COMPUTER CORPORATION OF AMERICA

(14)

(6) DATACOMPUTER PROJECT •

(9) SEMI-ANNUAL TECHNICAL REPORT. 1 Jul - 31 Dec 76.

(11) 31 Dec 76

July 1, 1976 - December 31, 1976

(12) 68p.

(15) Contract No. MDA903-74-C-0225

ARPA Order No. 2687





Submitted to:

Defense Advanced Research Projects Agency
1400 Wilson Boulevard
Arlington, Virginia 22209



397 285

Computer Corporation of America

575 Technology Square

Cambridge, Massachusetts 02139

# DATACOMPUTER PROJECT

# SEMI-ANNUAL TECHNICAL REPORT

July 1, 1976 to December 31, 1976



This research was supported by the Defense Advanced Research Projects Agency of the Department of Defense and was monitored by the U.S. Army Research Office, Defense Supply Service -- Washington under Contract No. MDA903-74-C-0225. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the U.S. Government.

Table of Contents

Table of Contents

# Chapter 1

## Introduction

This report describes our work on the Datacomputer system from July 1, 1976 through December 31, 1976. The project is supported by the Information Processing Techniques Office of the Advanced Research Projects Agency of the Department of Defense. The current work is being carried out under contract MDA903-74-C-0225. Related work discussed herein is supported by the Nuclear Monitoring Research Office of ARPA under contract MDA903-74-0227.

Work during the reporting period falls into two of the following categories: production operation of Datacomputer Version 1; preparation and release of Datacomputer Version 2, the first version of the Datacomputer to support an Ampex Terabit Memory System; production operation of Version 2; preparation and release of Datacomputer Version 3.

Chapters 2 - 6 provide detailed descriptions of this work. Chapter 2 is a discussion of the Datacomputer architecture, with emphasis on the increasing levels of functional abstraction beginning with the hardware and moving outward. Chapter 3 is a report on the usage of the Datacomputer during the reporting period. Chapter 4 is a detailed discussion of the work on the Datacomputer software.

Chapter 5 describes Datacomputer site, hardware, and operations work. Chapter 6 is a brief overview of the NMRO work and its implications for the Datacomputer in general. Finally, Chapter 7 presents a brief summary of technical development over the duration of this contract.

# Chapter 2

## System Description

The Datacomputer is a very large scale data storage facility with substantial data management capabilities. Its design is optimized for use as a data resource in a network of large-scale computers which are connected via medium speed (50,000 bits/second) communications lines. The data storage functions of the Datacomputer will support the storage of data sets over a trillion bits, and the hardware facilities include an expandable Ampex TBM currently configured for 200 billion bits of storage (other mass storage systems could also be used).

The development of the Datacomputer has been strongly affected by its nature as a network data utility. Its design does not preclude very fast data transfers -- the Datacomputer can feed data to the network or some other interface at speeds approximating the bandwidth of its storage devices, as long as no special processing is required -- but this is not how it operates in the most frequent case. The combination of very large storage capacities and only moder-

---

(1) An earlier Semi-annual Technical Report contained a lengthy tutorial section titled "System Description," regarding which a number of positive comments have been received. In the interest of making the present document self-contained, a somewhat shortened and updated version of the same material is included here.

ately fast communications facilities implies the Datacomputer must provide powerful facilities for data selection and subsetting, in particular, to minimize data transmission back and forth through the network. (To transmit a trillion bits through a 50,000 bits/second channel requires about 231 days, assuming no errors or interruptions!) Furthermore, in order to make simple changes to large numbers of records, a facility is necessary for self contained requests which modify files without data transmission over the network.

The Datacomputer's existence in a network environment also implies that other computer systems mediate between the Datacomputer and its ultimate human users. Consequently, functions that are not intrinsic to data management, such as carefully human-engineered terminal interfaces, are relegated to those other systems. This has several benefits: it allows work at the Datacomputer to concentrate on issues intrinsic to data storage and management. It allows other systems which concentrate on human engineering to provide better interfaces than would otherwise be produced. Most importantly, it ensures that the Datacomputer is developed as a resource available to other systems, capable of being incorporated into larger projects. Careful attention is paid to such issues as error flagging and resynchronization after serious error detection.

## 2.1 Levels of Functional Abstraction

Many large computer systems may be usefully examined in terms of their functional hierarchies. A level may be characterized in two ways. First, more fundamental operations which are provided by the previous level (and may already be abstractions themselves) are combined into new, more powerful, and more abstract operations. For example, the stream of magnetic flux reversals seen by the disk controller becomes a stream of fixed (or variable) length blocks of binary words when seen by the operating system. A subset of this arbitrary collection of unformatted words is presented to user programs as a "file" in a "file system". Second, intermediate functions exist to prevent certain combinations of operations which would damage system integrity from occurring, and to hide other functions entirely from the next level out.

The term normally used for the particular collection of functions available to any given level of a system hierarchy is "virtual machine". In many ways, the programmer working at level n in such a system may behave as if level n-1 were hardware; All n-1 functions are immutable and part of the

machine environment. Using terms that will be explained in the rest of this section, the TENEX implementer programs a PDP-10; the SV programmer programs a TENEX (which looks a lot like a PDP-10 with some major abstractions); the Request Handler programmer programs an SV machine, and the ultimate user programs a Datacomputer. (We shall see that the set of functions presented by the Request Handler is equivalent to the Datacomputer virtual machine.)

The following levels will be discussed in detail:

1) The hardware consists of a Digital Equipment Corporation (DEC) PDP-10 and its supporting peripherals including communications links to the ARPA Network and a very large storage device.

2) The TENEX operating system is in direct control of the hardware resources and provides many services to the Datacomputer.

3) The programs known collectively as SV or Services are a pseudo-operating system which interacts with TENEX, managing input/output, scheduling, and storage strategies for Datacomputer files.

4) The Request Handler (RH) is the interface to external processes using the Datacomputer. It accepts control and data-management statements in

"Datalanguage", provides messages concerning the state of the Datacomputer job to the user, and supervises data flow in both directions under the control of Datalanguage statements, and with the help of the other levels of the system.

Above these four levels of processing, there are an indefinite number of levels of functioning, outside the actual Datacomputer. These are the processes on other machines which interface to the Datacomputer, and co-operate with it in the accomplishment of whatever tasks its ultimate users undertake.

## 2.2 The Hardware Level

Conceptually, the hardware for a Datacomputer is quite simple. A processor of some sort is required along with some form of primary store (e.g., core). In addition, one needs a very large store (e.g., TBM) and a medium-to-high speed communications port. A great deal of efficiency can be gained by adding one or more levels of intermediate storage such as disk.

The hardware base of the Datacomputer as it is currently implemented consists of a processor, an address mapping device, three levels of store, medium and low speed communications lines, and several I/O devices.

The processor is a Digital Equipment Corporation KA-10 CPU (PDP-10). A Bolt Beranek and Newman "Pager" provides address translation for all memory references, and (along with software in TENEX) provides the illusion of a 256K (1K = 1024) word primary store regardless of the size of the physical memory.

