	

Contents

Figures

Tables

Preface

Manual Objectives

The *Introduction to VAX-11 Record Management Services* is an introductory manual; it is not intended as a guide to the actual programming techniques required for creating and processing data files. Rather, its purpose is to illustrate the basic file concepts and their application to VAX-11 Record Management Services (VAX-11 RMS). Actual programming techniques are covered in the *VAX-11 Record Management Services Reference Manual.*

Intended Audience

This manual aims at all levels of user. If you are a novice, you can use the manual as a stepping-stone to a basic understanding of record and file management. Experienced users unfamiliar with Digital Equipment Corporation software can use the manual to gain familiarity with DIGITAL terms and techniques. Experienced DIGITAL users should find the manual useful because it explains, in general terms, the record management services of the VAX/VMS operating system.

Structure of this Manual

Information in this document is organized as follows.

- Chapter 1 outlines the basic concepts of disk and tape files, and describes their interface with VAX-11 RMS.
- Chapter 2 describes the various file organizations, record access modes, and record formats you can use.
- Chapter 3 outlines the general procedures you can use to create and process a file, and explains the different modes of file access that a program can use with VAX-11 RMS.

A glossary of terms appears at the end of this manual.

Associated Documents

VAX/VMS provides record management for all supported languages. To use VAX-11 RMS, VAX-11 MACRO programmers will find the *VAX-11 Record Management Services Reference Manual* and the *VAX-11 Record Management Service User's Guide* of special interest. Users of higher-level languages such as VAX-11 FORTRAN, VAX-11 BASIC, or VAX-11 COBOL 74, should refer to their particular language manual for the necessary information on using VAX-11 RMS. The *RMS-11 MACRO-11 Reference Manual* will be

especially helpful to system programmers using MACRO-11 to employ RMS-11 facilities. Users of RMS utilities (DEFINE, CONVERT, BACKUP, DISPLAY, RESTORE, and IFL) will be aided by referring to the *RMS-11 Users's Guide*. If you are using, or planning to use, RMS-11 on VAX/VMS to build and access files, consult the *RMS-11 User's Guide*, which describes RMS-11 concepts and facilities. The *VAX-11 Information Directory and Index* briefly describes all system documentation, defining the intended audience for each manual and providing a synopsis of each manual's contents.

Summary of Technical Changes

This manual has been revised to reflect VAX/VMS support for file sharing for sequential files with 512-byte fixed-length records.

Chapter 1
Introduction

1.1 File Storage Media in General

As technology has increased, so has the volume of information that must be saved. Business and industrial concerns, for instance, have compiled many types of data files about a wide range of subjects. For a long time, all these data files were stored efficiently on paper in desk drawers and filing cabinets. But, as these paper files grew, it often took longer to locate the needed data than to create it in the first place. Storing all of a company's data files on paper reached the point of diminishing returns.

As the need for saving data increased, the need also arose for a better medium to store and retrieve data quickly, reliably, and economically. The computer and computerized filing systems provide such storage media.

At first, computerized files consisted of collections of punched cards.

Cards provide a convenient means of grouping related pieces of information. This information might represent, for example, a business event, such as a purchase or sale of office furniture. Or, in the engineering environment, the information could represent variable equations and data constants related to stress analysis. This information, grouped on a single card, represents a record of that event. Records of similar events, grouped together, constitute a file.

As a storage medium, cards have certain advantages. They are easy to add, delete, or rearrange. However, cards become worn, require physical handling, and are bulky.

Cards are also relatively slow to process because they allow only sequential file access. Sequential access means that the search for a record starts at the beginning of the file and proceeds in order through each record. At times, when the needed record (or group of records) is near the end of the file, the search wastes computer processing time.

The introduction of magnetic tape as a storage medium eliminated some of the disadvantages of cards.

Magnetic tape provides both a storage medium and a means for input/output (I/O), and its uses have grown along with the needs of the user. Magnetic tape offers virtually unlimited storage. A large amount of data can fit on a single tape reel, and you can use as many reels as necessary. Magnetic tape requires much less storage space than a card file with a comparable amount of data.

But, magnetic tape is limited as a storage medium because, like a card file, it allows only sequential file access.

In contrast to magnetic tape, disk storage allows direct file access. Direct access means that the computer can locate the desired record without first

searching the records that precede it in the file. The time needed to access the record is relatively independent of the record's location in the file.

1.2 What is VAX–11 RMS?

VAX–11 Record Management Services (VAX–11 RMS) is the file and record access subsystem of the VAX/VMS operating system for the VAX–11 processor. In combination with the VAX/VMS operating system, VAX–11 RMS allows efficient and flexible data storage, retrieval, and modification.

When you write a program in one of the supported source languages, you can select, from among VAX–11 RMS's file organization and record-access modes, the processing features best suited to your specific application. File organization is the arrangement of data within a file. The file organization and the storage media together determine which techniques can be used to manipulate data. The record access mode is the manner in which data in the file is stored and retrieved.

When you create a VAX–11 RMS file, you specify, through either an application program or a VAX–11 utility routine, the file storage medium and the file and record characteristics. Chapter 2 of this manual outlines VAX–11 RMS file organizations, record access modes, and record characteristics.

After VAX–11 RMS creates the file according to the specified file and record characteristics, your application program can store, retrieve, and modify data records in the file. Chapter 3 of this manual outlines creating, storing, retrieving, and modifying data records in file.

Magnetic tape and disk are the media for storing VAX–11 RMS files. Sections 1.3 and 1.4 below describe the basic concepts of the disk and tape storage media that VAX–11 RMS uses. VAX–11 RMS also supports unit-record devices, such as line printers, terminals, and card readers in a device-independent fashion. The main purpose of this manual, however, is to provide you with an introduction to record keeping on magnetic media.

1.3 Basic Disk Concepts

VAX–11 RMS disk files reside on Files–11 disks. Files–11 refers to the logical structure given to the disk, that is, a hierarchical organization of files, their data, and the directories needed to gain access to them. The VAX/VMS file system implements the disk structure and provides access control to the files located on the disk. VAX–11 RMS provides the organization of the files and method of access to the data within the files.

This section describes the Files–11 structure and defines the terminology related to it.

The smallest addressable unit of information on a disk is a block. On a Files–11 disk, a block consists of 512 eight-bit bytes. One or more blocks may be treated as a single unit for transfer between a Files–11 disk and memory.

Blocks are logically grouped into a cluster, which is the basic unit of disk space allocation. The system manager or operator determines the number of

blocks in a cluster. This is done when a given disk (a volume) is first prepared for use (initialized). Allowable cluster sizes are in the range of 1 to 65535 blocks. In general, a disk with a relatively small number of blocks would usually be given a small cluster size, while larger disks would be given larger cluster sizes to minimize the overhead for disk space allocation.

Contiguous clusters allocated to a particular file are called an extent. An extent can contain all or part of a file. The user need not be concerned with managing extents. This feature is provided by the system. If enough contiguous area is available on the disk, the entire file is allocated as a single extent. Sometimes, however, not enough contiguous area is available to contain the entire file. Or sometimes when you initially create a file you may not wish to reserve the entire required amount of space. Then, when the file is eventually extended, it is unlikely that the adjacent clusters will still be unallocated, thus the extension will not occur contiguously. In either of these cases, the file is divided into two or more parts, and each part is an extent. So, a file can consist of multiple extents located in separate areas on the disk, as shown in Figure 1–1.

Figure 1–1: File Extents

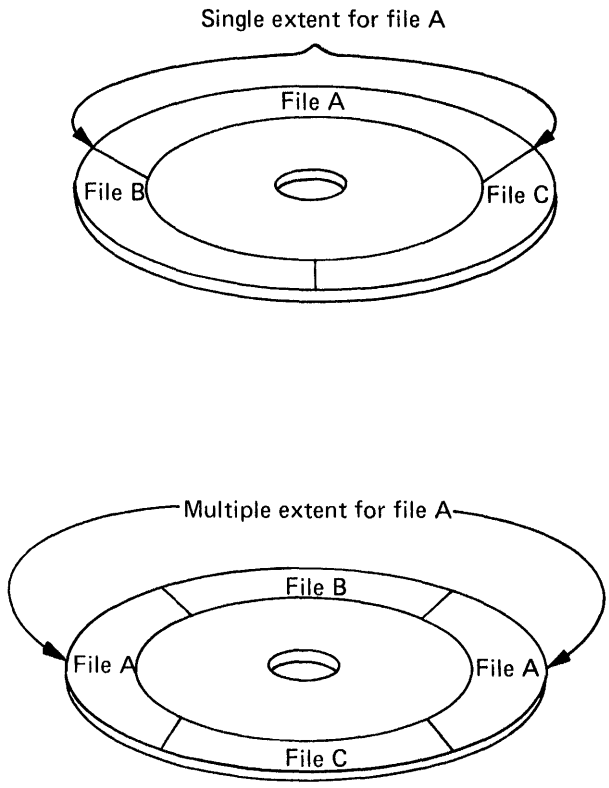
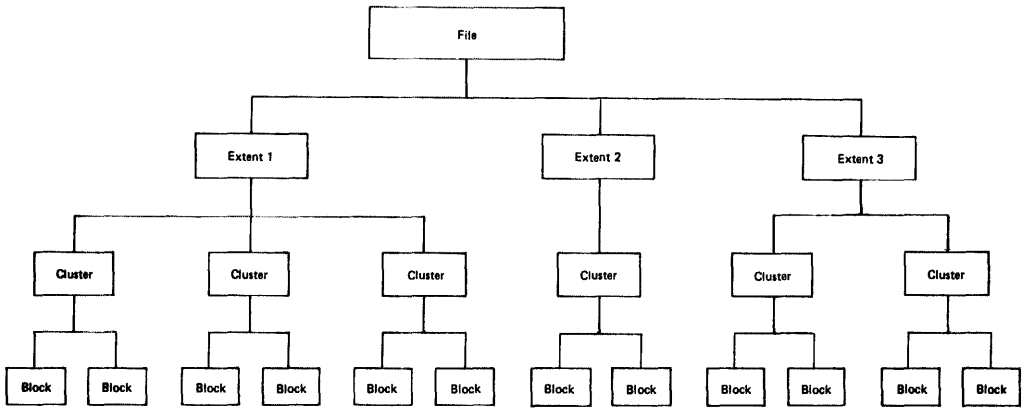


Figure 1–2 shows the hierarchy of blocks, clusters, extents, and files in the Files–11 structure.

Figure 1-2: Files-11 Hierarchy



You can exercise as little or as much control as you desire over the allocation of space to a Files-11 disk file. At one extreme, you can specify the number of blocks to be allocated and even give their exact location on the volume. At the other extreme, you can allow VAX-11 RMS to automatically handle all disk space allocation details. As an intermediate position, you can specify, for example, the size of the initial space allocation and the size to be used by VAX-11 RMS each time the file is extended. In addition, if a file should later require less space, you can specify that the unused clusters are to be deallocated from the file. These clusters are then available for allocation to other files on the volume.

When a large amount of file storage space is needed, you can combine several Files-11 volumes into what is called a volume set. A volume set has the appearance of one large volume. The different extents of a file can be located on different volumes in the volume set. In general, you need not specify a specific volume in the set to locate a file or create a new one, although it is possible and desirable, for performance reasons, to specify a specific volume to be used for a particular allocation request.

The smallest item discernible to the Files-11 structure is the block. VAX-11 RMS uses the blocks in the Files-11 structure and allows you to interface with the individual records in the block.

Other basic terms related to disk structure are track and cylinder.

A track is the collection of blocks at a single radius on one recording surface of a disk. A track is accessible to a given read/write head position on the disk device.

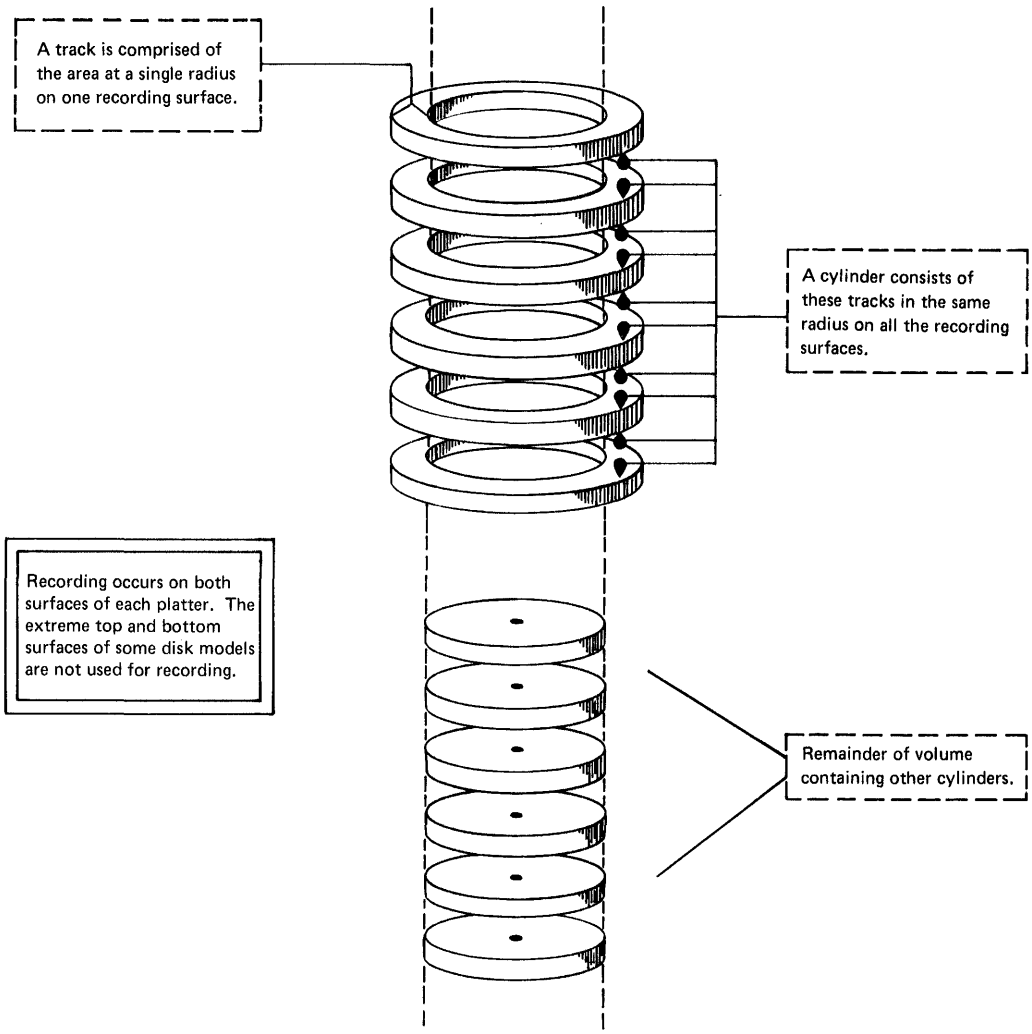
A cylinder consists of all tracks at the same radius on all recording surfaces of a disk.

