Chapter 7:

THE INDEXER

INTRODUCTION

Indexer is the system's file directory: it manipulates files and is, itself, easy to manipulate. Version 1.18 Index consists of a simple database of extended directory records and three specialized pointers.

INDEX.CHN is loaded onto bank #3 and is resident throughout Valdocs operation. INDEX.CHN provides system hooks for either TPM or CPM files. Filenames may have up to 16 words and extensions, unlike CPM's 11-character specification.

INDEX STRUCTURE

The database is a single file, called INDXDATA.NDX, that contains extended directory records of valuable document files (also called val files or data files). Three index or pointer files point to the data file. These three index files, INDEXALPH.NDX, INDEXDATE.NDX and INDXCROS.NDX, index according to alphanumeric, date, and cross-reference, respectively. INDXCROS.NDS cross-references alphanumerically (e.g., "Moms apple pies" will appear before "Xerxes apple pies").

Sequencing is determined by whichever pointer file is in memory. When records are added or deleted, corresponding changes are made in the pointer files.

FILE STRUCTURE

INDXDATA.NDX Data File Header:

2 bytes = the next free (deleted) record. Simply a record number between 0 and EFFF. FFFF means no deleted records, in which case it looks for the next four bytes, which is the available record number.
2 bytes = next available record number.
4 bytes = Julian Date, which is the date the file was last used.
2 bytes = next sequence, the next sequential number. The number of the file created that day. It resets when the day changes.

Data File Record Header:

1 byte = deleted record flag.
    * = deleted record.
FF = not deleted.
4 bytes = a pointer to the next free record if there is an asterick in the first byte. Between 0 and EFFF.
FFFF = the last deleted record.

Data File Record Format:

111 bytes = index entries by keywords; 16 keywords maximum.
13 bytes= CPM file name to which index entry refers.
    This file name has its own format of:
    2 bytes = year
    1 byte = month
    2 bytes = day
    3 bytes = sequence number
    1 byte = decimal point
    3 bytes = sequence number
    1 byte = backup flag, not used

Index File Headers

Each of the three index files has the same type of header, which merely counts the number of index entries. INDEXALPH.NDX, INDEXDATE.NDX and INDXCROS.NDX index by alphanumeric, date, and cross-reference, respectively. The first two files index their entries with a 2-byte record pointer, while the third file uses 3 bytes. The last byte places the keywords in alphanumeric order (e.g., "Mom's apple pies" will appear before "Xerxes' apple pies").

Header:
2 bytes = number of index entries.

Index Entry Format:
2 bytes = record pointer to data file.

Cross-index Entry Format:
2 bytes = record pointer.
1 byte = alphanumeric keyword number.

INDEX OPERATIONS

Control over INDEX operations is determined by the command line that is passed during the chain operation (see TPM section for a description of the chain operation). The command line is at 80Hex in memory. All operations require user input, but there are provisions in the design that allow other programs to make direct use of INDEX operations.

The command line can contain several parameters. While

the order in which parameters are supplied is not strict, it is best to supply them in the order given. All parameters must be described in uppercase and separated by at least one space.

OPERATION COMMANDS

There are four operation commands: INDX, STOR, RTRV, and DISP.

INDX causes INDEX to bring up a display of the most recent files, and a menu permitting multiple operations on the index database.

STOR passes a file name to INDEX and requests that it be stored for the user.

RTRV requests that a file or list of files be retrieved for the calling program.

DISP permits display of the index database for use by such programs as Copydisk and Menu.

PRIMARY FILE NAME <+F>

A primary file name is passed to INDEX by the caller and is returned by INDEX on a retrieve or UNDO from STOR. The parameter appears in the form '+F=' immediately followed by a file name (i.e., no intervening spaces). For example:

```
RTRV +F=filename.fil
```

The file name can contain a disk name, or, both a name and extension, but it may *not* contain a user number (that number which divides a disk into sets or groups--for example, on disk B the user number would be B10 or B3, etc.).

The file name can also specify multiple extensions by means of a list of extensions, separated by commas and enclosed in angle brackets ("<" , ">") following the period (".") in the file name. Currently, the list is used only on a STOR operation. The file name with each extension will be renamed, for example:

```
STOR +F=filename.<val,tmp,4th>
```

ORIGINAL FILE NAME <+O>

In those instances where a file has a previous version, the original name is passed to STOR following a '+O=' (letter O) prefix. For instance, STOR +O=original.name.

If the index reference does not change, the original

file will be deleted; if the reference does change, the original version will be retained. The file will not be deleted if the drives specified by '+F=' and '+O=' are different, as INDEX assumes that this indicates the result of Copydisk. Thus, copying between users with Copydisk can cause problems.

NEW NAME RETURNED <+N>

If a STOR operation is successful, a '+N=' followed by the new name assigned to the file is returned to the caller (i.e., +N=newfile.name).

If the user does an UNDO out of STOR, INDEX returns a '+F=' followed by the incoming file name, just as if this were a return from RTRV (+F=incoming.filename).

CALLER RETURN INFORMATION <\>

The name of the caller and any initial arguments the caller expects upon return from INDEX are specified following a '\' on the command line. In other words, INDEX must know who was calling it in order to be able to return to it--otherwise, INDEX will default to the Editor. For example, to return the INDEX to MAIL:

```
STOR +F=filename,val \MAIL
```

NO LIST FLAG <-L>

A '-L' indicates that the caller cannot handle a list of files on retrieve, and restricts INDEX to returning a single name. While this parameter is recognized, it is not acted upon since list building features have not yet been implemented.

PARAMETER COMPATABILITY

The following chart is provided for reference purposes:

OPERATION	-L	+F=	+O=	+N=	\
INDX	Y	N*	N	N	Y
STOR	N	Y	Y	Y	Y
RTRV	Y	Y*	N	N	Y
DISP	N	Y*	N	N	Y

* Supplies drive name only. Will support extension lists in later versions.

Again, the command line may have several parameters,

best supplied in the order given. All parameters must be described in uppercase, and separated by at least one space.

CONCLUSION

By use of the provided parameter chart and file structures, the programmer should find Indexer easy to use. Any error messages encountered can be found in the TPM section.