The real primary store in the current Datacomputer is 336K words of 36 bit core memory. This includes five 16K DEC ME-10's and two 128K STOR-10's from Cambridge Memories, Inc. PDP-10 characters are typically stored five to a word, so this is the equivalent of slightly more than a million characters of memory.

The system has two types of secondary store. Six spindles of DEC RP02 disk (IBM 2314 equivalent) provide space for the TENEX file system. These hold about four million 36 bit words each, for a total of 24 million words. In addition, four spindles of CalComp 230 disk (IBM 3330 equivalent) are attached to the PDP-10 via a Systems Concepts SA-10 IBM data channel simulator. These disks each will store approximately 20 million words of data and are

used for staging devices between the tertiary store and the PDP-10.

The main data repository of the Datacomputer is its tertiary store, an Ampex Tera-Bit Memory System also attached through the SA-10. The TBM is described in detail in Chapter 5 (Hardware / Site / Operations).

The Datacomputer's communications equipment consists of a connection to an Interface Message Processor (IMP) which is in turn connected to the Arpanet. The Arpanet connection is the Datacomputer's only channel to the outside world. All Datacomputer usage consists of messages and data passed back and forth through this port. Nodes in the Arpanet are connected by up to four 50,000 bit/second phone lines, and the combined traffic of all concurrent Datacomputer is limited by this (except for the special case of usage from another host connected to the same IMP).

## 2.3 The Primary Operating System - TENEX

The second level in the Datacomputer's functional hierarchy is the TENEX operating system. An excellent overview of the nature and facilities of TENEX can be found in

"TENEX, a Paged Time Sharing System for the PDP-10."(1)

### 2.3.1 TENEX Modifications for the Datacomputer

The virtual machine provided by TENEX - a PDP-10 arithmetic processor with full memory capabilities and file/-process address space integration - has proved to be satisfactory for Datacomputer development. However, changes to the TENEX monitor have been necessary to optimize the Datacomputer's performance. The Datacomputer's Network Control Program has been modified to optimize voluminous data transfers rather than the high level of short message (due to user terminal I/O) that is more typical of TENEX network traffic. The scheduler was modified to give special considerations to the resource utilization patterns of the Datacomputer. For example, there is little urgency in the Datacomputer to satisfy highly interactive jobs, but it is necessary to respond promptly to events like high-speed disk operations. Higher throughput with reduced overhead is achieved by rescheduling at shorter intervals than in a normal TENEX, but giving larger quanta of CPU time when jobs are scheduled. Routines to support the Calcomp 230 disk drives and the TBM have been added.

---

(1) Bobrow, Daniel G., et al., "TENEX, a Paged Time Sharing System for the PDP-10, "Communications of the ACM, V. 15, no. 3, March 1972, pp. 135 - 143.

## 2.4 The Pseudo Operating System - Services

The preceding two levels of the Datacomputer system were not products of the development effort being discussed. They are described here because an understanding of their functions and capabilities is useful to understanding the functions and capabilities of the two outer layers of the system - Services and the Request Handler.

These two levels constitute what could reasonably be called the "Datacomputer proper", and are the primary output of the Datacomputer project. They are conceptually and functionally separate - to the point of having separate personnel. This section discusses the Services programs (hereafter known interchangeably, and in accordance with time-honored tradition, as SV).

SV functions as a pseudo operating system for the Datacomputer. It provides the basic functions of a traditional operating system in a form which is maximally convenient for the construction of a user-level Datacomputer interface (of which the current Request Handler is but one example). In particular, SV provides a specialized file system, stream oriented input/output facilities, and a set of scheduling/-monitor functions.

The special disk area index module, known as SDAX, serves two critical functions. First, it controls the staging mechanism which brings required data pages from relatively slow tertiary store (TBM) to relatively fast secondary store (3330 and RP02 disk). Pages thus stored are kept on disk as long as they are in use, and are migrated back to TBM only after they are no longer needed by active users. The second function of SDAX is to provide a mechanism for permitting access to several versions of a file by any number of users. It does this by a map-chaining technique which permits multiple readers and a concurrent updater to access any number of active versions of a file.

Access to SV functions for the Request Handler is via a special instruction known as "SVCALL". SVCALL's exist to manipulate the state of Datacomputer files including reading and writing pages from them; to perform input and output over the Datacomputer's Arpanet connections, and to handle special error conditions.

### 2.4.1 The SV File System

The primary function of SV is to provide a convenient interface to the data storage facilities of the Datacomputer: the tertiary store and the staging device. A Data-

computer file as seen by the Request Handler programmer consists of a number of "sections", each of which is an ordered set of pages. (For convenience, SV pages are the same size as TENEX pages - 512 36 bit words.)

### 2.4.2 The Directory System

The Datacomputer file system may be thought of as a tree-structured hierarchy. At the top of the tree is a node whose conventional name is "%TOP". There are two types of nodes in the directory system - terminal and non-terminal. Terminal nodes (files) contain only data, and non-terminal nodes (directories) contain other nodes which exist at a lower level in the tree. Node creation is independent of the intended use of the node. In other words, a node in the tree is created, then at a later time it is specified whether it is terminal (a file) or non-terminal (a directory). Levels of the hierarchy are specified as a list of names connected by periods, such as "%TOP.DFTP.CCA". In the example, %TOP and DFTP are non-terminal nodes, and CCA may be either terminal or non-terminal (in the example, not enough context is present to determine which).

In addition to maintaining the directory hierarchy, the directory system provides protection for contents of nodes, whether other nodes or data. This protection takes the form of a set of "privilege tuples" associated with each node. A privilege tuple describes two things: the set of privileges allowed (or denied) to the user accessing the node, and the specification of the class of users to whom this particular set of privileges applies.

Just to give the flavor of privilege tuple application, one tuple might specify that, for a particular node, a user may login to the node and create new nodes under it, but only if connected to the Datacomputer from socket number 1000001 on ARPANET host number 31. Another privilege tuple might grant the same privileges to any user who knows that the password is "WASHINGTON". A third might only grant read access to files under that node to users giving the password "DC". For a full discussion of privilege tuples, please refer to the latest Datalanguage manual.

This external view of the Datacomputer's file system - a tree-structured hierarchy with multiple protection classes enforced on each node in the tree - is dealt with transparently by the Request Handler. This means that the structure seen by, and the functions available to the ultimate Datacomputer user are essentially the same as those provided by Services to the Request Handler.

### 2.4.3 Access to Datacomputer Files

As mentioned above, a Datacomputer file is stored as a number of sections, each of which is broken into 512 word blocks called pages. When the Request Handler wishes to access some page of a Datacomputer file, the following sequence of events must take place:

1) The file is opened. To open a file, RH supplies SV with the string representing the file's pathname in the Datacomputer file system (along with any needed passwords). SV determines that the current user is allowed to access the file in the manner requested (and the file exists), then returns a small integer, known as a Relative File Number or RFN. The RFN is the handle used by RH in all future references to the file until it is closed, at which time the RFN becomes invalid.