Because access to any of the blocks in a given cylinder does not require any movement of the disk's read/write heads, it is generally advantageous to keep related data blocks in the same cylinder. For this reason, when choosing a

cluster size for a large-capacity disk, a system manager will often select a cluster size that divides evenly into the cylinder size.

Figure 1–3 illustrates the track and cylinder concept.

Figure 1–3: Track and Cylinder Concept



A record is the set of data that your program treats as a unit, since it contains data (about a specific item or event) that you want to manipulate. A record can be the same size as a block, smaller than a block, or even larger than a block. Therefore, a block can contain one record, multiple records, or only part of a record.

The remainder of this section contains a brief explanation of some basic elements of the Files–11 structure.

A Files–11 structure resides on a volume, which is a physical device medium, such as a disk pack. A Files–11 volume is an ordered set of 512-byte blocks. The blocks are numbered consecutively from 0 to n (the value of n depends on the size of the disk).

Each Files-11 volume has an index file, which is created when the volume is initialized. (You cannot use a disk until it has been initialized with the INITIALIZE command.) The index file contains the following information:

- Bootstrap block
- Home block
- File headers

The bootstrap block is physically the first block on the volume. All Files-11 volumes have an area for this bootstrap block even if the operating system does not require a bootstrap block. If the volume is not the operating system volume, the bootstrap block area contains a different program. This program displays a message on the system console indicating that the volume is not the operating system volume, but is, rather, a volume containing only user files.

The home block is normally the next block on the volume; it identifies the disk as a Files-11 volume. If for some reason the home block cannot be read (physically unusable), an alternative block will be selected for use as the home block. This block provides specific information about the volume and default values for files on the volume. Among the items in the home block are the following:

- The volume name
- Information to locate the remainder of the index file
- The maximum number of files that can be present on the volume at any one time
- The User Identification Code (UIC) of the owner of the volume
- Volume protection information (specifies which users can read and/or write the entire volume)

Files-11 volumes contain several copies of the home block to ensure against accidental destruction of this information and the consequent inability to locate other files on the volume.

The bulk of the index file consists of file headers; each file header describes one file on the volume. File headers contains information such as the file ownership, protection, creation date, and creation time. Most importantly, the file header contains a list of extents that make up the file, describing where the file is physically located on the volume. If a file has a large number of extents, multiple file headers are used to describe them. A file identifier number is associated with each file header.

When you create a file, you normally specify a file name to VAX-11 RMS, which assigns this name to the file on a Files-11 volume. VAX-11 RMS places the file name and file identifier associated with the newly created file in a directory, which contains an entry defining the location for each file. When you access the file, you supply the file name, which associates with the file identifier through the directory entry. The file identifier, in turn, points to the

location of the file header which contains a listing of the extent or extents that locate the actual data.

1.4 Basic Tape Concepts

You can also use tape as a file storage medium with VAX-11 RMS. VAX-11 RMS supports the standard magnetic tape structure defined by American National Standard X3.27-1977, *Magnetic Tape Labels and Record Formats for Information Interchange.*

Data records are organized on magnetic tape in the order in which they are written. In other words, the organization of data on magnetic tape is sequential.

Characters of data on magnetic tape are measured in bits per inch (bpi). This measurement is called density. A 1600 bpi tape can accommodate 1600 characters of data in an inch of recording space.

Even though a tape may have a density of 1600 bpi, there are not always 1600 characters on every inch of magnetic tape because of the interrecord gap (IRG). The IRG is an interval of blank space between data records; it is created automatically when records are written to the tape. The IRG is a breakpoint on the tape, which allows the tape unit to decelerate, and stop if necessary, after a record operation and then accelerate to working speed before the next record operation.

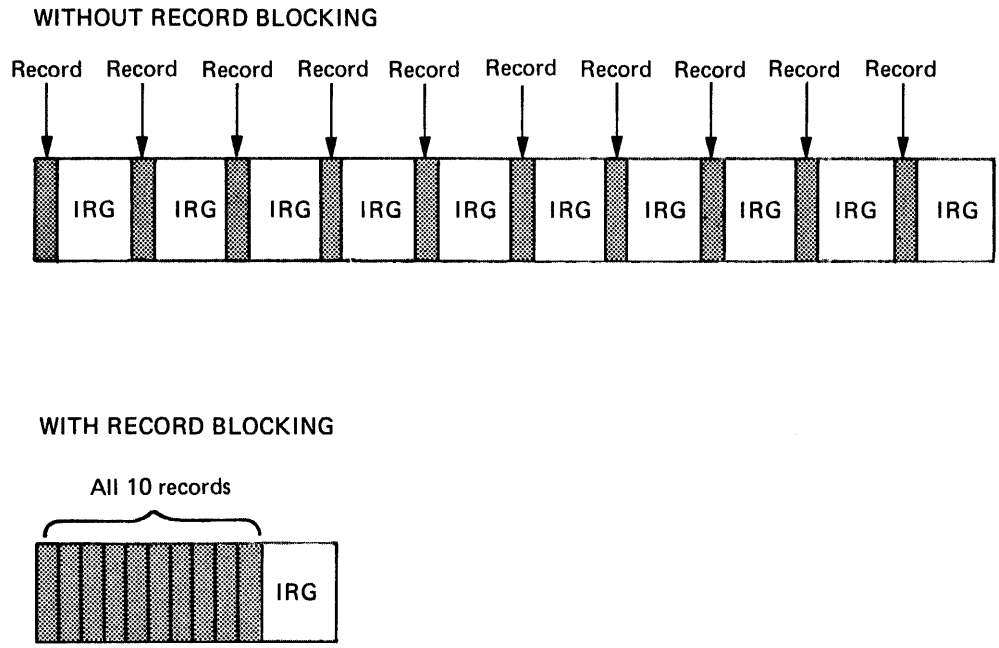
Each IRG is approximately 0.6 inch in length. Writing an 80-character record at 1600 bpi requires 0.05 inch of space (the IRG, therefore, requires twelve times more space than the data). Clearly, this wastes valuable storage space. VAX-11 RMS can reduce the size of this wasted space by using record blocking. This technique groups individual records into a block and places the IRG after the block rather than after each record. (A difference exists between a block on disk and a block on tape: on disk, a block is fixed at a size of 512 bytes; on tape, the size of a block is user determined.) However, record blocking requires more buffer space to be allocated for your program, thus increasing VAX-11 RMS memory requirements. The greater the number of records in a block, the greater the buffer size requirements. You must determine the point, based on the configuration of your computer system, at which the benefits of record blocking cease.

The following example shows how space can be saved by record blocking. A 1600 bpi tape contains 10 records that are not grouped into a block. Each record is 160 characters long (or 0.1 inch at 1600 bpi) with a 0.6 inch IRG after each record. This would use 7 inches of tape. However, placing the same 10 records into one block uses only 1.6 inch of tape (1 inch for the data records and 0.6 inch for the IRG). Figure 1–4 illustrates this example of record blocking.

Record blocking also increases the efficiency of the flow of data into the computer. For example, 10 unblocked records require 10 separate physical transfers, while 10 records placed into a single block require only one physical

transfer. Moreover, a shorter length of tape is traversed for the same amount of data; thus the operation is completed in less time.

Figure 1-4: Interrecord Gaps



Data on magnetic tape is also organized into files. When you create a file on tape, the file system writes a set of header labels on the tape immediately preceding the data blocks. These labels contain information such as the user-supplied file name, creation date, and expiration date. Additional labels, called trailer labels, are also written following the file. To access a file on tape by the file name, the file system searches the tape for the header label set that contains the specified file name.

When the data blocks of a file, or related files, do not physically fit on one volume (a reel of tape), the file is continued on another volume. This is a multivolume tape file. All the volumes contained in the multivolume file constitute a volume set. When access is made to a file on a volume set, all volumes in the set are searched.

For additional information about tape and disk structures, see the *VAX-11 Record Management Services Reference Manual.*

Chapter 2
VAX–11 RMS Files

2.1 Overview

This chapter briefly describes the following:

- VAX–11 RMS file contents
- VAX–11 RMS file organization
- VAX–11 RMS record access modes (retrieving and storing)
- VAX–11 RMS record characteristics (format)
- VAX–11 RMS file size

All functionalities described in this chapter are available at the MACRO level. This is not always the case, however, for higher-level languages. Users of higher-level languages should refer to the appropriate section of their language manual to see which RMS functionalities are available to them.

2.2 File Contents

A VAX–11 RMS file is a collection of logically related records treated as a unit and arranged in a useful order. The contents of each record in the file are defined by your program that creates the file. Any subsequent programs that access this file must use the same definitions. For example, your program to create records in a payroll file must contain the definition of each item of data in the record. In that program, you divide the record into functional groups, known as fields (which can also be divided into subfields). For a payroll application, each record in the file might be divided into fields such as the following:

- Employee name
- Social security number
- Pay rate
- Deductions

The user that creates the file selects the length of each field and the number of fields, and thus determines the length of each record. Each programmer with access to this file determines how each record is used in a given application.

2.3 File Organization

File organization is the physical arrangement of records in the file. File organization is related to, but distinct from, the record access mode, which is the

manner in which the records in the file are stored and retrieved. The purpose of file organization is to arrange the records in an orderly manner so that you can access them in the quickest and most feasible way. You select the type of file organization at file-creation time; it cannot be changed thereafter (although a utility conversion routine may be used to copy this file to another volume, or a different area on the same volume, re-creating it with a different type of file organization). VAX-11 RMS allows you to select from three types of file organizations:

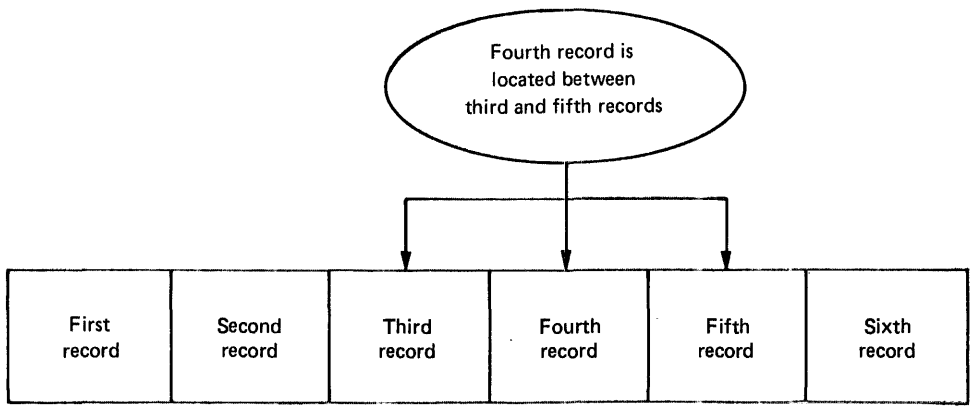
1. Sequential
2. Relative
3. Indexed

The following subsections describe the fundamental structure of each of these organizations.

2.3.1 Sequential Organization

In sequential file organization, records in a file are arranged in one-after-the-other fashion. The sequential file organization is supported for all device types; indeed, for all devices except disk, it is the only organization supported. The sequence in which records appear is the order in which they were written to the file. In other words, if you need to access the fourth record, it can be found between the third and fifth records. In general, this would look like Figure 2-1.

Figure 2-1: General Picture of Sequential File Organization



You can only add records to a sequential file at the current end of the file. This is because the physical location of each record is fixed in relation to all other records in the file. Therefore, there is no unused space where a new record might be inserted.

When you update an existing record in a sequential file (retrieving the record, altering its contents, and reinserting the record in its original position), you cannot change the length of the record.

