Networking with dBASE IV

Networking with dBASE IV

9/88

Contents

Network User

Network Programmer

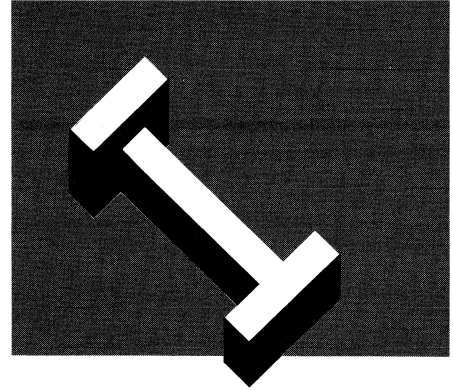
Network Administrator

Error Messages

Glossary

Index

Introduction



Networking with dBASE IV describes how to use dBASE IV™ in a Local Area Network (LAN) environment. Networking allows two or more microcomputers to share valuable hardware, software, and data.

Local Area Networks

The microcomputers and peripherals in a local area network are connected by cables, which provide a common communication resource. Each microcomputer on a network can use shared peripheral devices, such as large-capacity disk drives, tape storage units, printers, plotters, and scanners. Network microcomputers can also share programs and data.

Figure i-1 shows one way microcomputers and peripherals can be connected in a LAN environment.

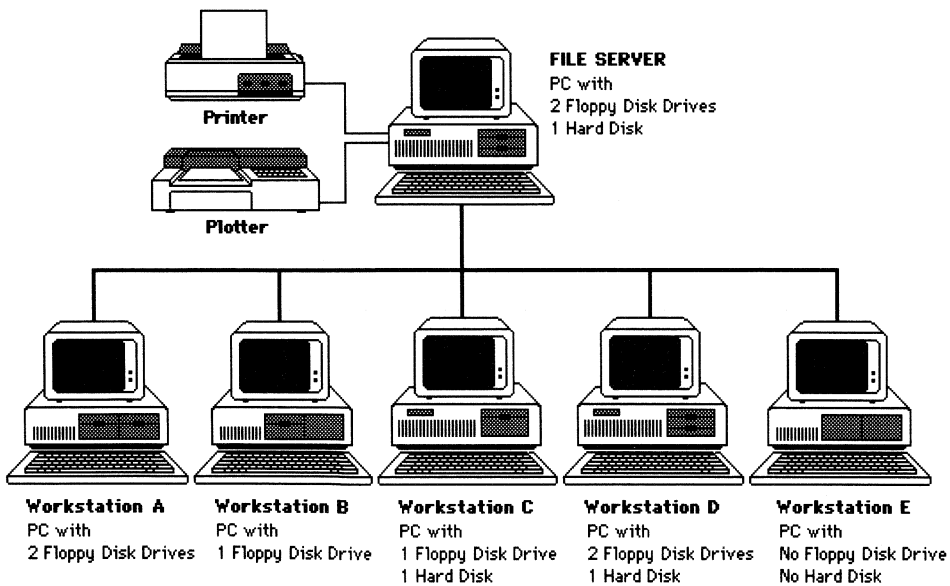


Figure i-1 The local area network (LAN) environment

In a network, at least one microcomputer is designated as the *file server*. The file server manages communication and the sharing of peripheral devices.

The microcomputers managed by a file server are called *workstations*. Workstations are also referred to as *network nodes*.

In a LAN environment, users with different levels of expertise and need can work with dBASE IV. Some will only need to enter data or use application programs. Others will program in dBASE IV, creating database files and application programs. Others will have the reponsibility of administrating or maintaining the network. This manual provides different information and reading requirements for each level of expertise.

Sharing Resources

The purpose of a local area network is to share resources such as the following:

- dBASE IV and other software products, which are set up as shared applications
- Centralized database and index files, which eliminate redundant data, transferring data, and the need for multiple updates
- Peripheral devices, such as disk drives, tape storage units, printers, plotters, and scanners

In a local area network, a shared application program is loaded into the workstation from the file server. Data files can be on the shared file server disk or on a non-shared workstation, but the program is executed at the workstation. If the program needs data from the file server, the system transmits the data over the network. Although resources are shared, your task executes on your workstation. Because other users run tasks on their own workstations, their work does not slow yours.