2) A buffer is allocated in the user process's address space. Buffers are managed by SV, but their allocation, freeing, and use is under the control of RH. A buffer is exactly the same size as a Datacomputer file page (and of a TENEX page). The buffer is identified by yet another small integer returned by SV.

3) If the page is being read (data already exists and is being referenced), an SVCALL known as PGRD is executed. This takes the RFN of the file, the section number, the page number within the section, and the buffer number into which the page is to be read as inputs. After the call, the page is available in the buffer.

4) If the page is being created, data is first entered into the buffer by the Request Handler, then the page is written to the file by the SVCALL PGWR. Arguments are the same as with PGRD.

5) If the page is being modified, the sequence is PGRD, modify, PGWR.

6) When the Request Handler is through with the buffer and the file, the buffer is released by an explicit SVCALL, and the file is closed.

### 2.4.4 The SV Input/Output System and Monitor

The input/output and monitor facilities provided by Services are fairly simple when compared with the directory system. Input/output consists primarily of a set of connections to the ARPANET, with the ability to read and write

buffers of data to/from a given connection. A special set of SVCALL's are provided for communication with the Datacomputer operator's console. The operator is consulted before particularly large requests are executed for user jobs, and certain kinds of messages about the state of the Datacomputer are routed there.

The Services monitor provides no particular facilities of its own, but is responsible for the creation/destruction of TENEX forks which represent particular Datacomputer sub-jobs. As users contact the Datacomputer via the network, they are assigned to a particular sub-job by the master process known as "Job 0", which is just like any other Datacomputer process, except it has the monitor code enabled.

## 2.5 The User's Level - RH

The "outermost" level of the Datacomputer is known as the Request Handler. RH is in some sense an application program, since it is possible for a reasonably naive user to interact directly with it, via a specialized data-management language known as "Datalanguage". It would not be unreasonable to consider Datalanguage as the Datacomputer's order code.

Datalanguage and the Datacomputer were designed to be used by PROGRAMS running in other hosts. However, since all control interactions with the Datacomputer are expressed as strings of human-parsable ASCII characters, it is possible for a human user sitting at a terminal which is transparently interfaced to the Datacomputer to interact directly and successfully with the Datacomputer. To avoid the anthropomorphization which usually creeps into descriptions of machine-machine interactions, this section is written as if the Datacomputer user were a real human being at a terminal. The reader is cautioned to bear in mind the system is not intended to be used in this mode; that the Datacomputer is a resource for machines and their programs.

### 2.5.1 User-Datacomputer Interactions

The Datacomputer maintains one or more input/output channels for the user. These are called "ports". All Datalanguage interactions flow over a particular port known as the "default port" or the "Datalanguage port". This port is the connection established when the user first contacts the Datacomputer from the ARPANET. Data may flow over the default port or over auxiliary ports which are created by Datalanguage statements as the session progresses. It is preferable to use auxiliary ports for data for two reasons:

first, only ASCII data may pass through the default port; and second, even though the data being passed is ASCII, care must be taken to insure that it contains no characters which are treated specially when passed through the default port.

Datalanguage statements fall into two categories - commands and requests. In general, commands control the state of the user's Datacomputer process; open and close files, create nodes, modify privilege tuples, etc. Requests refer directly to the contents of files. A large part of Datalanguage is devoted to the detailed description of the contents of files, and the Request Handler makes extensive use of such descriptions in planning its actions.

### 2.5.2 Request Handler Structure

When the user first connects to the Datacomputer, Services initializes a new Datacomputer process, then passes control to the Request Handler. RH does some initialization of its own, then asks SV for the next line of input from the Datalanguage port. If the input line is a command, it is executed immediately. Requests are compiled, then executed.

## 2.5.3 The RH Compiler

The Request Handler's compiler is invoked for most requests. (A special subset of easy-to-handle requests are interpreted by a special module known as "SLURP".) The compiler consists of three parts.

1) The first phase of compilation is handled by a routine known as the "pre-compiler". The pre-compiler takes the request as received from the user, does validity/syntax checking, and produces a new representation of the request known as "intermediate language". Intermediate language consists of a set of functions which are an abstract description of the entire set of operations which are legal on Datacomputer data. These functions are essentially the low-level machine language of the Datacomputer. They represent elementary operations such as "move an item from container 1 to container 2" with appropriate ancillary information such as the type and location of containers 1 and 2. Most of the "smartness" of the Request Handler lies in the pre-compiler. It is completely responsible for the syntactic and semantic interpretation of user requests (but not their execution).

2) After the pre-compiler has abstracted and simplified the request, the intermediate language generated and the descriptions of the real files which are named in the request are fed to the rest of the compiler. This section is responsible for generating the instructions for actually moving data from one file (or port) to another under the control of the request. The output of this phase of the compiler is a data structure which contains all the messy loops, skips, and such for plowing through and pulling the data specified in the format requested from the file. The descriptions of these operations are called "tuples".

3) Finally, the routines which actually execute the request on the data are, in some sense, part of the compiler. Many of the tuples have distinct sub-routines which are responsible for their execution, and those routines constitute both the run-time environment and part of the compile-time data base of the compiler. Because of the multitude of data-types, byte sizes, etc. allowed by the Datacomputer, each tuple has many "modes", which are identified by bits in the data structure. For any given request, a particular set of modes is used, and a particular subset of the tuple code is executed. The last phase of the

compiler walks through the tuple list that defines the request, and extracts the instructions which perform the tuple functions as constrained by the active mode bits in the tuples, producing the final "compiled request", which is executed with the real data.

# Chapter 3

# Datacomputer Usage

The dominant fact in usage of the Datacomputer during the latter half of 1976 was the release of Version 2, with its incorporation of the terabit memory system. This fulfilled the Datacomputer's promise of very large, cost-effective on-line storage, and use of the Datacomputer reflected that reality. The number of bits stored has increased dramatically; projects which had filled their available allocations in system development went into production as the TBM's facilities became available.

## 3.1 Seismic Usage

The seismic application of the Datacomputer is discussed in more detail in section 6 of this report; at this point it suffices to sketch the explosive growth in use of the Datacomputer as this application went on line. Beginning in October, real-time raw seismic data was fed to the Datacomputer through the Arpanet at rates of 7 - 12 kilobaud, around the clock. By the end of the year, nearly 70 billion bits of seismic data -- raw readings, event sum-

maries, instrument status reports, and various historical files -- had been stored in the Datacomputer. If stored on conventional disk storage, this volume of data would have required more than 85 spindles of 100-megabyte drives!

As the seismic database grew, researchers began retrievals against it, initially to develop and test procedures for use in ongoing seismic studies. A special program was developed at CCA to assist analysis of seismic data by providing quick graphs of waveforms from the raw data files; and the RDC program developed earlier at CCA was used extensively in these inquiries. The Applied Seismology Group at Lincoln Laboratories also developed several systems for accessing seismic data through its PDP-11 UNIX system, and were active users of seismic data during this period.