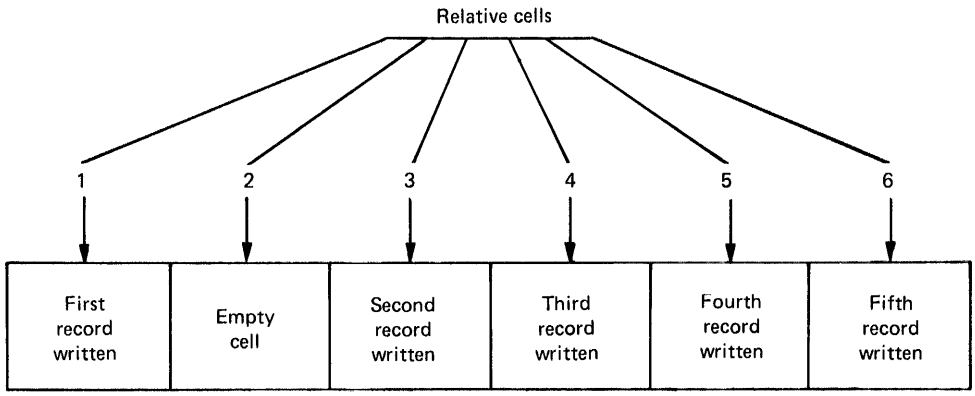
2.3.2 Relative Organization

The second method of file organization, called relative file organization, is available on disk devices only. This type of organization is more complex than sequential organization but also more powerful. A relative file consists of a series of fixed-length positions (or cells) that are consecutively numbered from 1 to n. This number is the relative record number; that is, the number of the record gives its position relative to the beginning of the file. Each record written into the file is assigned to a specific cell within the file. For example, the second record written might be placed in the fourth cell; it would then be relative record number 4. When searching for this record, you must direct VAX-11 RMS to the record by specifying its relative record number. This assignment can result in cells that do not contain records being interspersed in the file.

Relative file organization allows random retrieval of records by means of the relative record number. The relative record number of a record in a relative file is the key value in random access mode; refer to Section 2.4.2 for a detailed description of random access by key. The fixed-length cells let VAX-11 RMS calculate the record's actual position. The unused cells accept new records as they are inserted into the file. A record can also be deleted from any cell in the file, thereby freeing cells for new records.

Figure 2-2 shows, in general, a relative file organization.

Figure 2-2: General Picture of Relative File Organization

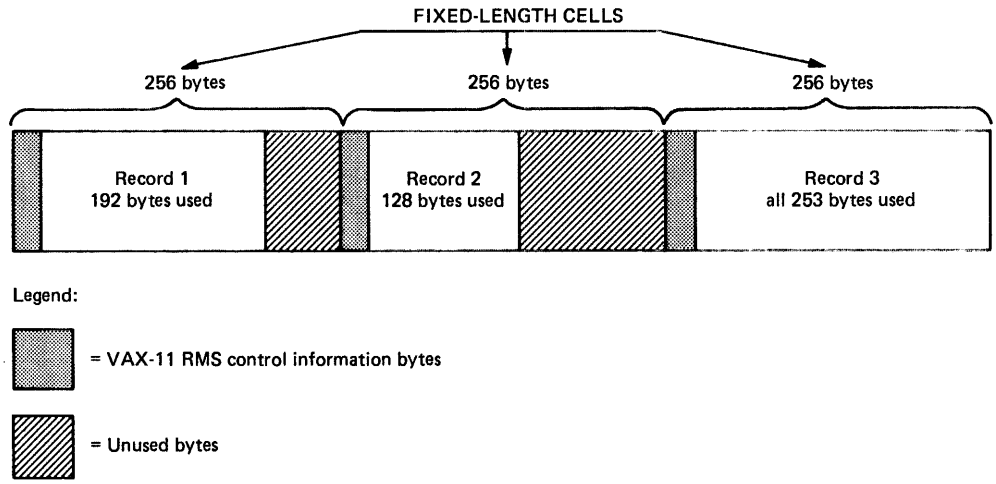


To access a record in a relative file by random mode, you must know the relative record number (key) of the record. One method of keeping track of each record's cell is to store records based on a numeric value within the record (for example, an account number being equivalent to the relative record number).

Although each record in a relative file is assigned to a fixed-length cell, the actual size of the individual records can vary in length (that is, different size records can be in the same file). Clearly, the size of each record must be less than or equal to the size of the fixed-length cell. For example, assume that the fixed length of each cell in a relative file is 256 bytes. Suppose only three

records are in the file, each of a different length; the first record is 192 bytes long, the second 128 bytes, and the third 253 bytes. Each of these records can reside in the same relative file because each occupies its own fixed-length cell, as shown in Figure 2–3. Note that each record is prefixed by three bytes that indicate the record length and whether this cell is empty (a delete flag). These bytes are used only by VAX–11 RMS; you need not be concerned with them, except when planning the file's space requirements.

Figure 2–3: Variable-Length Records in Fixed-Length Cells



2.3.3 Indexed Organization

The third method of file organization, called indexed file organization, is available on disk devices only. This type of organization allows you to store records in predefined sort orders. The keys of the file define the sort orders. You can then retrieve these records sequentially by one of the sorted orders or randomly by one of the record's sort values. Because VAX–11 RMS completely controls the placement of records in an indexed file, the location of records is transparent to your program.

A key (a character string, a packed decimal number, a 2- or 4-byte unsigned binary number, or a 2- or 4-byte signed integer) is defined by its location and length within each record. When identifying the location and length of character string keys, you can define single or segmented keys. A simple key is a single contiguous string of characters in the record. A segmented key, however, can consist of from two to eight such strings; these strings do not need to be contiguous. When processing records that contain segmented keys, VAX–11 RMS treats the separate segments as a logically contiguous string. Integer, binary, and packed decimal keys are always simple keys.

At least one key, or sort order, must be defined for an indexed file. This mandatory key is the primary key. Optionally, additional keys may be defined, which are referred to as alternate keys. Each alternate key is also defined by its location and length within the record. Your program uses the contents of an alternate key in a record to identify that record in the alternate

sort order. As your program inserts records into an indexed file, VAX-11 RMS uses the values of the primary and alternate keys to build indexes. An index is the structure that allows the sorted records to be retrieved randomly.

In addition to defining simple or segmented keys, you can specify key characteristics. Those key characteristics are as follows:

1. Duplicate key (primary and alternate) values are allowed
2. Key (alternate only) values can change
3. Key (alternate only) values can be null values

When you specify that duplicate key values are allowed, you indicate that more than one record in an indexed file can have the same value for that key. Such records, therefore, have the same record identifier. Only the first record with a duplicate key value can be retrieved randomly; subsequent records must be retrieved sequentially. This capability is applicable only to indexed files. In relative files, the record identifier, representing a relative record number, is always unique.

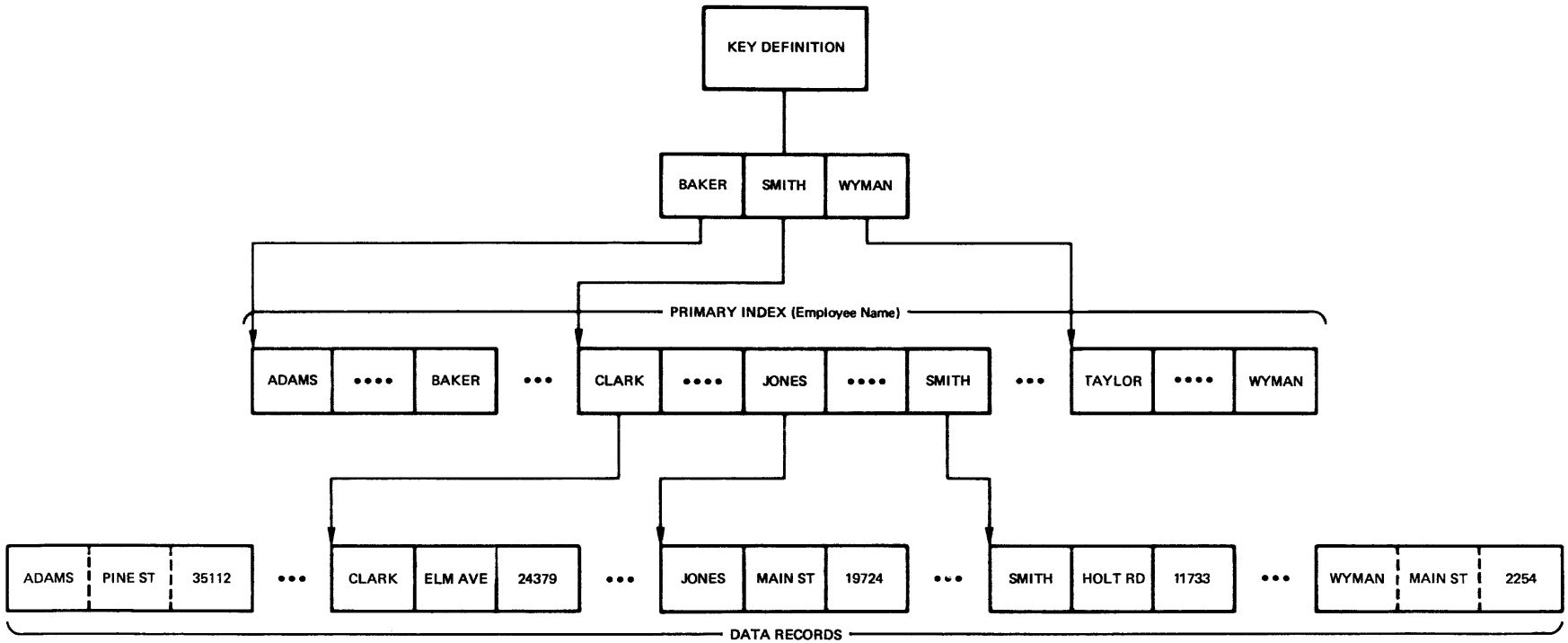
When you specify that key values can change, you indicate that records can be updated with a modified value in the key; however, this characteristic is limited to alternate keys. When such modification occurs, VAX-11 RMS automatically updates the appropriate index to reflect the new key value.

The third key characteristic that you can specify, the null key value, indicates that when the record is put into the file, the key value may be a null value; however, this characteristic is limited to alternate keys. VAX-11 RMS does not make an entry in the index for this record. If the record is subsequently updated with a key value, VAX-11 RMS then makes an entry in the appropriate alternate index.

You can also declare the converse of these characteristics. That is, you can specify that for a given key, duplicate key values are not allowed, key values cannot change, and there is no null value. When duplicate key values are not allowed, VAX-11 RMS rejects any request to put a record into the file when that record contains a key value that is already present in another record. Similarly, when key values cannot change, VAX-11 RMS does not allow your program to update a record by modifying the key value. Records whose keys cannot be null values always have an entry in the appropriate alternate index.

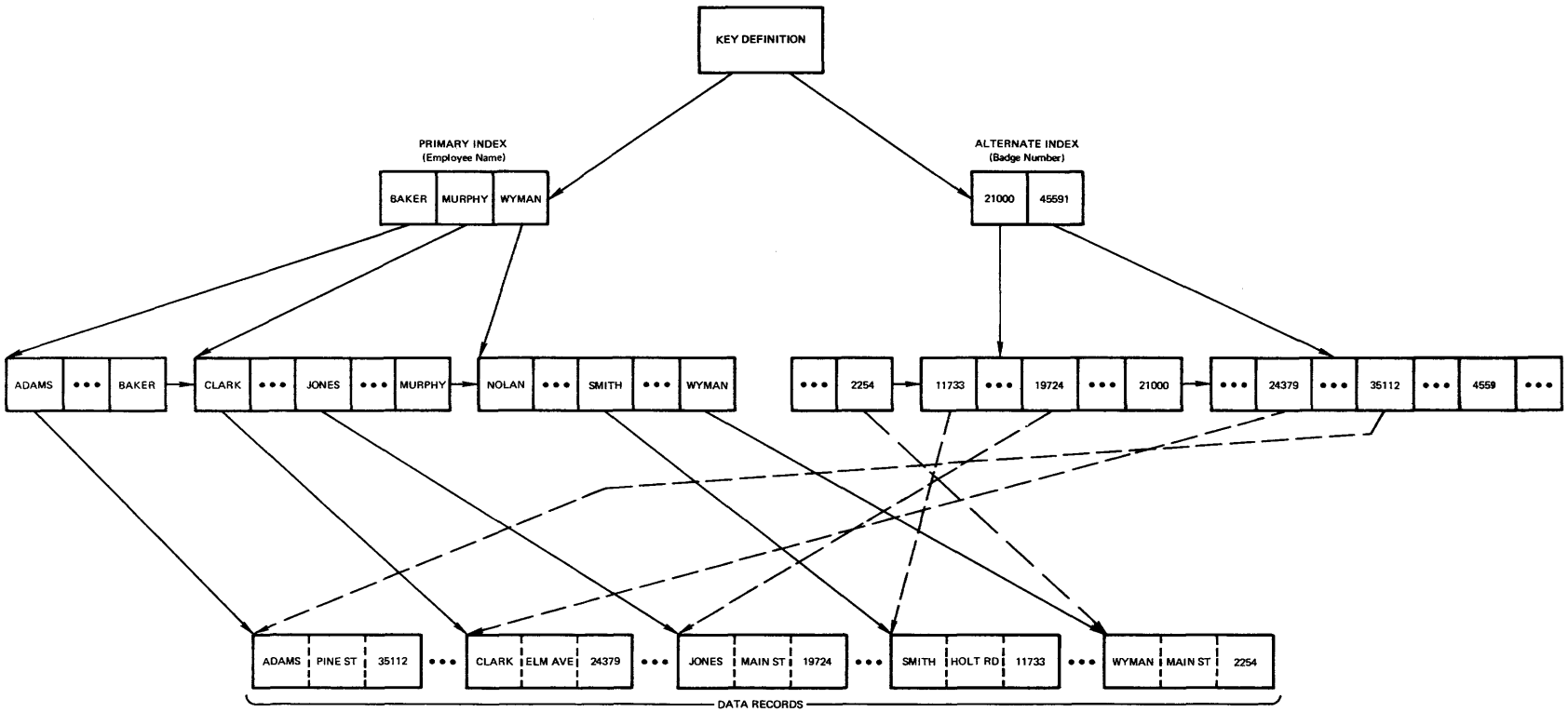
VAX-11 RMS builds and maintains a separate index for each key that is defined for the file. Each index is stored in the file. Thus, an indexed file contains at least one index, for the primary key, and when alternate keys are defined, the file contains another index for each alternate key.