About This Manual

Before using this manual, your network should be installed according to the instructions in the *Network Installation* manual.

This manual is written for three types of network personnel: users, programmers, and administrators. A section is provided for each type. The Appendix lists error messages specific to the network environment, while the Glossary contains a list of networking terms. For general dBASE IV terminology, refer to the Glossary in *Learning dBASE IV*. This manual ends with a thorough index, to help you quickly look up a specific term or concept.

Chapter 1, "Using the Network," describes how to log in to dBASE®, share printers, and share files. This chapter also explains basic file locking and record locking concepts, and describes dBASE commands useful to a network user.

Chapter 2, "dBASE Network Programming Concepts," provides the following:

- Basic programming techniques for supplying data protection in a LAN environment
- A description of the refresh capability, which automatically updates database information to show changes made by other network users
- Procedures for creating programs that will run in both single-user and multi-user environments

Chapter 3, "Transaction Processing," describes how you can create a transaction log file of all changes to database files and, if necessary, recover your database files using the transaction log file.

Chapter 4, "Networking Commands and Functions," describes the syntax, defaults, and behavior of each dBASE command and function as it is used in programming for a LAN environment.

Chapter 5, "dBASE Security," describes techniques for defining and maintaining system security, such as creating passwords and protecting data files.

Conventions Used in This Manual

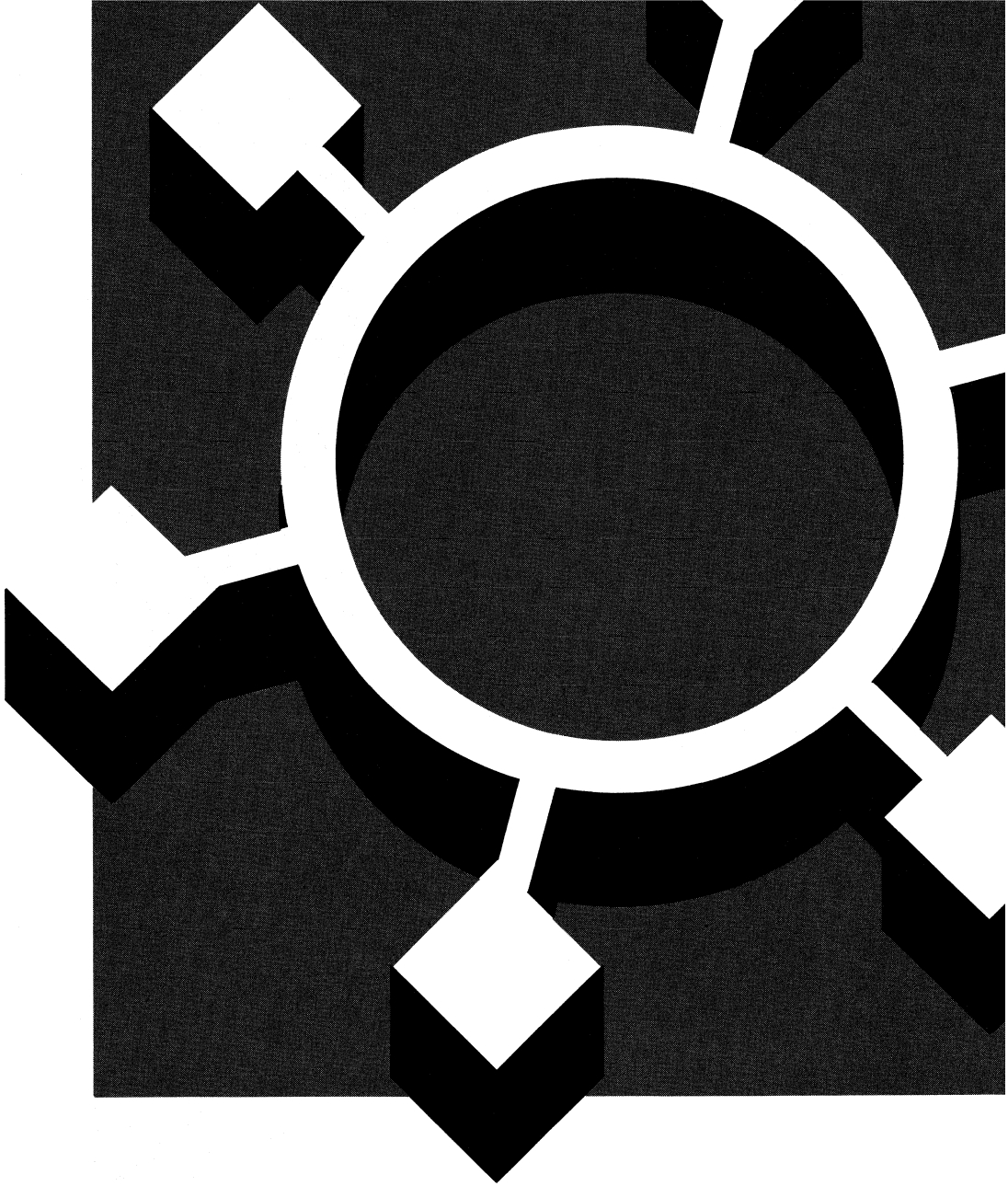
Understanding the information in dBASE IV documentation is easier when you are aware of the following conventions:

- The word *press* is used for keys you press; the word *type* or *enter* is used for information you must type in.
- Function keys (in the dedicated keypad on your keyboard) are shown by themselves (**F1**) or with their assigned functions (**F1 Help**).
- Commands or entries that you type appear in a different typeface and color.
- Instructions to press the RETURN or ENTER key are represented by the ↵ symbol.
- Italics are used for new terms when they first appear, and also for emphasis within the text.
- Menu names and items, messages you receive from the program, and other on-screen references are in bold type.
- Program listings appear in a different typeface, with a shaded background.
- Displays of computer screens and illustrations show the correct results of your instructions to the program, or clarify a definition or instruction.
- Notes and tips are in italics, with special marks, so you'll be sure to see them. Warnings have lines above and below their text, in addition to the warning mark, and thus stand out even more.

  Notes explain exceptions to or provide additional information about general rules. Tips give you helpful hints or shortcuts. Warnings help you avoid disastrous events such as losing data.

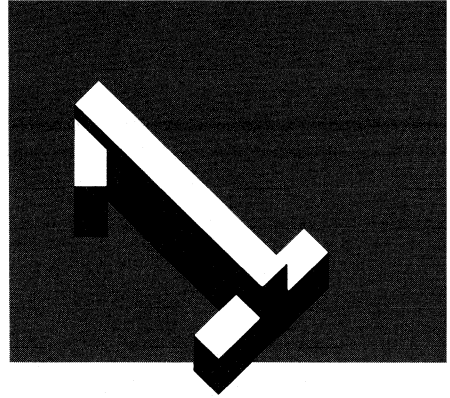
Network User

1. Using the Network



i 1 2 3 4 5 A Gl In

Using the Network



About This Chapter

This chapter describes concepts important to a network user, that is, anyone who uses a workstation on a local area network (LAN). This chapter includes information on the following topics:

- User procedures for logging in to dBASE
- Groups and how the group name you specified at login may prohibit you from using some dBASE files
- Locking concepts and how file and record locking may temporarily prohibit access to a file or a field in a file
- dBASE commands that are useful to a network user

Logging In to dBASE

To start dBASE IV, enter the DBASE command at the DOS prompt:

F > DBASE ↵

or enter the DBASE command followed by the name of the application program you want to start:

F > DBASE < filename > ↵

If your dBASE IV is not PROTECTed, the copyright notice appears followed by the Control Center. If you specified an application program name with the DBASE command, that program starts and the Control Center does not appear.

If your dBASE IV is PROTECTed, when you enter the DBASE command, the log-in screen appears (see Figure 1-1).

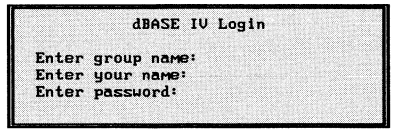


Figure 1-1 dBASE IV user log-in screen

You must enter a valid *user login* to access the system. A user login consists of a user group name, a log-in name, and a password.