## 3.2 DFTP

Growth of user activity in other areas of Datacomputer service was somewhat less dramatic, but equally real. The most widespread usage continued to be the DFTP system, which provides a uniform file archival service for various PDP-10 systems throughout the Arpanet. Several significant developments occurred with DFTP during this period: The new file

organization discussed in the previous semi-annual report was implemented, and most user's data was transferred to that format. Installation of the TBM allowed usage to grow beyond the artificial bounds imposed by disk space available in earlier versions of the Datacomputer. A new implementation effort extended the class of operating systems on which DFTP is available, and thereby its domain of users.

The new format of DFTP storage collects many files for a particular user into a single Datacomputer file. The space savings in the scheme are impressive, but relatively unimportant (averaging around 60% for most users). More significant is the reduction in Datacomputer directory overhead and in TBM accessing which goes along with this compacter storage. The new format also provide users with additional directory information and integrity features, and incorporates several user features which make it considerably more convenient & effective. By year-end, all but three existing sites had had their user's data transferred, and the new version of the program made available to users. The remaining sites were delayed by coordination difficulties, but all were transferred before this report was written.

A number of sites which had expanded to fill their allocation long ago were allowed increased space when the TBM became available. There was also a significant increase in the number of active users of DFTP; as a result, file

storage rose from about 850 to about 1300 megabits in the second half of the year, and was increasing steadily.

Concurrently with the release of the new DFTP to old users, versions of the program were made available to users of the ITS, SAIL, and TOPS-20 operating systems. This provided complete coverage of the PDP-10 family of systems on the network; efforts were underway to spread DFTP to MULTICS and UNIX systems during the period.

In addition to its primary function as an effective file storage resource for members of the Arpanet community, DFTP served as an example system for groups investigating means of accessing the Datacomputer; these included researchers in Facsimile message processing at the University of London, researchers from several ERDA laboratories, and the UCLA-Security research group.

## 3.3 IMP statistics

The Arpanet Network Control Center has been an established user of the Datacomputer, storing statistics on performance of the IMPs which implement the network. This application continued unabated, and artificial limits on the amount of data stored were removed with installation of the

TBM. Usage has grown from 200 to 600 megabits during the period of this report, with steady retrieval activity against the data. The retrieval programs written at BBN incorporate a package of Datacomputer interface subroutines (DCSUBR) described in the predecessor to this report.

## 3.4 SURVEY

Another well-established use of the Datacomputer is the SURVEY application, carried on in conjunction with MIT's Laboratory for Computer Science. Current survey data on the status of hosts on the Arpanet continued to be stored at the rate of about 10,000 probes per day. Efforts also began to restore SURVEY data which had been migrated off-line due to space considerations in old versions of the Datacomputer. By year-end, all four quarters' data for 1976 were restored, raising usage from 350 megabits to about 600.

The SURVEY database was used during this period by researchers in the Very Large Data Base project at MIT. They used the Datacomputer's processing of requests against 2 quarters' worth of SURVEY data to test their work in estimating the cost of query processing in very large databases. This usage involved approximately 300 more megabits of storage.

## 3.5 ACCAT

The ARPA Command and Control Advanced Testbed, being undertaken at the Navy Electronics Center Laboratories in San Diego, became a major focus of interest in the Datacomputer in this period. Initial investigations begun at the Stanford Research Institute in 1975 were continued, with further demonstrations of the system which interfaced a natural language query processor to the Datacomputer. The conversion of the database which supports this system to a relational format was begun, and steps were also undertaken to load a much larger command and control database into a version of the Datacomputer to be brought up at NELC. Datacomputer personnel provided consultation and programming support for this conversion and loading task; previously implemented user software (like the DCSUBR package) proved valuable in this effort. Major research in distributed data management will be carried out under a separate contract, using the Datacomputer as the basic DBMS.

## 3.6 ERDA

Several of ERDA's national laboratories began investigations of the Datacomputer during the reporting period; most active were groups at the Argonne and Lawrence Berkeley Laboratories. The major project in this period was beginning installation of a climatological database by personnel at Argonne. This database contains 16 files, each with a year's worth of hourly readings for some U.S. city; the data are used by a number of sets of programs which model energy usage in buildings and communities (CAL-ERDA, ATMES, and ACUC systems).

Interfacing with Argonne personnel provided a test of the scope of Datacomputer applicability beyond its previous extent: the database was being transferred from a 370/195 at Argonne, through a Varian network interface, into the Datacomputer, and later retrieved for processing at Berkeley's CDC 6600. Despite problems with the network interface at Argonne, this sequence of transfers was successfully accomplished with minimal user programming effort and no modifications to the Datacomputer. It is interesting that the standard File Transfer Protocol (FTP) programs implemented at Argonne and Berkeley proved sufficient for accessing the Datacomputer, although they might be replaced by

more specialized interface programs at some later date. The concept of the Datacomputer as a resource for data sharing thus received additional validation.

## 3.7 Message Archiving

Under a separate, short term contract, CCA began investigations into the use of the Datacomputer for archiving network messages. Preliminary investigations dealt with interfacing to the various message handling programs now available on the Arpanet, and designing possible Datacomputer implementations which would be consistent with such systems. Experience gained in implementation and operation of DFTP proved valuable in this effort.

## 3.8 NSW

The National Software Works project began work on using the Datacomputer as the archival system for its historical databases as storage became available for that purpose. Initial design relied on existing Datacomputer user support facilities, such as the DCSUBR package mentioned above. At

the end of the reporting period, consultations were underway between Datacomputer and NSW personnel on specifics of file description and accessing characteristics.

## 3.9 Accounting

As a final task in the implementation of the Datacomputer, a system was implemented for accounting for usage. There were several interesting problems involved here: It was necessary to choose a set of parameters which at once constituted an adequate measure of resource utilization and could be made accessible to the accounting system with reasonable effort. Given this information, the actual processing of that usage data into meaningful accounts was a relatively straightforward reporting task. There were certain kinds of reporting, however, which were unique to the Datacomputer's status as a network utility. These involve access to shared databases. Questions arise as to who is ultimately responsible for the storage and accessing of shared data; if charging for these functions is separated, the "owner" of a database may lose information on how it is being used, and by whom.

## Datacomputer Usage

Collection of appropriate usage statistics proved simple, as the Datacomputer already supplied more than enough statistics in its normal processing; modifications to the Datacomputer involved simply formatting the appropriate counters in a separate file, and adding routines to account for storage and directory space used. This information is dumped to a file external to the Datacomputer; from there several straightforward programs manipulate it to produce usage reports which can serve as the basis for billing when needed. Shared databases have their storage and accessing charges separated; however, advisory access reports are supplied to the official "owners" of data detailing times, modes and user identities for access to that data.

Preliminary reports were produced at year-end, and regular reporting is expected to begin monthly in 1977.

# Chapter 4

# Software Development

Description of work on Datacomputer software will follow the division between the Services (SV) and Request Handler (RH) sections of the project introduced in Chapter 2.