Figure 2-4 illustrates the general structure of an indexed file which has only the primary key defined, the primary key being the names of employees in an employment record file. Figure 2-5 illustrates the general structure of an indexed file which has the primary key and one alternate key defined, the primary key being the names of employees and the alternate key being the badge number of employees in an employment record file.



Note: Assumes one record per bucket.

Figure 2-4: Single-Key Indexed File Organization



Note: Assumes one record per bucket.

Figure 2–5: Multiple-Key Indexed File Organization

Each data record is placed in the file in sorted order by primary key. In alternate indexes, the sort sequence is established by key ordering pointers to the actual record. These mechanisms enable the records to be read sequentially in sorted order by any key.

2.4 Record Access Modes

Record access mode is the manner in which the records in a file are stored and retrieved. This is different from file organization, which is the physical arrangement of records in the file.

VAX-11 RMS provides three types of record access modes:

1. Sequential
2. Random by key
3. Random by record's file address (RFA)

While you cannot change the file organization once it is established (at file-creation time), the record access mode can change each time the file is used. In fact, the record access mode can even change for each record access to a file. This means, for example, that a relative file can be processed in sequential record access mode one time and in random by key (relative record number) record access mode the next time. Table 2-1 lists the combinations of record access modes and file organizations permitted by VAX-11 RMS.

Table 2-1: Record Access Modes and File Organizations

Record Access Mode Permitted	File Organization		
	Sequential	Relative	Indexed
Sequential	Yes	Yes	Yes
Random by Key	No[1]	Yes[3]	Yes
Random by Record's File Address	Yes[2]	Yes	Yes

[1] Random access by relative record number for sequential files is permitted for fixed-length record format on disk devices.

[2] Random access by record's file address is permitted only on disk devices.

[3] The key for relative files is the relative record number.

The following subsections describe the record access modes and the capability for switching record access modes during program execution.

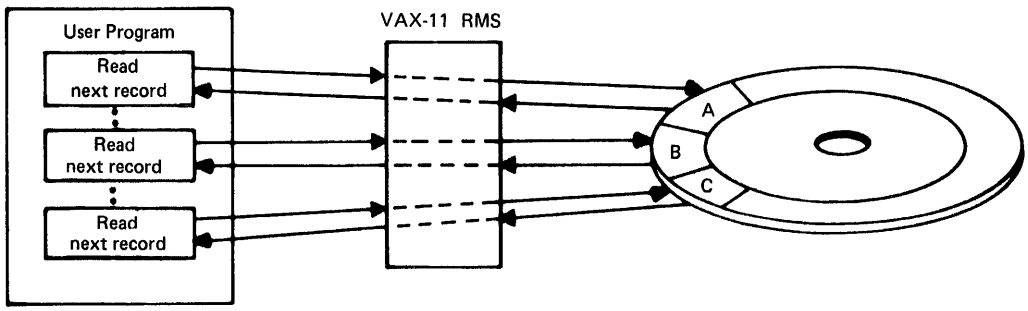
2.4.1 Sequential Access

In sequential access, record storage or retrieval starts at a designated point in the file and continues in one-after-the-other fashion through the file. When using the sequential record access mode, a program issues a series of requests to VAX-11 RMS to retrieve or store the next record in the file. VAX-11 RMS interprets these requests in the context of the organization of the file. As shown in Table 2-1, the sequential access mode is available for all types of file organization: sequential, relative, and indexed. The subsections that follow describe how sequential access deals with each type of file organization.

2.4.1.1 Sequential Access to Sequential Files — In a sequential file, records are virtually adjacent to each other. To read a particular record within the file using sequential access mode, your program must open the file (initiate file processing) and successively read the records preceding this record. Each record can be retrieved only by first retrieving all the records that physically precede it.

Figure 2-6 shows a surface of a disk volume. Each lettered section on the surface represents a record in a sequential file, beginning with record A. When your program issues a series of requests to access the records sequentially, VAX-11 RMS interprets these instructions in the context of the file organization. Since this particular file is sequential, VAX-11 RMS knows that every record after the first record follows the current record; thus, it implicitly knows the position for the next record access.

Figure 2-6: Sequential Access to a Sequential File

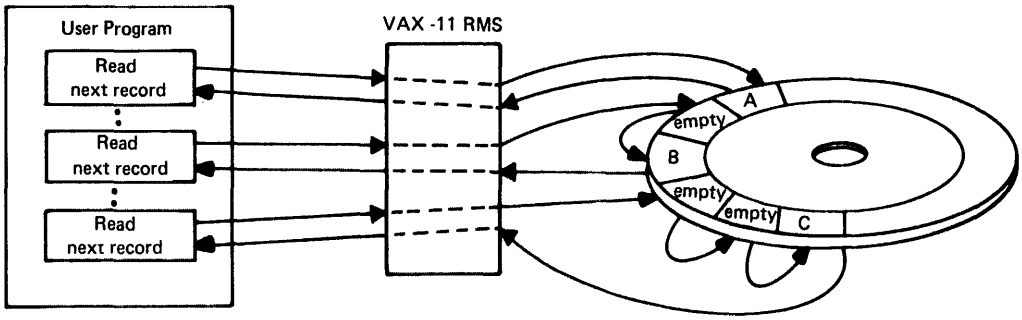


Sequential access does not allow you to backspace to a record that was already passed. You have to restart at the first record by either reopening the file or rewinding the file, or by using dynamic access (see Section 2.4.4). In addition, you can add records only to the end of the file.

2.4.1.2 Sequential Access to Relative Files — Relative files also permit sequential access (see Table 2-1). This holds true even if not all the fixed-length cells in the file contain records. Empty cells, representing records that were

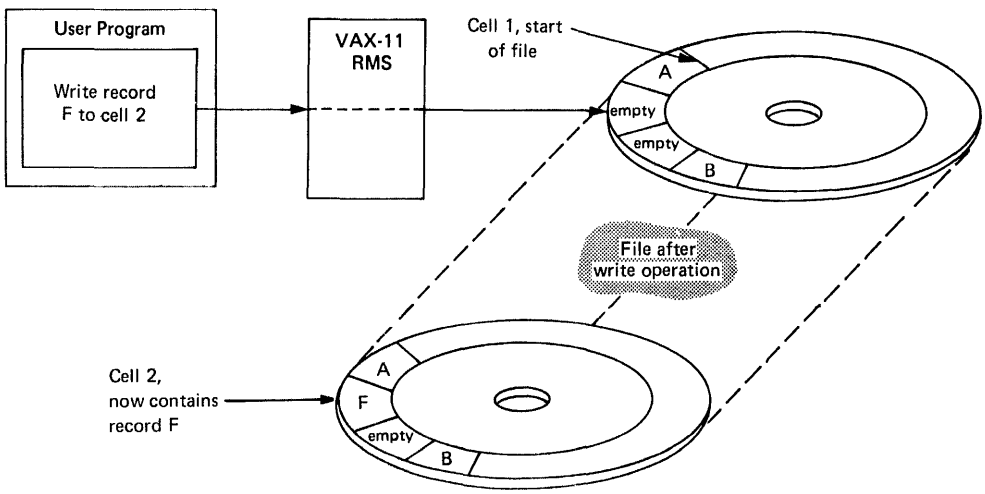
never written (or records that were deleted), can occur in a relative file. VAX-11 RMS recognizes whether successive record cells in a relative file are empty or contain records; VAX-11 RMS ignores the empty cells and searches, in succession, for cells that contain records. For example, a relative file might contain records only in cells 1, 3, and 6. Successive sequential read requests cause VAX-11 RMS to read the record in cell 1, then cell 3, and then cell 6. Figure 2-7 shows how VAX-11 RMS checks each cell, ignores an empty cell when it finds one, and then checks the next cell for a record.

Figure 2-7: Sequential Read of a Relative File



When using sequential access to write records to a relative file, VAX-11 RMS places the new record in the empty cell whose relative number is one higher than that used for the previous write (or read) request. New records cannot be written into cells that already contain records. As Figure 2-8 shows, your program directs VAX-11 RMS to write record F into cell 2. Record A already occupies cell 1, but cell 2 is empty, so VAX-11 RMS can write the record into this cell. If this request were followed by a request to write the next record, VAX-11 RMS would write into cell 3 which is also empty. If you attempt to write into a cell already occupied by a record — for example, by issuing yet another request to write the next record — the attempt will fail. You can, however, request VAX-11 RMS to update (change) an existing record.

Figure 2-8: Write to a Relative File



2.4.1.3 Sequential Access to Indexed Files — In an indexed file, VAX-11 RMS uses one or more indexes to determine the order in which to process records in sequential access mode. The entries in an index are arranged in ascending order by key values. Thus, an index represents a logical ordering of the records in the file. If you define more than one key for the file, each index associated with a key will represent a different logical ordering of the records in the file. Your program, then, can use the sequential access mode to retrieve records in the order represented by any index.

When reading records from an indexed file in sequential mode, your program must initially specify to VAX-11 RMS a key of reference (for example, primary key, first alternate key, second alternate key, and so on). Thereafter, VAX-11 RMS uses the index associated with that key of reference to retrieve records in the same sequence as the entries in the index. Each successive record that VAX-11 RMS returns in response to your program's read request contains a value in the specified key field that is equal to (when duplicate keys are allowed) or greater than that of the previous record returned.

In contrast to a sequential read request, a sequential write request to an indexed file does not require a key of reference. Rather, VAX-11 RMS uses the stored definition of the primary key to place the record in the file and the definition of alternate keys to place entries for the record in the alternate indexes. When your program issues a series of sequential write requests, VAX-11 RMS verifies that each successive record contains a key value in the primary key field that is equal to or greater than that of the preceding record.

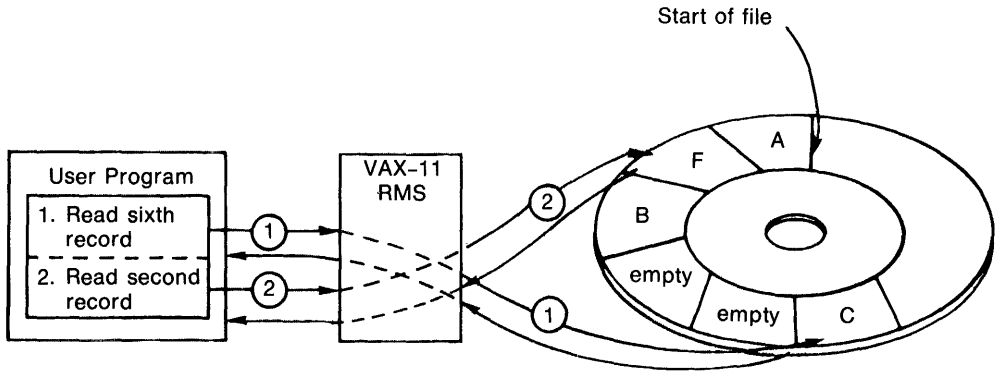
2.4.2 Random Access by Key

In random access by key, your program, not the file organization, determines the order in which record processing occurs. Each program request for a record must include the key value (relative record number for relative files and key of reference plus the key value for indexed files) of the particular record to be accessed.

Random access is always supported for relative and indexed files. VAX-11 RMS also permits random access by relative record number for sequential disk files, but only if the records in the file are of fixed length (see Sections 2.4.3 and 2.5.1.1).

Unlike sequential access, which always follows the one-record-after-the-other access order, random access follows no specific, predefined pattern. Your program randomly identifies, by means of the key value (relative record number for relative files and key of reference plus the key value for indexed files), any record within the file and VAX-11 RMS retrieves (or writes) that record. Your program may make successive requests for accessing or retrieving records that are anywhere within the file. For example, as shown in Figure 2-9, your program directs VAX-11 RMS to read the sixth record of a relative file. The circled numbers indicate the order of access requests; in this case the sixth record is C. The program then directs VAX-11 RMS to read the second record (record F).

Figure 2–9: Random Access by Relative Record Number (Key)



Compare this figure with those for sequential access (Figures 2–6 and 2–7).

Each of your program's read requests in random access mode to an indexed file must specify a key value and the index (for example, primary index, first alternate key index, second alternate key index, and so on) that VAX-11 RMS must search. When VAX-11 RMS finds, by means of the index, the record that matches the key value, it reads the record and passes it to your program. Random access can be accomplished on any key by any of the following methods:

- Exact match of key values.
- Approximate match of key values (that is, record key value greater than your program-supplied key value, or record key value greater than or equal to your program-supplied key value).
- Generic match of key values. Generic match is applicable to string data type keys only. A generic match is defined as a match on some number of leading characters in the key. You determine the number by specifying a search key smaller than the entire field.
- Combination of approximate and generic match.