Enter your user group name, log-in name, and password. The password does not appear on the screen.

If you make a mistake, you will not be granted access, and you will need to re-enter all three log-in items. You get three tries to type your log-in information correctly. If the information is still not correct, dBASE IV exits to the operating system.

If you log in correctly, the dBASE IV copyright notice appears, followed by the Control Center.

If you want to use dBASE IV from the dot prompt, select **Exit to dot prompt** from the Control Center **Exit** menu.

Understanding Groups

Your database administrator may have assigned files to *groups* for security purposes. The group name you enter on the dBASE log-in screen limits the database files you can access. If you are denied access to a file, you may not belong to the group allowed to access that file. Ask your network administrator. You may be given a login with a different group name that will allow you to access the file you want. Each database file can be accessed by only one group.

Locking Concepts

In a LAN environment, database files are shared with other network users. When you are using dBASE IV, another user may want to update the file or record that you are updating. The system includes *file locking*, so that two users cannot update a file at the same time, and *record locking*, so that two users do not update a record at the same time. These locks occur automatically when you enter information to modify a file or record, that is, when you are using a BROWSE, CHANGE, or EDIT command and you press any key that changes a record.

After you complete your update and press a navigation key to move to another file or record, dBASE IV automatically releases the file or record so other users can update it. Locking does not prevent you from viewing a file or a record that another user is updating.

The following sections describe what to do when you encounter a locked file or record.

Locked Files

If you attempt to open a file that another user has already opened for exclusive use, or attempt to lock a file that another user has already locked, this error message appears: **File is in use by < name >. Retrying lock, press Esc to cancel**. dBASE IV tries to lock the file repeatedly. If you press **Esc**, dBASE IV stops trying to lock the file.

Alternatively, the system may show the error message **File is in use by another. Retrying!** If you select **RETRY**, dBASE IV tries to lock the file again. If the lock still is not granted, the error message reappears. Try to access the file later. If you select **ABORT**, dBASE IV cancels your request to open the file. Select **HELP** to obtain additional information.

Locked Records

If you attempt to update a record, the system attempts to lock that record and all related ones. If another user has already locked that record or a related one, this error message appears: **Record is in use by < name >. Retrying lock, press Esc to cancel**. dBASE IV tries to lock the record repeatedly. If you press **Esc**, dBASE IV stops trying to lock the record.

If the lock is successful, the system determines if any of the fields have changed since you accessed the record. If any have, the system updates your screen to show the changes and allows you to abandon or continue your update.

Alternatively, the system may display the error message **Record is in use by another. Retrying!** If you select **RETRY**, dBASE IV tries to lock the record again. If the lock still is not granted, the error message reappears. Try to update the record later. If you select **ABORT**, dBASE IV cancels your request to update the record. Select **HELP** to obtain additional information.

Using dBASE Commands

Table 1-1 lists dBASE commands that can be entered at the dot prompt by the network user in a LAN environment. These commands, summarized here, are described in detail in Chapter 4.

Table 1-1 dBASE utility commands

dBASE Command	Use
DISPLAY/LIST STATUS	Displays or lists the status of file locks, and whether the files are opened for exclusive use
DISPLAY/LIST USERS	Displays the network-assigned workstation name of dBASE users who are currently using dBASE IV from the shared directory
SET PRINTER	Redirects print output between a local printer port and the shared network printer

DISPLAY STATUS and LIST STATUS

These commands provide information about the current dBASE IV session on the network. DISPLAY STATUS and LIST STATUS are identical, except that LIST STATUS does not pause periodically; DISPLAY STATUS does.

To use the DISPLAY STATUS command, enter **DISPLAY STATUS** or **DISPLAY STATUS TO PRINT**.

To use the LIST STATUS command, enter **LIST STATUS** or **LIST STATUS TO PRINT**.

If TO PRINT is not specified, the display or listing appears on the workstation screen. If TO PRINT is specified, the STATUS display or listing is directed to the currently defined printer and to the display. The currently defined printer is either a shared network printer or a local printer defined through the SET PRINTER command (described later in this section).