## 4.1 Services

The Services work during this reporting period was concentrated in five areas. Staging routines, terabit memory (TBM) support, accounting, directory security and maintenance/testing.

### 4.1.1 Staging

The design and implementation of the staging area replacement algorithms was accomplished during the first part of the reporting period. As the staging area or its directory become full these algorithms are used to determine which files should be removed to free additional space.

File selection is based on the amount of user generated activity and the number of modified pages that must be copied back to TBM before the file can be deleted. Three algorithms were implemented for the Version 2 release. The primary algorithm involved the Datacomputer monitor (Job0) periodically running a background process to monitor the staging area and its associated directory table. When space used exceeded a certain threshold the background process would begin to migrate updated files back to TBM freeing up additional space. In the event the background process was unable to keep up with user demands for staging space, two overflow algorithms were implemented to enable user subjobs to free their own space.

From observing the operation of the Version 2 Datacomputer, it was discovered that Job0 was not always able to run the staging area monitor process often enough to prevent staging area overflows. This problem was solved in the Version 3 Datacomputer by moving the staging area monitor to a fork inferior to Job0. The staging routines were also enhanced to support multiple staging disks. This gave the Version 3 Datacomputer a staging area capacity of up to 120,000 pages. Several new operator commands were implemented to facilitate operator control of the staging area. These included staging area status and variouscommands to enable staging disks to be dynamically added or deleted from

the Datacomputer. For users with applications requiring fast access to files, a capability was added to allow files to be "frozen" in the staging area. A "frozen" file can remain staged indefinitely, thus avoiding the overhead that might be caused by re-staging.

### 4.1.2 Terabit Memory (TBM) Support

One utility was modified and several new utilities written to support TBM operation. The Directory Cross Checker (XD) is a utility that verifies the internal consistency of the Datacomputer Directory. XD was modified to support off-line TBM Volumes and to build the enormous bit patterns required to verify that no two Datacomputer files are inadvertently mapped to the same TBM block.

Because the blocks on TBM tapes eventually wear out, two utilities were written to create backup copies of Datacomputer files. The first utility is automatically run by the Datacomputer every night. It walks the Datacomputer's directory tree and creates backup copies of recently created/updated files. The storage maps for all copies of a file are chained together on the file's directory page. In the event that a TBM tape is accidentally damaged, the storage map pointers for all effected files can be simply

changed to point to their previous file copies, resulting in little or no loss of data.

Since space on TBM tapes is never physically deleted, a utility was written to copy the latest version of all undeleted files from one tape to another. The old tape can then be archived or reused.

A new version of the Datacomputer software will frequently contain directory modifications that make it incompatible with older versions. The normal procedure for upgrading a database for a new software release was to dump all data and directory information and reload everything with the new version. With the large amounts of data stored on TBM this is no longer possible. A utility was implemented to dump and reload only the directory information. Lastly, TBM tapes are now mountable/demountable per operator command. This permits very large or seldom used files, or full TBM tapes, to be stored off-line and mounted only when needed by specific users.

### 4.1.3 Accounting

The accounting package provides a facility for monitoring costs incurred by individual users accessing the Datacomputer. Services support for this effort included generating a history file of dynamic charges (i.e., pages read/written, cpu time) incurred during user sessions and implementing a utility routine. This utility is run at the end of a billing period. It walks the Datacomputer's directory tree and records static charges (i.e., file space, directory space) associated with each node in the directory. A facility to allow the database administrator to mark certain nodes as billable and to insure that all users are loged in beneath a billable node was also implemented.

### 4.1.4 Directory Security

A general mechanism to allow deferred privileges was implemented. A deferred privilege is one that is granted or denied at one level in the directory tree but does not take effect until the directory tree but does not take effect until the next deeper level. The immediate result of this was to prevent users from modifying their own space allocation limits.

### 4.1.5 Maintenance / Testing

During this reporting period at least two and occasionally three Datacomputers were being actively supported. The Version 1 Datacomputer was operational throughout the reporting period. It is currently being phased out as the last of its database is transferred to the Version 3 Datacomputer. A preliminary version of the TBM Datacomputer was available for limited network access prior to the Version 2 release. The Version 2 Datacomputer became operational in October 1976.

A great deal of time was spent testing Services code for both the Version 2 and Version 3 releases. In addition, the first month of the reporting period was spent testing and debugging the Datacomputer/TBM interface.

## 4.2 The Request Handler

During the second half of the year, the majority of the work involving the request handler section of the Datacomputer system fell into two areas: software development and system maintenance.

### 4.2.1 Datacomputer Version 2

Datacomputer Version 2 was released mid-way through the reporting period. Although some software development took place prior to the release of Version 2, the majority of the (Version 2) effort was directed toward testing and system integration.

#### 4.2.1.1 Space Allocation

Users have been given the option of exercising more detailed control over the allocation of space for their files. The system continues to supply default allocation parameters that will suffice for most applications. However, with the advent of the mass memory, the variation in possible file sizes has passed the point where default allocation parameters will serve all users reasonably.

Allocation of space in the Datacomputer is done in terms of physical blocks on the mass memory, each of which holds 1,032,192 bits. This block is the basic allocation unit, and its size is referred to as an "M", to reflect its approximation to a megabit. It is also used when the user specifies an allocation limit, as the ",M=" option of the

CREATE and MODIFY commands. Output from a LIST command (%DIRECTORY or %INFORMATION options) describes space allocated, used, or charged, and is now given in Ms correct to 2 decimal places.

Several options have been added to give users more control over various aspects of the space allocation for their files. All of these new options have default values which are compatible with the internal values used by the Version 1 Datacomputer; thus applications that were designed for the Version 1 system may use default values for the allocation options in the Version 2 (and 3) Datacomputer. Most new Datacomputer applications may use default values for these parameters.

If a file contains variable length containers which may be updated, it must have the CHAPTER option specified on it. This causes the base to be broken down into smaller pieces, called chapters, and thereby eases the problem of expanding or shrinking records in the middle of a file. It also causes space to be left unused in each chapter to allow for possible expansion.

Two parameters are provided to allow the user to specify the manner in which the file is chaptered: CF (Chapter Fill level) and CR (Chapter Record count). Both options specify an integer value. CR is the number of

records to be included in each chapter and CF is the percentage of a chapter which is to be filled with data when it is first written.

The default value for CF is 80; for CR, it is the number of records that will fit in a 1 M block, given the average record size calculated from the file description, and the (default or explicit) value of CF.

The %DESCRIPTION option of the LIST command will show these values for a chaptered file as specified or defaulted.

Three new parameters were implemented to control the space allocated for inversion: IA (unique number of attribute value pairs), ID (inversion density) and II (inversion increment). The Datacomputer Version 2 User Guide presents a comprehensive explanation on how and when to use these options. The combination of default values for these parameters results in an inversion allocation of 1 M and should suffice for most applications.

The %DESCRIPTION option of the LIST command will list the values of IA and ID only if they have been specified by the user; it will list II whether it was explicitly set by the user or left as a default.