In contrast to read requests, which require your program-specified key value, your program requests to write records randomly in an indexed file do not require the separate specification of a key value. All keys (primary and, if any, alternate key values) are in the record itself. When an indexed file is opened, VAX-11 RMS retrieves all key definitions stored in the file. Thus, VAX-11 RMS knows the location and length of each key field in a record. After writing a record into the file, VAX-11 RMS examines the key values in the record, and creates new entries in the appropriate alternate indexes.

In this way, VAX-11 RMS ensures that the record can be retrieved by any of its key values.

2.4.3 Random Access by Record's File Address

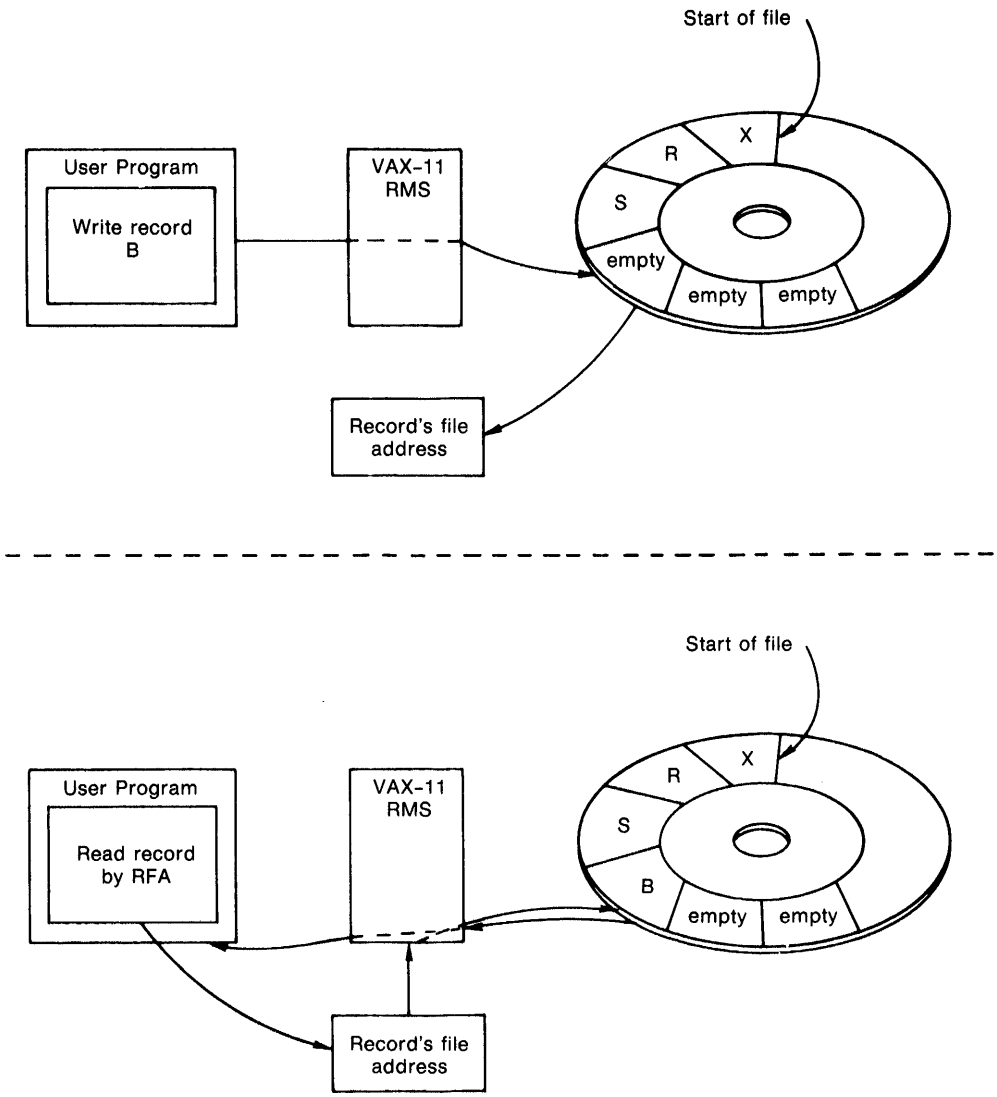
Every record in a file has a unique address within the file — the record's file address (RFA). This is another mode of retrieving records in all types of file

organizations, as long as the file resides on a disk. RFA mode provides the only means of randomly accessing a sequential file with variable-length records.

An important feature of RFA is that the address remains constant as long as the record exists in the file. VAX-11 RMS returns the RFA to you in a parameter block when the record is read or written. You can either ignore the RFA or retain it to use as a pointer to the record whenever you need to access the record.

Figure 2–10 contains two illustrations. The first shows that when you write a record in a file, its RFA is returned to you. The second illustration shows that when this record is needed for a subsequent operation, and you supply the RFA, VAX-11 RMS simply uses the RFA to find the record directly.

Figure 2–10: Random Access by Record's File Address



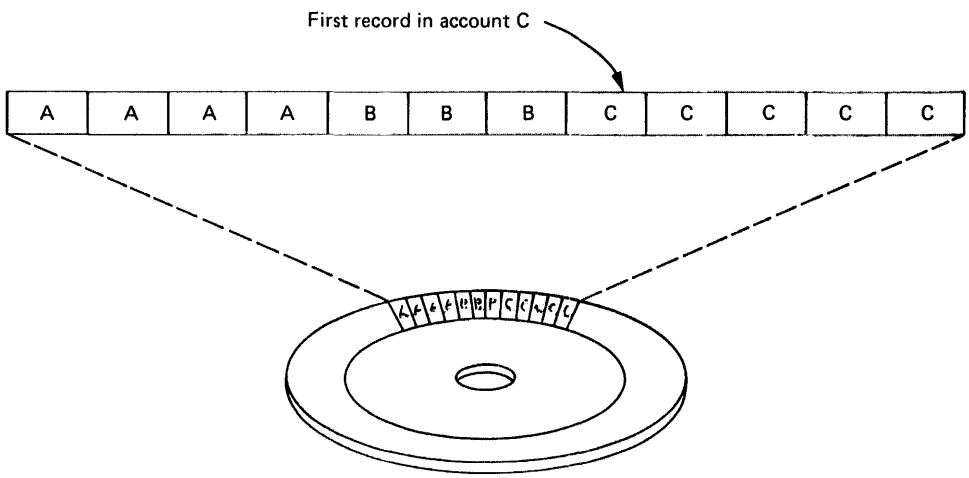
2.4.4 Record Access Mode Switching

VAX-11 RMS lets you switch from one type of record access mode to another while processing a file. This is known as dynamic access. Dynamic access is not a record access mode in itself; rather, it is a term that describes the capability of changing record access modes in mid-stream. No limitation exists on the number of times such switching can occur. The type of file organization must support the types of record access modes selected (see Section 2.4.2 for the restrictions on random access by relative record number with sequential files).

Dynamic access can be most effective immediately following a random-access operation. When a program randomly accesses a record, VAX-11 RMS establishes a current record position in the file for the program. If the program switches dynamically to sequential access, VAX-11 RMS interprets this switch to sequential access as a request for the next record (or records, if desired). Then, using the randomly accessed record as the starting point of the file, VAX-11 RMS can obtain subsequent successive records in the file using the sequential access mode.

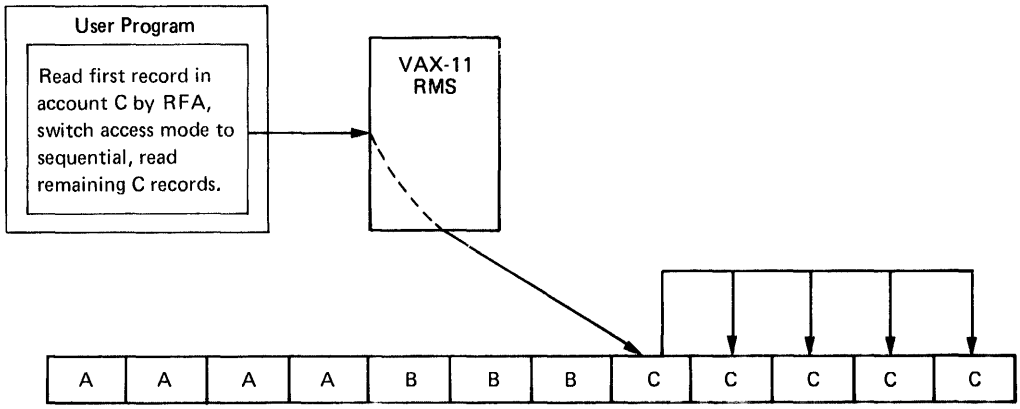
For example, in the sequential file shown in Figure 2-11, the records were written in account-number order, and each account number has more than one record (for every transaction, the user created one record).

Figure 2-11: Sequential Account File for Dynamic Access



Suppose the need arises for a list of all transactions in account C. As described above, the normal manner of processing a sequential file is to sequentially access each record in the file. However, if the user has saved the RFA returned in the parameter block when the first record in account C was created, then random access by RFA can be used to establish a starting point at the first record in account C, rather than at the beginning of the file. Then, with dynamic access, the user can switch to sequential access to retrieve the remaining records, as shown in Figure 2-12.

Figure 2-12: Using Dynamic Access to Switch Record Access Modes



The program logic must allow the program to identify the point in the file at which sequential access is no longer needed. The program can then direct VAX-11 RMS as to what step to follow, such as switching back to random access to select other records. It is also possible to use sequential access, switch to random access, and return to sequential access, if desired. This is very useful when you want to return to a record that was already passed in the sequential record access mode.

2.5 Record Characteristics

VAX-11 RMS provides the mechanism for programs to process files. Certain characteristics of these files are apparent to the user. These characteristics are:

- The record formats
- The storage structure (buckets and areas)

The following subsections describe record formats and storage structure.

2.5.1 Record Formats

VAX-11 RMS is not concerned with the actual contents of records, except for the values contained in key fields of records within indexed files. It deals instead with the record format, which is the way a record physically appears on the recording surface of the storage medium.

VAX-11 RMS provides a choice of three different record formats:

1. Fixed length
2. Variable length
3. Variable with fixed-length control (not supported for indexed files)

These record formats are supported for sequential, relative, and indexed file organizations, except as noted. In relative files, all records are stored in fixed-

length cells, even though the records themselves might be of variable-length or variable with fixed-length control format.

2.5.1.1 Fixed-Length Records — The term fixed-length record format refers to a file whose records are all of the same size. In a fixed-length record file, the number of bytes of data per record is set to a given number at file-creation time. Each record, then, occupies an identical amount of space in the file; however, not all of a record's data bytes need contain information. Unused bytes can exist in each fixed-length record. If so, you are responsible for recognizing any unused fields.

You specify the size of the fixed-length records when you create the file. This size specification becomes part of the information that VAX-11 RMS stores and maintains for the file; it cannot be changed.

2.5.1.2 Variable-Length Records — The term variable-length record format refers to a file whose records need not be of the same size. Because the number of bytes of data per record is not fixed, each variable-length record occupies only as much space as it needs to contain the data of that record.

To allow retrieval of variable-length records from a file, VAX-11 RMS prefixes a count field to each record when it writes the record to the file. VAX-11 RMS constructs this count field from information that you provide in your program, and considers the count field separate from the record data.

VAX-11 RMS uses two types of variable-length formats: V format and D format. The format used depends on the device on which the file resides:

1. Disk files — V format

   For variable-length records in files residing on disk volumes, VAX-11 RMS writes a 2-byte (1-word) binary count field before the data field of each record. This number does not include the count field itself.

2. Tape files — D format

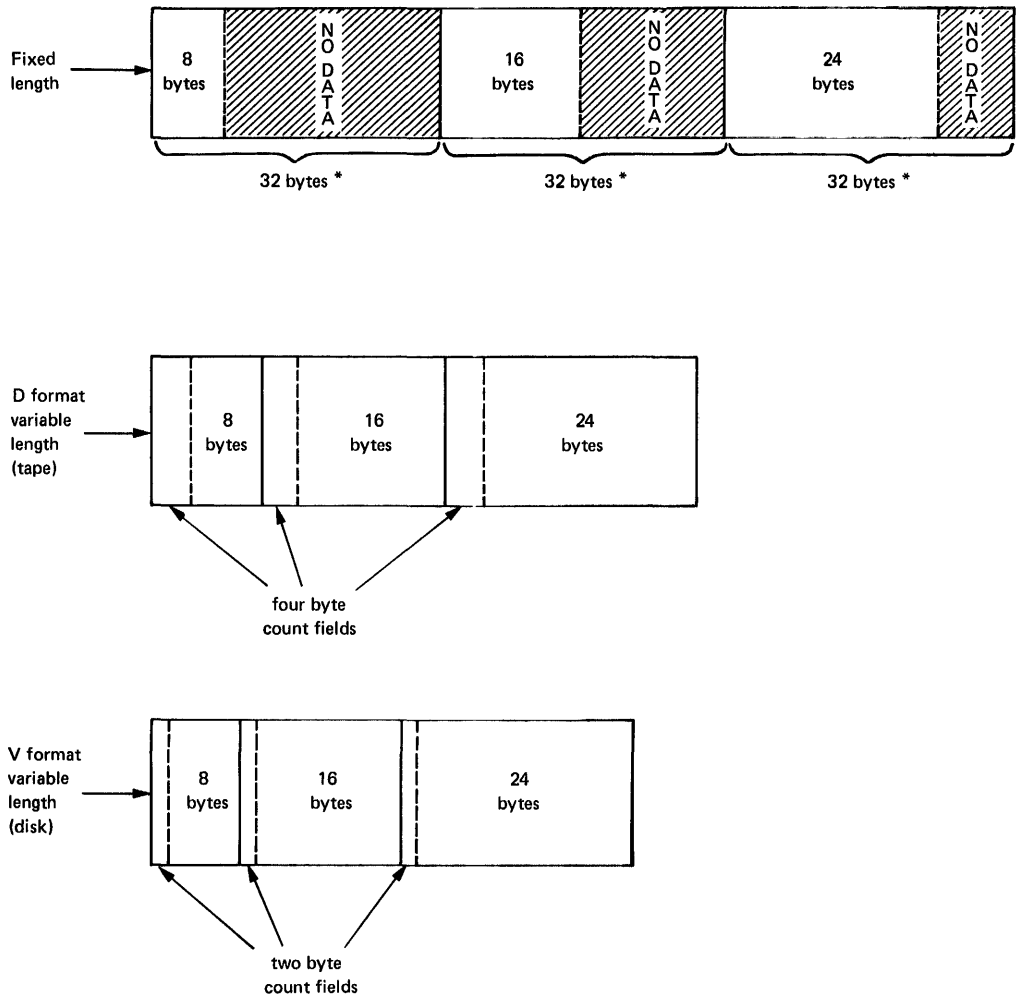
   To comply with the American National Standard X3.27-1977, when using magnetic tape volumes, VAX-11 RMS writes a 4-byte decimal count field before the data field of each record. In contrast to V-format records, this number does include the count field. Before VAX-11 RMS returns the count to you, it adjusts the count so that the value returned always reflects the length of your data.

When creating variable-length records, you can specify the size of the largest record permitted in the file. Any attempt to write a record larger than this size results in an error. If you specify a value of 0, any length record can be written in the file; note, however, that for relative and indexed files, a record must be entirely contained within a bucket.

2.5.1.3 Record Formats and File Organizations — The following paragraphs describe the effects of the various file organizations under the fixed-length record and variable-length record formats.

Figure 2–13 compares the fixed- and variable-length record formats described above, as they apply to sequential files. Each format shows a portion of a file that contains three records. The comparable record in each format contains the same number of bytes. The first record has 8 bytes, the second 16, and the third 24. For the fixed-length record format, the record length was set at 32 bytes. Therefore, VAX–11 RMS considers all 32 bytes to be used, when, in fact, unused bytes may exist as far as the user is concerned. This figure shows how much storage space the variable-length record can save. On the other hand, if records are to be updated in place, the original size of the record cannot be changed. Thus, for some applications, providing some "wasted" space achieves a certain amount of flexibility.

Figure 2–13: Comparison of Fixed- and Variable-Length Records



*VAX-11 RMS considers all 32 bytes to be used, even though they may not contain useful information in the eyes of the user.

In the relative file organization, all records, both variable length and fixed length, are written into fixed-length cells. Therefore, variable-length records do not save space in the relative file organization. In fact, two additional bytes per record are required to store the record size.

In the indexed file, variable-length records are truly variable. Their lengths are limited only by the size of the data bucket or by a user-defined maximum record size. Since record size can change on update, you need not write variable-length records at their maximum size.

2.5.1.4 Variable with Fixed-Length Control Records — Variable with fixed-length control records are similar to variable-length records.[1] However, variable with fixed-length control records include a fixed-length control field in addition to the variable-length data portion. At file-creation time, you specify two sizes. The first specification indicates the size of the largest record permitted in the file. Any record larger than this size results in an error (unless you specify 0). The second specification is the size of the fixed control area, which precedes every record in the file. You specify the size of the fixed control area once, and that size is fixed for every record in the file.

A fixed control area allows you to construct records that contain additional data that may have no direct relationship to the other contents of the record. For example, the fixed control area may contain line-sequence numbers for every record in the file. The sequence numbers would not be used in any program operations involving the record, but you would probably find them useful in locating the record on a search operation, such as in an edit of the file.

Before writing a variable with fixed-length control record to a file, you construct the record in two separate locations: the fixed control area and the data portion. When writing the record to the file, VAX-11 RMS fetches both the fixed control area and the data portion from their respective locations and joins both portions to form the record (placing the fixed control area before the data portion). VAX-11 RMS uses the record size information that you provide to determine the length of the data portion only. This joined record is then prefixed by a count field that describes the total length of the record. Figure 2-14 shows the path VAX-11 RMS takes to write a variable with fixed-length control record to a file.

When reading a variable with fixed-length control record, VAX-11 RMS uses the count field to determine the combined length of the fixed control area portion and the data portion. Then VAX-11 RMS separates the fixed control area from the data portion of the record for processing, as shown in Figure 2-15. VAX-11 RMS subtracts the fixed length of the fixed control area portion from the record count, thus returning to your program the length of the data portion only. You can ignore the fixed control area portion of a record simply by not specifying a separate buffer for the fixed control area.

[1] This record format is not applicable to indexed files.

Figure 2–14: Writing a Variable with Fixed-Length Control Record

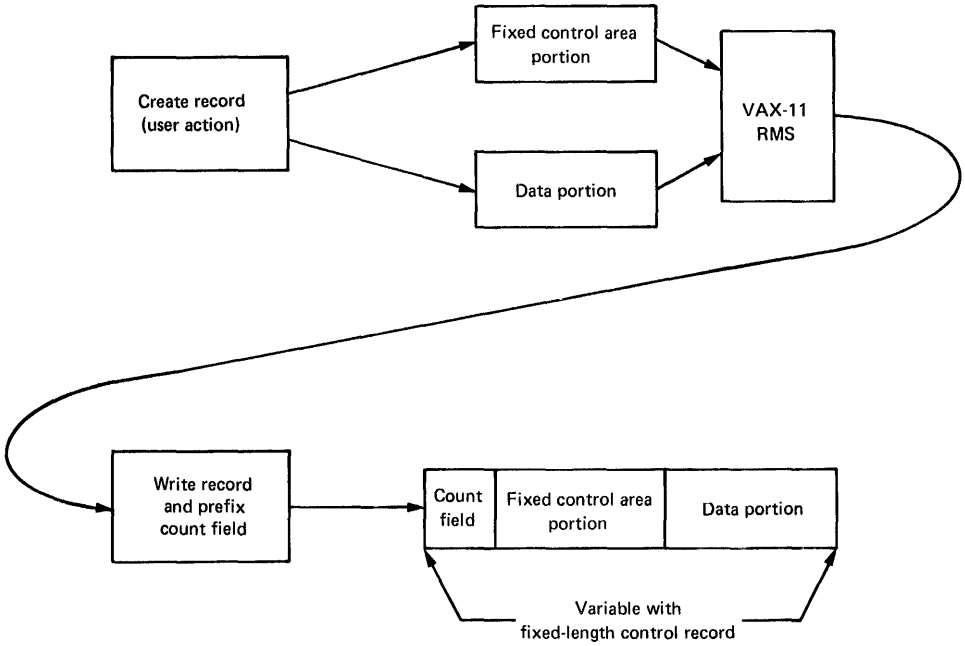
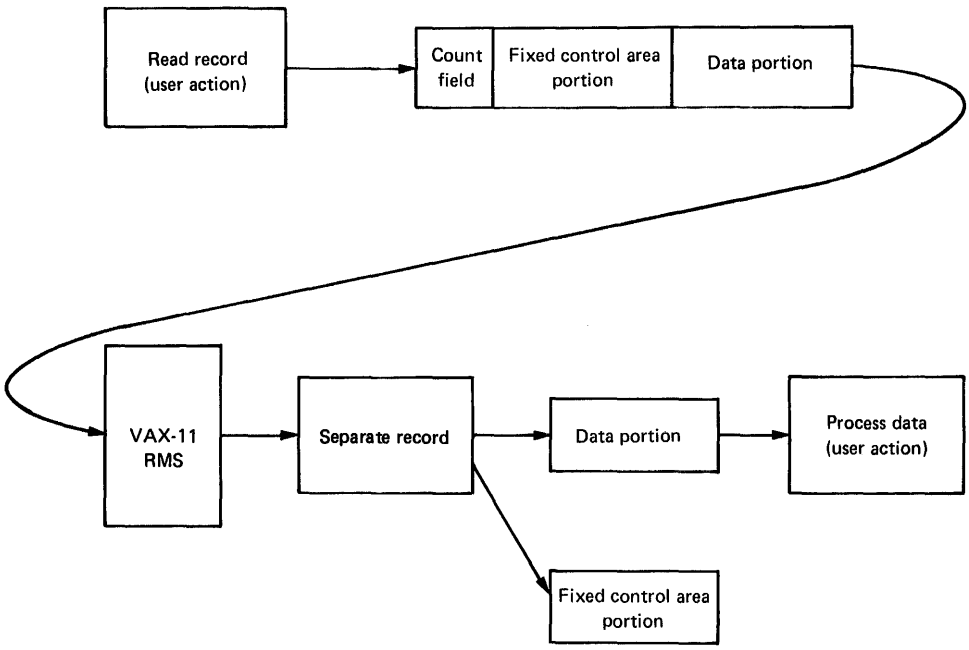


Figure 2–15: Reading a Variable with Fixed-Length Control Record



2.5.2 Storage Structure

VAX-11 RMS uses a logical storage structure called a bucket for building and processing relative and indexed files. A bucket contains from 1 to 32 blocks. You specify the number of blocks in a bucket when you create the file.

Unlike a block, however, a bucket cannot contain a portion of a record; VAX-11 RMS never permits records to span bucket boundaries. When specifying the size of a bucket, you must consider the maximum size of any record that can be written into the file, record overhead, and bucket overhead, as well as the number of records that will fit in the bucket. Since buckets are units of transfer, large buckets help provide good disk throughput for sequential operations, but require more buffer space in programs that process them.

Indexed files can be segregated into areas. Areas are portions of a file that are treated independently in terms of initial allocation, extensions, and bucket sizes. Areas enable the data records and each index of a file to be separate units. This allows the bucket sizes to be customized according to the sizes of the data records and keys. It also allows localization of the various parts of an indexed file.

2.6 File Size

The size of a file is expressed as a number of blocks. Each block within a file is a unit of data whose size depends on the physical medium on which the file resides. For example, blocks for files on disk devices are each 512 bytes. This size is established by the disk device itself and cannot be altered. On magnetic tape, a block is the information between two interrecord gaps. This size varies between 18 and 65,535 bytes, and is established by you, the user. The interrecord gaps separate the end of one block from the start of another (see Section 1.4).

When you create a disk file, as explained in Section 1.3, you can specify how many blocks are to be initially allocated to the file, in the range of 1 to 4,294,967,295. These blocks can be allocated over multiple volumes of a related volume set. VAX-11 RMS allocates the required number of blocks, and stores their retrieval information in the file header. Block numbers in a file always start at 1, and continue to the total number of blocks in the file. If you do not specify the number of blocks to be allocated, VAX-11 RMS, by default, allocates blocks depending on the space it calculates is needed for the file. This uses disk space very efficiently, because only a very small amount of space is wasted by allocating more space than necessary.

However, if you eventually need more space for the file, VAX-11 RMS must perform another allocation, a moderately time-consuming operation. (You can also specify how many additional blocks for VAX-11 RMS to automatically allocate if the file must be extended.) Furthermore, if VAX-11 RMS performs a large number of relatively small extensions to a file, the relocation information necessary to retrieve the blocks increases, thus adversely affecting retrieval performance. For large or frequently extended files, you can achieve considerable processing efficiency by specifying, to VAX-11 RMS, appropriate initial and default extension quantities.

This numbering scheme for blocks, called virtual block numbering, makes a file seem like a series of adjacent blocks. In terms of physical layout, however, blocks are not always adjacent, even though they are numbered consecutively. For example, blocks 10 and 11 are adjacent in terms of the virtual block numbering scheme, but may not be physically adjacent. One or more other blocks belonging to other files could be placed between them.

The virtually numbered blocks of a file on disk consist of groups of blocks called extents. Within an extent, consecutively numbered blocks occupy consecutive physical locations. However, a file can be segmented into several different areas on disk if not enough contiguous area is available to contain the whole file, or if different extents were allocated at different times. The blocks in each extent are contiguous, but each extent is not contiguous to the other extents in the same file. For example, blocks 1 through 5 may be in one extent, and blocks 6 through 10 may be in another.

When a file consists of only one extent, the file is contiguous, since all consecutively numbered blocks of the file correspond to consecutive physical locations. When creating a file, you can request that VAX-11 RMS allocate the file contiguously. One advantage of contiguous files on devices accessed by a single user is that disk access arm movement is minimal compared to noncontiguous files; therefore, access is faster.

The blocks of a file contain the records written into the file. Depending on the size of the records, a block can contain one record, more than one record, or only part of a record. When a particular record is larger than the size of a block, the record is stored in as many consecutively numbered blocks as necessary to contain the entire record. In this instance, the record is said to cross or span block boundaries. Total file size will vary, of course, based upon whether the records of a file are allowed to cross block boundaries.

On magnetic tapes, consecutively numbered blocks actually occupy consecutive physical locations. Blocks from one file are never interspersed with blocks from another.