#### 4.2.1.2 DIRECT Mode

A new mode option, DIRECT, has been added to the OPEN and MODE commands. There are now two inversion-related mode options for these commands: DIRECT and DEFER. Mode options pertain to the internal manner in which the addition of records to inverted files is handled by the Datacomputer. DEFER is appropriate in almost all cases and is now the default. DIRECT causes the system to update the inversion data structure for every inverted container's value as it is written to the base and should be specified only when the number of records being added to the file is exceedingly small. DEFER causes the system to batch a group of inverted values and update the inversion structure 'occasionally' during the request. In most cases, this is much more efficient.

### 4.2.2 Datacomputer Version 3

In the second half of the reporting period, development efforts were concentrated on the implementation of new features for the Version 3 Datacomputer.

4.2.2.1 File Groups

Under support from a related contract from the ARPA Nuclear Monitoring Research Office, the File Group feature was implemented and made available to all Datacomputer users with the release of Version 3.

Modifications were made to all levels of the compiler and the command handler to support this feature.

The CREATE command was expanded to support the creation of a group node. A group node itself is a special kind of file which defines the members of the group. Two options were added to the description mechanism for files. The logical constraint option (,LC=<boolean>) allows the system to determine which subfiles of a group should be accessed during the execution of a request. The automatic include option (,GROUP=<group-pathname>) causes the file to be included as a subfile of the named group at creation time.

The user must take the initiative in defining and maintaining a group's domain using three new commands. INCLUDE causes the system to include a file as a subfile in a group. At INCLUDE time, the user may also specify a logical constraint. EXCLUDE causes a subfile to be marked as deleted from the domain of a group. COMPRESS causes the system to

garbage collect the group's domain, removing all subfiles marked as deleted.

In support of these three new commands, a general mechanism for internal requests was implemented. An internal request is a request initiated by the Datacomputer itself rather than a user. INCLUDE results in an internal request to append a record to the group's list of members describing the newly included subfile. EXCLUDE results in an internal request to update this list, marking the subfile as deleted. COMPRESS results in the list being rewritten, skipping all records marked as deleted.

A new option, %DOMAIN, was added to the LIST command. This option causes the system to read the list of the files in the group (its "domain"), and output the names and logical constraints of all non-excluded subfiles.

The pre-compiler, compiler and code generator were modified to handle requests run on groups. Essentially, a loop is generated to read the group's domain. For each non-excluded subfile in the group, an analysis is performed on the subfile's logical constraint and the request qualification to determine if the subfile must be accessed. If it is determined that the subfile should not be accessed, the 'inner-loop' which would process the subfile is skipped and the next subfile is considered.

4.2.2.2 Functions

Four special arithmetic functions were added to Datalanguage and made available to all Datacomputer users with the release of Version 3. Each of these functions may be used semantically as an expression or on the left hand side of a relation.

GCDIST returns the great circle distance from position A to position B accurate to the nearest nautical mile. BEARING returns the great circle bearing from position A to position B accurate to the nearest degree. RLDIST returns the rhumb line distance from position A to position B accurate to the nearest nautical mile. COURSE returns the rhumb line course from position A to position B accurate to the nearest nautical mile.

4.2.2.3 Priority

A comprehensive priority mechanism is planned for a future version of the Datacomputer. The Datalanguage to support this mechanism was implemented during the reporting period.

Each node will have a priority limit associated with it. A system default priority level (queue position) will be established. All requests will be run at this level unless overridden.

A new option was added to the CREATE (node) and MODIFY commands (,P=<integer>). Its function is to set the priority limit at a node, similar to the allocation limit option.

The PRIORITY command was implemented. Its function is to override the system default priority level for a session, up to a user's priority limit.

A new privilege option was added to the CREATEP command (,Q=<integer>). Its function is to override the system default priority level when matched, without the need for issuing a PRIORITY command.

### 4.2.3 Maintenance and Testing

Due to the fact that two versions of the Datacomputer were released during this reporting period, a large amount of time was spent on testing. Our catalogue of test decks was expanded to insure comprehensive testing of old and new features. All of the test decks were run twice, once in August prior to the release of Version 2 and again in December prior to the release of Version 3.

# Chapter 5

## Hardware / Site / Operations

During the first half of 1976, an Ampex Tera-Bit Memory system (TBM) was installed at CCA. After extensive tests during the second half of 1976, the TBM was accepted.

### 5.1 Site Improvements

Major site improvements were made to accommodate the TBM near the end of 1975. An additional air conditioning unit became operational in January 1976 and in February the TBM hardware was delivered.

### 5.2 The TBM

The TBM system at CCA is a 200 billion bit configuration with 50 billion bits per drive and four drives. The following figure shows the general structure. The TBM's transfer rate is 6 million bits per second, similar to an IBM 3330 type disk. A high speed seek from one end of a tape to the other takes approximately 45 seconds.