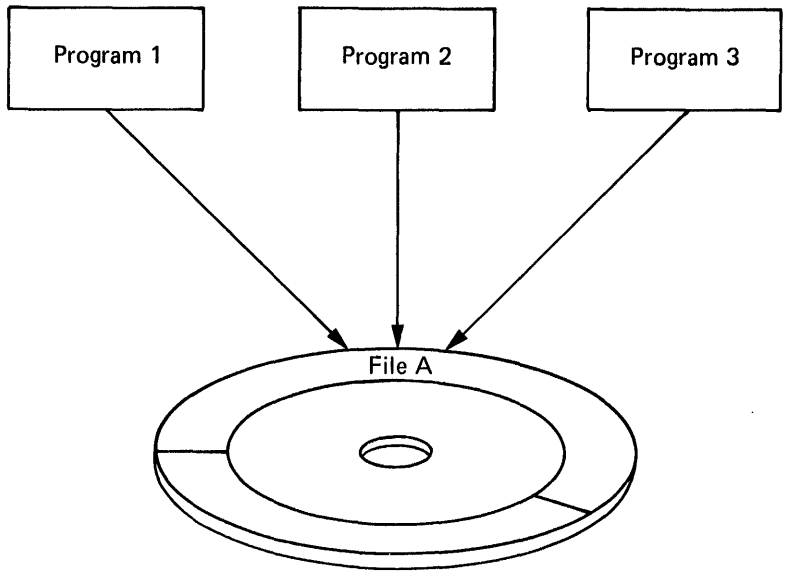
Chapter 3
File and Record Processing

3.1 Processing Levels

A program processes VAX-11 RMS files at two different levels: the file level and the record level.

At the file processing level, VAX-11 RMS and the operating system provide operations that act on the file as a whole. They also provide an environment that controls access to the file by other users and permits more than one program to access, or share, a given file at the same time, as shown in Figure 3-1.

Figure 3-1: Shared File Access



VAX-11 RMS determines the types of permissible sharing from information contained in the accessing programs.

At the record level, programs perform operations on individual records. Record operations can be performed either synchronously or asynchronously. That is, VAX-11 RMS allows your program to choose between receiving program control only after VAX-11 RMS completely satisfies a record operation request (synchronous), or possibly receiving control after VAX-11 RMS has initiated the request but before it has been completely satisfied (asynchronous). Asynchronous operations thus allow the program to overlap processing and I/O operations. This is in contrast with file-level operations, such as opening a file, which are always synchronous.

For both synchronous and asynchronous record operations, VAX-11 RMS provides two record transfer modes: move and locate. In move mode, VAX-11 RMS copies a record from an I/O buffer into a buffer which you have specified. For input operations, data is first read into the I/O buffer from a peripheral device (such as a disk), and then moved to your user buffer for processing. For output operations, you first build the record in your buffer; VAX-11 RMS then moves the record to an I/O buffer. When an entire block is filled in the I/O buffer, the block is written to the peripheral device.

In locate mode, VAX-11 RMS addresses records directly in an I/O buffer. Usually, this reduces program overhead because records can be processed directly within the I/O buffer. Locate mode is available only for input operations.

Other VAX-11 RMS facilities allow programs to control I/O buffer space allocation or simply leave all space management to VAX-11 RMS.

The following subsections briefly describe the file-level and record-level processing environments, and the different modes of file access that a program can use with VAX-11 RMS.

3.2 The File Level Operations

VAX-11 RMS provides seven operations at the file level:

- Create

  Creates a new VAX-11 RMS file of any organization. You must create a file before VAX-11 RMS can process it.

- Open

  Initiates the file processing for an existing VAX-11 RMS file. You must open a file before its records can be accessed.

- Display

  Returns the attributes of a VAX-11 RMS file to your program. Among the attributes are such items as protection information, record format and size, and file allocation information.

- Extend

  Increases the allocated size of an existing VAX-11 RMS disk file. You use this operation when you want to explicitly control file-space allocation.

- Modify

  Alters certain file attributes of an existing VAX-11 RMS disk file. Such alterations include changing the file to a noncontiguous file and changing the limit on the maximum number of records

- Close

  Terminates VAX-11 RMS file processing. Once a file is closed by a program, it cannot be accessed by the same program unless it is reopened.

- Erase

  Deletes a VAX-11 RMS disk file and makes its space available for use by another file. Erasing a file also removes its entry from the directory file.

3.2.1 File Protection

VAX-11 RMS allows you to assign a protection specification to a file at the time the file is created. This determines the file access rights of users and potential users of the file by assigning them membership in one of the following classes:

- System

  This class specifies the access rights for users that the system manager designates as privileged users.

- Owner

  This class specifies the access rights of the owner of the file. The owner is the particular member of a group to which the file belongs.

- Group

  This class specifies the access rights granted to users that the system manager designates as members of the same group as the owner of the file.

- World

  This class specifies the access rights of any user. Normally, a user with world privileges is one who is not a member of the system, owner, or group class.

Within each of the above classes, there are four separate access right specifications, representing the different types of access granted to the users in that class. These access right specifications provide for read, write, execute, and delete operations.

3.2.2 File Sharing

Timely access to files requires that more than one concurrently active program be allowed to simultaneously process the same file. Through its file-sharing capability, VAX-11 RMS permits several programs to access a file at the same time. File sharing depends on:

- Device and file organization

  Because tape units are limited to sequential operations, magnetic tape files cannot be shared. Disk files, however, can be shared by multiple processes.

  Unrestricted sharing by any combination of readers and writers of relative and indexed files is permitted. In addition, sharing is allowed for sequential files with 512 byte fixed-length records. Sharing of other types of sequential files, however, is permitted only with restrictions.

- Explicit file-sharing information specified by the accessing programs

  Although the organization of a file determines whether multiple readers and writers can access the file, your program provides the information that allows the actual sharing. You control the degree of sharing of the file by an explicit sharing specification that you give to VAX-11 RMS when you create or open the file. The sharing specification indicates which types of operations can be used by the sharing users.

  When your program opens a file, it must provide two types of information. First, it must state what type of operations it intends to perform, such as read, write, or update. The intended operations are then checked against the protection for the file. Second, the program must specify which types of operations other concurrently active programs can perform on the file. The combination of these two types of information allows VAX-11 RMS to determine which programs can gain access to the file. When the sharing specification of one program is compatible with the sharing specification of another program, both programs can gain access to the file simultaneously.

  File sharing is supported only for native-mode programs using VAX-11 RMS. Programs (tasks) that run in compatibility mode using the RSX-11 Applications Migration Executive and RMS-11, cannot share files with native mode programs using VAX-11 RMS or with other programs running in compatibility mode and using RMS-11.

3.3 The Record Level Operations

VAX-11 RMS provides five major operations at the record level:

- Get (Read)

  Reads (retrieves) a record from a file

- Put (Write)

  Writes (inserts) a record into a file

- Find

  Locates a record in a file

- Update

  Alters the contents of a record that currently exists in a file

- Delete

  Removes an existing record from a file

See the *VAX-11 Record Management Services Reference Manual* for other record-level operations.

3.3.1 Synchronous and Asynchronous Modes of Operation

Your program can handle record operations on a file in either of two ways: synchronously or asynchronously. When operating synchronously, the pro-

gram issuing the record operation request regains control only when the request is completely satisfied. Most high-level languages support synchronous operation only. In asynchronous operations, the program can regain control before the request is completely satisfied.

For instance, when synchronously reading a record from a file, the program regains control only after the record is passed to the program. In other words, the program is in a wait state until the record returns; no other processing (for this program) takes place during this read-and-return cycle. On the other hand, when asynchronously reading a record, the program might be able to regain control before the record is passed to the program. The program can thus use the time normally required for the record transfer between the file and memory to perform some other computations. However, another record operation cannot be started on the same stream (an access-window to a file, associated with a record access control block supporting record operation requests) until the prior record operation is complete. Record operations on other streams can be initiated.

Whether the program will actually regain control before the record operation is complete depends on several factors. For example, the required record may already reside in the I/O buffer (see Section 3.3.2); or the operating system may schedule another program, thus possibly allowing a necessary I/O operation to be completed before the original program is rescheduled.

One factor to consider in the use of asynchronous record operations is that you must include a separate completion routine or VAX-11 RMS wait request in the issuing program. This routine is required to determine when the record operation is complete, since the next record operation for that stream cannot be initiated until the prior operation is completed.

3.3.2 Record Buffering

VAX-11 RMS uses internal memory areas called I/O buffers to read or write blocks of data. To your program, record processing under VAX-11 RMS appears as the movement of records directly between a file and the program itself. This is, in fact, not the case. Transparently to your program, VAX-11 RMS reads or writes blocks or buckets of a file into or from an I/O buffer. Records within the buffer are then made available to the program.

The transfer of data between a file and the I/O buffers depends on the file's organization. For the sequential organization, VAX-11 RMS reads and writes a block or series of blocks. For relative and indexed organizations, VAX-11 RMS reads and writes buckets.

3.3.3 Record Transfer Modes

In addition to specifying synchronous or asynchronous record operations, your program can use either of two record transfer modes to gain access to each record in memory. These modes are move mode and locate mode.

VAX-11 RMS permits move mode record operations for all file organizations. Move mode requires that an individual record be copied between the I/O

buffer and your program buffer. For read operations, VAX-11 RMS transfers a block (or series of blocks or a bucket, depending on the file organization and size of the I/O buffer) into an I/O buffer, finds the desired record within the buffer, and then moves the record to a program-specified location (a user buffer). For a write or update operation, your program must first build or modify the record in a program-defined work area. When you issue a write request, VAX-11 RMS moves the record to the I/O buffer that VAX-11 RMS will use for writing to the file.

In contrast to move mode, in locate mode the data records stay in place; VAX-11 RMS need not copy records from an I/O buffer to a user buffer. A pointer is set up to allow your program to access the records directly. This reduces the amount of data movement, thereby saving processing time. This mode is actually a partial locate mode because VAX-11 RMS cannot always avoid moving data records as, for example, when records cross block boundaries. Because of this, a user buffer is still required.

3.3.4 Record Locking

VAX-11 RMS can lock records to control operations to a relative or indexed file that more than one record stream within a process, or more than one process, can access simultaneously. This record-locking facility ensures that a program can add, delete, or modify a record on one stream without the program's needing to check that the same record is not simultaneously accessed by another stream or process. For example, when your program opens a file with the declared intention of writing or updating records on multiple streams, VAX-11 RMS locks the record accessed by the program. This locking prevents another stream or process from accessing the same record until the first stream or process releases the record.

VAX-11 RMS also handles the automatic locking of the entire bucket that contains the record for the short period required to initially access the record. This automatic lock also occurs later, when the actual update takes place. In the interim, the record remains locked, but other records in the bucket can be accessed and modified as required.

There are facilities you can use to allow VAX-11 RMS to handle automatic record locking and unlocking; facilities also exist that give you explicit control over the release of record locks.

VAX-11 RMS does not provide record locking support for sequential files with other than 512 byte fixed-length records. However, the user can provide the necessary synchronization logic to allow simultaneous reading and writing of those files.

3.4 Creating the File

You can use several methods to create a VAX-11 RMS file. The most common is to issue a $CREATE macro instruction (or its equivalent in a high-level language) within an application program. At creation time, you specify the file definition information, which VAX-11 RMS stores within the file for reference in later processing. A second method is to use the VAX-11 RMS DEFINE utility program.

Note that neither of the above creation procedures actually stores records in the file; this must be done by your application program or by the VAX-11 RMS CONVERT or Indexed File Load utility programs.

3.5 Processing Records in a File

The organization of a file determines the types of operations that a program can perform on records. VAX-11 RMS can perform the following major operations:

- Read a record — VAX-11 RMS returns an existing record within the file to your program.
- Write a record — VAX-11 RMS adds a new record to the file.
- Find a record — VAX-11 RMS locates an existing record in the file. It does not return the record to your program, but establishes a new current position in the file (see Section 2.4.4).
- Delete a record — VAX-11 RMS removes an existing record from a file.
- Update a record — Your program modifies the contents of a record that has been read from the file into a buffer area. VAX-11 RMS writes the modified record into the file, replacing the old record.

Table 3-1 shows the combinations of major record operations and file organizations that VAX-11 RMS permits, and notes the exceptions to these operations.

Table 3-1: Record Operations and File Organizations

Record Operation Permitted	File Organization		
	Sequential	Relative	Indexed
Read (Get)	Yes	Yes	Yes
Write (Put)	Yes[1]	Yes	Yes
Find	Yes	Yes	Yes
Delete	No	Yes	Yes
Update[3]	Yes[2]	Yes	Yes

[1] In a sequential file, VAX-11 RMS allows records to be added at the end of the file only. (Records can be written to other points in the file through an update operation.)

[2] When performing an update operation to a sequential file, the user cannot change the length of the record.

[3] VAX-11 RMS allows update operations on disk devices only.

3.5.1 Processing Sequential Files

In a sequential file, a program can read existing records from the file using the sequential record access mode. If the file resides on disk, the random by RFA record access mode can also be used (see Section 2.4); and, if the file is of the fixed-length record format, the random by relative record number record access mode is permitted. You can add records only at the end of the file.

All record access modes (sequential, random by relative record number, and random by RFA), allow you to establish a new current position for a record in the file (find operation). In the sequential record access mode, you can skip records with a find operation. In either the random by relative record number or the random by RFA record access mode, a find operation can establish a random starting point in the file for sequential read operations.