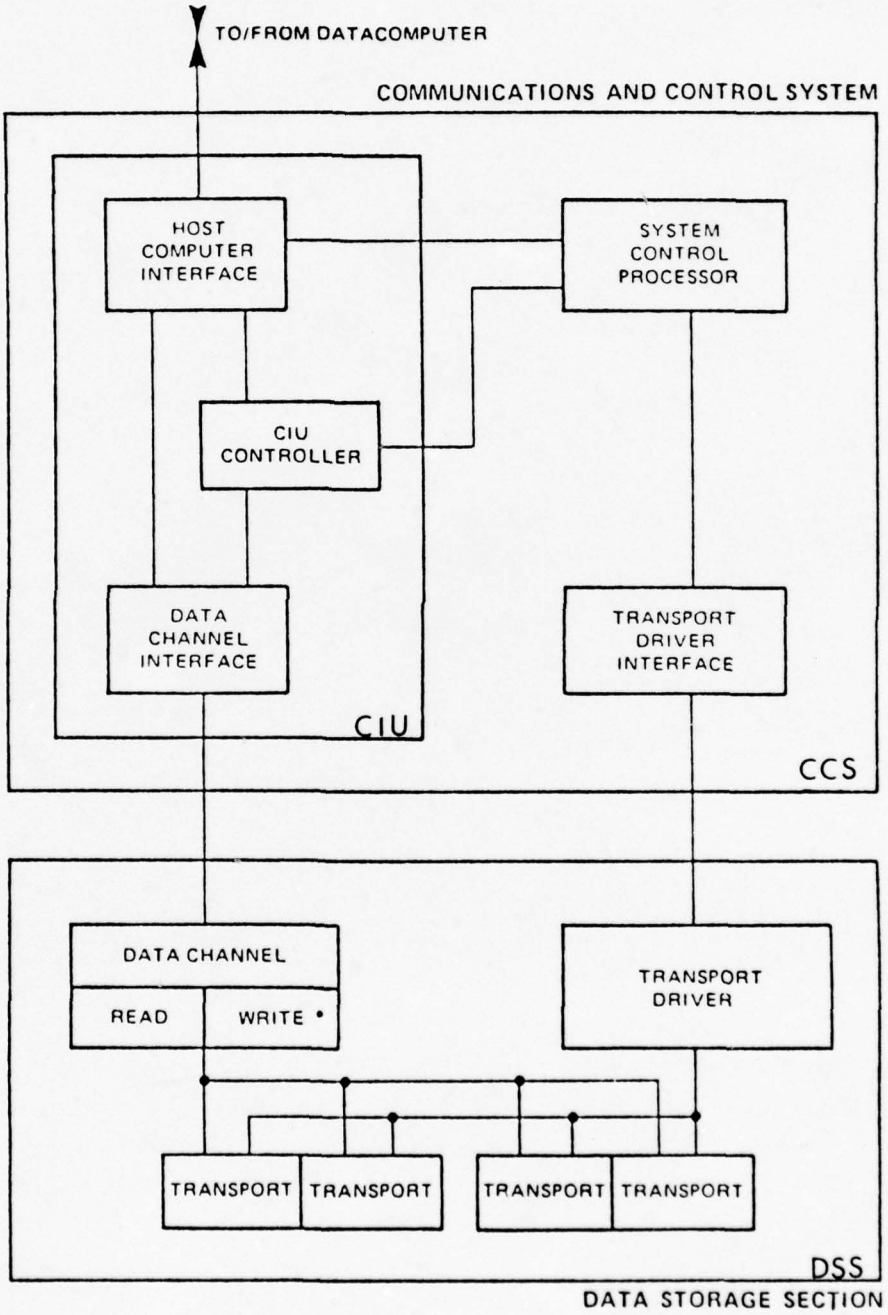


Figure 5. Datacomputer TBM Configuration

By March 1976 data had been transfered to and from the Ampex TBM but Ampex continued to work on ironing out difficulties in the TBM at CCA through the end of June. Agreement on a mutually acceptable detailed test plan was reached in early June and formal acceptance testing started in July. The hardware performed well during this test but some difficulties were encountered with the internal TBM software particularly in the CIU (Channel Interface Unit, a PDP-11/05) and the SCP (System Control Processor, a PDP-11/35). A further software test in August demonstrated that these problems had been corrected and the system was accepted.

## 5.3 Operations

The Version 1 Datacomputer as described in its User Manual was operational using three 3330 type disk spindles for storage and provided service over the Arpanet at the start of this reporting period. When the Version 2 Datacomputer was made publicly available, using the TBM, some information was transfered into it from the operational Version 1 Datacomputer and the Version 1 Datacomputer was squeezed down to two 3330 disk spindles to provide staging room for Version 2.

The Version 2 Datacomputer was provided experimentally starting in June 1976 using a 2314 type disk as staging. Starting in October, 1976, Version 2 using 3330 type disks for staging and TBM tape for storage became the standard Datacomputer.

At the end of 1976, the Version 3 Datacomputer, with the enhancements described above, became available over the network.

# Chapter 6

# Seismic Data Base Support

As mentioned in the Introduction, some work on the Datacomputer is funded under a separate contract from the Nuclear Monitoring Research Office (NMRO) of ARPA. A short discussion of this work is included here because of its intimate relation to the work of the primary Datacomputer development contract.

## 6.1 Overview

This work is directed toward establishing an on-line, real-time data base of seismic information from a world-wide network of monitoring sites and toward making this data available to computers for seismic analysis and other purposes.

Since the system will work in real time and some of the data will be retrieved by computers on the Arpanet, the Arpanet was chosen as the most appropriate communications medium available for the entire system. Seismic data is collected at the Seismic Data Analysis Center from sensors at various locations around the world. The data is then

transmitted from the SDAC Control and Communications Processor to CCA over the network. At CCA a small highly reliable computer known as the Seismic Input Processor, or SIP, absorbs the incoming data and stores it in a disk buffer. Periodically, the SIP connects to the Datacomputer (again via the network) and bursts the collected data into the Datacomputer at a very high rate. A limited amount of data from sources that do not report in real time will also be sent to the Datacomputer directly.

## 6.2 The SIP and Network Considerations

The SIP became operational during the first half of 1976. In the later part of 1976, NORSAR and LASA data were being successfully processed through the SIP.

Observation of network behavior while real time seismic data is being sent from SDAC to the SIP and older seismic data is being burst from the SIP to the Datacomputer leads to the conclusion that the current network capacity at CCA is marginal for this application. The difficulty is due to limited processor power and very limited reassembly buffers in the 516 IMP. Some relief was gained last year when the 316 TIP at CCA was replaced with a 516 IMP to increase the

bandwidth of CCA's network node. Temporary relief was also provided by moving the Lincoln Laboratories VDH that had terminated at CCA to another IMP. However considering the much higher loads planned in the future, the most appropriate long run solution to this problem appears to be the installation of a properly configured PLURIBUS IMP at CCA. Action has been taken to accomplish this and installation is expected to take place in 1977.

# Chapter 7

## Summary

Over the course of this contract the Datacomputer has grown from a very basic system into a sophisticated large scale data storage facility with substantial data management capabilities. At the beginning of this contract the Datacomputer supported only a limited amount of storage and a restricted set of user functions. Data description and manipulation facilities were limited. Today the system provides facilities for data sharing among dissimilar machines on the Arpanet, rapid access to large on-line files, storage economy through shared use of tertiary store and improved access control.

Internally, all Datacomputer modules fall into one of two principle subsystems, which are called "Services" and "Request Handler". The Request Handler is the "outer" layer of the Datacomputer. It provides the interface to the Datacomputer user, parses and compiles Datalanguage, formats and interprets user data, and is in general control of overall Datacomputer functions. Services modules provide the lower-level functions internal to the Datacomputer itself, such as device and network I/O control, core and disk buffer management, file directory services, and operating system interface.

## 7.1 Services

When this contract began, the Services subsystem offered rudimentary capabilities. Many of the then-existing modules have been extensively reworked, and several major modules have been added. Major changes or additions have been made in the areas of tertiary memory I/O functions, testing and support utilities, file security and accounting. Each of these facilities is described in more detail below.

### 7.1.1 Tertiary Memory Support

The most important addition to Services is the support system for the tertiary memory. This has required implementation of the data-staging mechanism known as SDAX, a generalized data-page moving mechanism ("slosher") capable of moving large amounts of data between TBM's, 3330-type disks, RP02-type disks and core. SDAX reduces I/O wait times for users who have data stored on tertiary memory by staging (portions of) file data onto dedicated 3330-type disks. The user's file data operations are then performed on the staged

data and at an appropriate time modified data is written back to tertiary memory or simply deleted from disk. These operations are completely transparent to the Datacomputer user. Implementation of SDAX and slosher routines required extensive modifications throughout many other Services modules, including the directory tree, error and core-management handlers.

### 7.1.2 Utility Support Programs

In addition and as an enhancement to tertiary memory device support, a number of utility functions were added to the Datacomputer. The most important of these are:

- The directory cross-checker, which exhaustively examines the internal consistency of the file directory;

- Backup and reload utilities for Datacomputer files, the file directory and TBM tapes (each of which contains the equivalent of approximately 6,000 standard magtapes worth of data);

- Extensive testing facilities, including a "Q-tester" which simulates the activity of several concurrent network Datacomputer users in creating and deleting nodes and files, and reading and writing patterned data to these files;

- Greatly improved error-handling facilities. The Datacomputer is able to recover and continue from a very large number of potential errors, especially including device and network I/O problems.

### 7.1.3 Security and Accounting

Two other major areas of work have been the addition of a sophisticated security subsystem and inclusion of an accounting package. Datacomputer security is provided by a privilege block and password system which operates at every level of the directory tree, is user-settable for all nodes under that user's control, and which offers per-user accessability to any given file. The accounting package offers a facility for prorating costs of Datacomputer operation, data storage and network interface to individual sites accessing the Datacomputer.

## 7.2 Request Handler

Key results of the Request Handler development effort have fallen into the following categories: data description, data manipulation, and efficiency enhancing mechanisms. The following subsections will consider each of these issues in turn.

### 7.2.1 Data Description

Data is stored by the Datacomputer in FILEs whose contents are described in a directory maintained by the system. Data is transmitted to or from the Datacomputer through PORTs which are also described in the system directory. The Datacomputer's role as a central point for data in a heterogeneous network gives it the rather unusual requirement of being able to deal with a large variety of data types and machine representations of data. The data description facilities were greatly expanded to handle strings whose character set is EBCDIC or BCD as well as ASCII; one's complement, two's complement and unsigned binary integers; and non-binary integers (signed as well as unsigned octal, decimal and hexadecimal).

## Summary

A full set of description options was implemented. The byte size option specifies the number of bits in each byte of the container. The fill character option specifies which value the system should use to fill the container if no data is assigned to it or to pad data which does not fill the container to its minimum size. A terminator tells the Datacomputer where to find the end of data in a variable-length container. There are three terminator options: count, delimiter and punctuation. Virtual data is data that is not stored within the file, but is system maintained and accessible by the user. No space is allocated for a container having a virtual option specified; the system supplies a value which may be retrieved and used like the value of a 'normal' container in assignments and expressions. The Datacomputer supports two virtual options, virtual expressions and virtual indices. A virtual expression may contain any arithmetic or string operators on constants or other containers within the same file. When referenced the system calculates the value of the expression. A virtual index supplies the position of the closest list (i.e., the record number at the outermost level) and may be used in any expression.

7.2.2 Data Manipulation

From the user's point of view the Datacomputer is a remotely-located utility. It would be impractical to use such a utility if, whenever the user wanted to access or change any portion of his file, the entire file had to be transmitted to him. Accordingly, Datalanguage has been expanded to be self contained. The user sends a 'request', which causes the proper functions to be executed at the Datacomputer without requiring entire files to be shipped back and forth. Requests are composed of one or more statements which transfer data (assignment statement); group statements so that two or more statements are treated as a single statement (BEGIN-END block); declare local variables (DECLARE statement); selectively execute statements (the IF-THEN-ELSE statement and the UNTIL loop); selectively receive, transfer, and transmit data (FOR and APPEND loops); update containers (UPDATE loop); delete records (REMOVE statement); and print messages (COMMENT, ERROR, ALERT, ABORT, QUIT statements).

The first implementation of the UPDATE loop restricted the containers which could be changed to fixed-length ones. Variable-length containers are typically the best choice in terms of storage efficiency and ease of data handling where

strings of characters (such as names and addresses) are involved. With these considerations in mind, the CHAPTERed file mechanism was implemented. Chaptering is a technique which divides a file into parts, called chapters, to facilitate the insertion and deletion of data. Each chapter can be independently expanded or contracted to accommodate shrinking or growing records.

In order to provide a full set of computational capabilities, arbitrarily complex arithmetic, Boolean and string expressions are handled by the Datacomputer. In addition to the representation of different data types a full set of conversions from one type to another must be available so that assignment, arithmetic and comparison operations across data types are possible.

### 7.2.3 Efficiency

Although often not as visible to the user as Datalanguage, efficiency considerations during the manipulation of data are an important aspect of datamanagement systems, especially when dealing with large volumes of data.

## Summary

For very large files, it is desirable to break up the data into physically smaller units so that each subfile or component of the group can be individually accessed, checkpointed, dumped, validated, etc. File groups were implemented to give the user control over how to break up the data into physically smaller more manageable units, those that are online or offline, etc. The user may also specify a logical constraint on each member of a group. 'Logical constraint' is a Boolean which limits the records in one subfile to having specific values in specified container(s) of the file. The system responds to each reference to a group's name by taking the appropriate action to handle a set of subfiles. The logical constraint is a very important efficiency consideration. When looping on a group without a qualification (WITH <Boolean>) each subfile would have to be opened and operated on. With an appropriate qualification (one based on the logical constraint), the system need only reference those subfiles selected. An analysis of the subfile's logical constraint and the request qualification is performed. If the analysis determines that no record of the subfile would be selected by the request qualification, the subfile is skipped. Otherwise the subfile is accepted for processing.

## Summary

An inversion is a secondary data structure that the Datacomputer can use to improve its efficiency in retrieving data by content from a file. Specifically, an entry in the inversion is constructed for every container with the inversion option. For each data value which occurs for the container, the inversion contains pointers to all the outermost list members for which that container has that value. When an inverted file is created, storage space is allocated for the secondary data structure. Although fixed-length inverted strings were previously supported by the Datacomputer, significant improvements have been made. Indirect inversion was implemented, permitting the retrieval of outermost list members based on the values of inverted containers in any inner list. All elementary containers, including variable-length strings, may be inverted. An inversion is not only automatically constructed by the Datacomputer when the file is loaded, it is also automatically maintained when data is appended or updated.

Complex Boolean expressions, those involving several comparisons, fall into three classes: those with all comparisons evaluate from the inversion, those containing no comparisons evaluable from the inversion, and those which mix the two kinds of comparisons. The first two classes pose no problems; the Datacomputer will use the inversion to evaluate expressions in the first category, and not for ex-

pressions in the second category. For those Booleans in the third category, the Datacomputer decomposes the expression into two Booleans. The first, based on inverted comparisons, will be used to retrieve the data. The second, know as 'direct search' and based on the non-inverted comparisons, will be applied to only that data selected by the inverted Boolean.

The container address table (CAT) is used for fast retrieval of variable-length list members. It is automatically created for variable-length list members which contain at least one inverted container and for chaptered files, but may be specified as an option on the description of a file. The CAT provides quick access to list elements and is primarily a tool for increasing efficiency at run-time. For example, to obtain the nth element of a list of variable-length, delimited strings with no CAT would require reading through the first n-1 elements, searching for delimiters. If the same list had a CAT, obtaining the nth element would require only loading a pointer from the nth CAT slot.

Virtual indices are treated as a pseudo inversion, as if it were an inverted container having the list member's record number as data, although no auxiliary data structure is maintained.