The sequential file organization does not allow the deletion of records from the file since this organization requires that records be adjacent to each other. However, you can update an existing record in a sequential file as long as:

1. The record resides in a file on a disk
2. The modification of the record does not alter its size

3.5.2 Processing Relative Files

The relative file organization permits greater program flexibility in performing record operations than the sequential organization. A program can read existing records from the file using sequential, random by key (relative record number), or random by RFA record access modes. You can write new records either sequentially or randomly, as long as the intended record position (cell) does not already contain a record.

All record access modes for relative files allow you to position to a record in the file (find operation). After finding the record, VAX-11 RMS permits you to delete the record from the relative file. After record deletion, the empty cell is available to a new record. In addition, your program can update records anywhere in the file. If the records are variable length, the update operation can modify the record length up to the maximum size specified when the file was created.

3.5.3 Processing Indexed Files

The indexed file organization provides the greatest flexibility in performing record operations. A program can read existing records from the file in sequential, random by RFA, or random by key access mode. When reading records in random by key access mode, the program specifies one of four types of matches:

1. Exact key match
2. Approximate key match
3. Generic key match
4. Approximate and generic key match

Exact key match requires that the key value in the retrieved record precisely match the key value specified by the program read operation.

Approximate key match allows the program to select either one of the following relationships between the key value of the retrieved record and the key value specified by the program:

1. Equal to or greater than
2. Greater than

The advantage of this kind of match is that if the requested key value does not exist in any record of the file, VAX-11 RMS returns the record that contains the next higher key value. This allows the program to retrieve records without knowing an exact key value.

Generic key match means that the program need specify only an initial portion of the key value. VAX-11 RMS then returns to the program the first occurrence of a record whose key value begins with these characters. This capability is useful in applications in which a series of records must be retrieved when only the initial portions of their key values are identical.

The final type of key match combines both the generic and approximate facilities. The program specifies only an initial portion of the key value, as with generic match. In addition, the program specifies that the key value of the retrieved record must be either one of the following:

1. Equal to or greater than the program-supplied value
2. Greater than the program-supplied value

VAX-11 RMS also allows you to write any number of new records into an indexed file. It rejects a write operation only if a key value in a record violates a user-defined key characteristic, such as duplicate key values not allowed.

You can perform a find operation, similar to the read operation, in sequential, random by RFA, or random by key access mode. When finding records in random by key value access mode, the program can specify any one of the four types of key matches provided for read operations.

In addition to reading, writing, and finding a record, your program can delete or update any record in an indexed file. During an update operation, the modification cannot violate user-defined key characteristics; for example, if key values cannot change, the update is rejected when that condition is detected.

3.6 Block Input/Output

Block input/output (I/O) lets programs bypass the VAX-11 RMS record processing capabilities entirely. Thus, rather than perform record operations by means of the supported record access modes, you can process a file as a logical structure consisting solely of a number of blocks.

Block I/O operations provide you with an intermediate step between VAX-11 RMS operations and coding directly at the explicit I/O instruction level of the

system services. Using block I/O allows you full control of the data of the individual blocks of a file while also enabling you to take advantages of the VAX-11 RMS capabilities for opening, closing and extending a file.

In block I/O, a program reads or writes one or more blocks by specifying a starting virtual block number in the file and the length of the transfer. Regardless of the organization of the file, VAX-11 RMS accesses the identified block or blocks for you.

Because VAX-11 RMS files contain internal information meaningful only to VAX-11 RMS itself, DIGITAL does not recommend that you modify an existing file using block I/O if the file is also to be accessed by VAX-11 RMS record-level operations. (Block I/O does not update the internal record information correctly.) The block I/O facility, however, does allow you to create your own file organizations. This file structure must be maintained through specialized user-written programs and procedures; VAX-11 RMS cannot access these structures through its record access modes.

Glossary

alternate key

An optional key within the data records in an indexed file; used by VAX-11 RMS to build an alternate index. See key (indexed files) and primary key.

area

Areas are VAX-11 RMS maintained regions of an indexed file. They allow a user to specify placement and/or specific bucket sizes for particular portions of a file. An area consists of any number of buckets, and there may be from 1 to 255 areas in a file.

asynchronous record operation

An operation in which your program may possibly regain control before the completion of a record retrieval or storage request. Completion ASTs and the $WAIT MACRO are the mechanisms provided by RMS for programs to synchronize with asynchronous record operations. See synchronous record operation.

bits per inch

The recording density of a magnetic tape. Indicates how many characters can fit on one inch of the recording surface. See density.

block

A group of consecutive bytes of data treated as a unit by the storage medium. A block on a Files-11 disk structure is 512 eight-bit bytes.

block I/O

The set of VAX-11 RMS procedures that allow you direct access to the blocks of a file regardless of file organization.

bootstrap block

A block in the index file of a system disk. Can contain a program that loads the operating system into memory.

bpi

See bits per inch.

bucket

A storage structure, consisting of from 1 to 32 blocks, used for building and processing relative and indexed files. A bucket contains one or more records or record cells. Buckets are the unit of contiguous transfer between VAX-11 RMS buffers and the disk.

buffer

An internal memory area used for temporary storage of data records during input or output operations.

cluster

The basic unit of space allocation on a Files-11 disk volume. Consists of one or more contiguous blocks, with the number being specified when the volume is initialized.

compatibility mode

A mode of execution that enables the central processor to execute nonprivileged PDP-11 instructions. The operating system supports compatibility mode execution by providing an RSX-11M programming environment for an RSX-11M task image. The operating system compatibility mode procedures intercept calls to the RSX-11M executive and convert them to the appropriate operating system functions.

contiguous area

A group of physically adjacent blocks.

cylinder

The tracks at the same radius on all recording surfaces of a disk.

density

The number of bits per inch of magnetic tape. Typical values are 800 bpi and 1600 bpi. See bits per inch.

directory

A file used to locate files on a volume. A directory file contains a list of files and their unique internal identifications.

dynamic access

Term applied to the switching from one type of record access mode to another while processing a file.

extent

One or more adjacent clusters allocated to a file or a portion of a file.

file

An organized collection of related items (records) maintained in an accessible storage area, such as disk or tape.

file header

A block in the index file describing a file on a Files–11 disk structure, including the location of the file's extents. There is at least one file header for every file on the disk.

file organization

The physical arrangement of data in the file. You select the specific organization from those offered by VAX–11 RMS, based on your individual needs for efficient data storage and retrieval. See indexed file organization, relative file organization, and sequential file organization.

files–11

The standard physical disk structure used by VAX–11 RMS.

fixed control area

An area, prefixed to a variable-length record, containing additional information that can be processed separately and that may have no direct relationship to the other contents of the record. For example, the fixed control area might contain line sequence numbers for use in search operations.

fixed-length record format

Property of a file in which all records are of the same size. This format provides simplicity in determining the exact location of a record in the file and eliminates the need to prefix a record size field to each record.

home block

A block in the index file, normally next to the bootstrap block, that identifies the block as a Files–11 volume and provides specific information about the volume, such as volume label and protection.

index

The structure which allows retrieval of records in an indexed file by key value. See key (indexed files).

index file

A file on each Files–11 volume that provides the means for identification and initial access to the volume. Contains the access data for all files (including itself) on the volume: bootstrap block, home block, file headers.

indexed file organization

A file organization which allows random retrieval of records by key values and sequential retrieval of records in sorted order by key value. See key (indexed files).

interrecord gap

An interval of blank space between data records on the recording surface of a magnetic tape. Allows the tape unit to decelerate, and stop if necessary, after a record operation and then accelerate before the next record operation.

IRG

See interrecord gap.

key

Indexed files: a character string, a packed decimal number, a 2- or 4-byte unsigned binary number, or a 2- or 4-byte signed integer within each data record in an indexed file. You define the length and location within the records; VAX–11 RMS uses the key to build an index. See primary key, alternate key, and random access by key value.

Relative files: The relative record number of each data record in a data file; VAX–11 RMS uses the relative record numbers to identify and access data records in a relative file in random access mode. See relative record number.

locate mode

Technique used for a record input operation in which the data records are not copied from the I/O buffer. See move mode.

move mode

Technique used for a record transfer in which the data records are copied between the I/O buffer and your program buffer for calculations or operations on the record. See locate mode.

multiple-extent file

A disk file having two or more extents.

native mode

The processor's primary execution mode in which the programmed instructions are interpreted as byte-aligned, variable-length instructions that operate on the following data types: byte, word, longword, and quadword integers; floating and double floating character strings; packed decimals; and variable-length bit fields. The other instruction execution mode is compatibility mode.

primary key

The mandatory key within the data records of an indexed file; used by VAX-11 RMS to determine the placement of records within the file and to build the primary index. See key (indexed files) and alternate key.

random access by key

Indexed files only: Retrieval of a data record in an indexed file by either a primary or alternate key within the data record. See key (indexed files).

Relative files only: Retrieval of a data record in a relative file by the relative record number of the record. See key (relative files).

random access by record's file address

Retrieval of a record by the record's unique address, which VAX-11 RMS returns to you. This record access mode is the only means of randomly accessing a sequential file containing variable-length records.

random access by relative record number

Retrieval of a record by its relative record number. See relative record number. For relative files, random access by relative record number is synonymous with random access by key. See random access by key (relative files only).

record

A set of related data that your program treats as a unit.

record access mode

The manner in which VAX-11 RMS retrieves or stores records in a file. Available record access modes are determined by the file organization and specified by your program.

record blocking

The technique of grouping multiple records into a single block. On magnetic tape an IRG is placed after the block rather than after each record. This technique reduces the number of I/O transfers required to read or write the data; and, in addition (for magnetic tape), increases the amount of usable storage area. Record blocking also applies to disk files.

record format

The way a record physically appears on the recording surface of the storage medium. The record format defines the method for determining record length.

record length

The size of a record; that is, the number of bytes in a record.

record locking

A facility that prevents access to a record by more than one record stream or process until the initiating record stream or process releases the record.

record cell

A fixed-length area in a relative file that can contain a record. The concept of fixed-length record cells lets VAX–11 RMS directly calculate the record's actual position in the file.

record's file address

The unique address of a record in a file, which is returned by RMS whenever a record is accessed, allows records in disk files to be accessed randomly regardless of file organization. This address is valid only for the life of the file. If an indexed file is reorganized, then the RFA of each record will typically change.

relative file organization

The arrangement of records in a file where each record occupies a cell of equal length within a bucket. Each cell is assigned a successive number, called a relative record number, which represents the cell's position relative to the beginning of the file.

relative record number

An identification number used to specify the position of a record cell relative to the beginning of the file; used as the key during random access by key mode to relative files.

reorganization

A record by record copy of an indexed file to another indexed file with the same key attributes as the input file.

RFA

See record's file address.

RMS–11

A set of routines which are linked with compatibility mode programs, and provide similar functional capabilities to VAX–11 RMS. The file organizations and record formats used by RMS–11 are identical to those of VAX–11 RMS.

sequential file organization

The arrangement of records in a file in one-after-the-other fashion. Records appear in the order in which they were written.

sequential record access mode

Record storage or retrieval which starts at a designated point in the file and continues in one-after-the-other fashion through the file. That is, records are accessed in the order in which they physically appear in the file.

synchronous record operation

An operation in which your program does not regain control until after the completion of a record retrieval or storage request. See asynchronous record operation.

stream

An access window to a file associated with a record access control block, supporting record operation requests.

track

A collection of blocks at a single radius on one recording surface of a disk.

variable-length record format

Property of a file in which records need not be of the same size.

variable with fixed-length control record format

Property of a file in which records of variable-length contain an additional fixed control area capable of storing data that may have no bearing on the other contents of the record. Variable with fixed-length control record format is not applicable to indexed files.

VAX–11 Record Management Services (VAX–11 RMS)

The file and record access subsystem of the VAX/VMS operating system for the VAX–11. VAX–11 RMS helps your application program process records within files, thereby allowing interaction between your application program and its data.

volume

Disk: An ordered set of 512-byte blocks. The basic medium that carries a Files–11 structure.

Magnetic tape: A reel of magnetic tape, which may contain a part of a file, a complete file, or more than one file.

volume set

A collection of related volumes.

Index

A

B

C

D

E

F

H

I

K

L

M

N

P

R

S

T

V

W

READER'S COMMENTS

NOTE: This form is for document comments only. DIGITAL will use comments submitted on this form at the company's discretion. If you require a written reply and are eligible to receive one under Software Performance Report (SPR) service, submit your comments on an SPR form.

Did you find this manual understandable, usable, and well-organized? Please make suggestions for improvement.

Did you find errors in this manual? If so, specify the error and the page number.

Please indicate the type of user/reader that you most nearly represent.

- ☐ Assembly language programmer
- ☐ Higher-level language programmer
- ☐ Occasional programmer (experienced)
- ☐ User with little programming experience
- ☐ Student programmer
- ☐ Other (please specify)_____

Name_____ Date_____

Organization_____

Street_____

City_____ State_____ Zip Code_____
or
Country

Do Not Tear - Fold Here and Tape

digital

No Postage
Necessary
if Mailed in t
United Stat

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO.33 MAYNARD MASS.

POSTAGE WILL BE PAID BY ADDRESSEE

BSSG PUBLICATIONS TW/A14
DIGITAL EQUIPMENT CORPORATION
1925 ANDOVER STREET
TEWKSBURY, MASSACHUSETTS 01876

Do Not Tear - Fold Here