Here is a sample STATUS display. Notice the status of file locks listed for each open database file.

```
. DISPLAY STATUS

Currently Selected Database:
Select area:  1, Database in Use:  H:MYFILE.DBF    Alias:  MYFILE
          Memo file:   H:MYFILE.DBT
          Lock list:       2,      4,       6,      8  locked


Select area:  2, Database in Use:  H:ACCTG.DBF    Alias:  ACCTG
          Memo file:   H:ACCTG.DBT
          Lock list:   database locked


File search path:
Default disk drive:  H:
Print destination:  PRN:
Margin =     0
Refresh count =    0
Reprocess count =    0
Number of files open =    4
Current work area:  1

ALTERNATE  - OFF   DELIMITERS  - OFF   FULLPATH   - OFF   SAFETY      - ON
AUTOSAVE   - OFF   DESIGN      - ON    HEADING    - ON    SCOREBOARD  - ON
BELL       - ON    DEVELOP     - ON    HELP       - ON    SPACE       - OFF
CARRY      - OFF   DEVICE      - SCRN  HISTORY    - ON    SQL         - OFF
CATALOG    - OFF   ECHO        - OFF   INSTRUCT   - ON    STATUS      - OFF
CENTURY    - OFF   ENCRYPTION  - ON    INTENSITY  - ON    STEP        - OFF
CONFIRM    - OFF   ESCAPE      - ON    LOCK       - ON    TALK        - ON
CONSOLE    - ON    EXACT       - OFF   NEAR       - OFF   TITLE       - ON
DEBUG      - OFF   EXCLUSIVE   - OFF   PAUSE      - OFF   TRAP        - OFF
DELETED    - OFF   FIELDS      - OFF   PRINT      - OFF   UNIQUE      - OFF

Programmable function keys:
F2         - assist;
F3         - list;
F4         - dir;
F5         - display structure;
F6         - display status;
F7         - display memory;
F8         - display;
F9         - append;
F10        - edit;
CTRL-F1    -
CTRL-F2    -
CTRL-F3    -
CTRL-F4    -
CTRL-F5    -
CTRL-F6    -
CTRL-F7    -
CTRL-F8    -
CTRL-F9    -
CTRL-F10   -
SHIFT-F1   -
SHIFT-F2   -
SHIFT-F3   -
SHIFT-F4   -
SHIFT-F5   -
SHIFT-F6   -
```

DISPLAY USERS and LIST USERS

These commands display the network-assigned names of workstation users currently logged in to dBASE IV. To display these names, enter DISPLAY USERS or LIST USERS. Here is a sample USERS display:

```
. DISPLAY USERS
  Computer name
  ─────────────
  WKSTN1
  WKSTN4
  WKSTN3
 >WKSTN2
  WKSTN7
```

The character > marks the currently logged user.

SET PRINTER

The SET PRINTER command sends dBASE IV printer output to a network or local device. Use this command to redirect print output from a shared network to a local printing device and vice versa.

To send output to a network printer on an IBM PC, IBM Token-Ring, or Ungermann-Bass network, you would enter:

```
. SET PRINTER TO \\<computer name>\<printer name>=<destination>
```

- < computer name > is the network-defined server name. If you have established the network printer with the NET USE command, you cannot use this command. Since < computer name > must be a unique identifier for the server, it cannot be a group name.
- < printer name > is the network-defined printer name. The printer must be defined as the network printer used by < computer name > .
- < destination > identifies the installed printer: LPT1, LPT2, or LPT3, as appropriate. LPT must be specified.

The following commands spool the output of report CENSUS to the network printer (parallel printer number 1) attached to the server named ATLANTIC1:

```
. SET PRINTER TO \\ATLANTIC1\PRINTER=LPT1
. REPORT FORM CENSUS TO PRINT
. SET PRINTER TO LPT1
```

To send the report to the network printer on a Novell network, use the Novell SPOOL or CAPTURE command prior to starting dBASE:

```
F> SPOOL L=2 TI=5
F> DBASE
```

Spool LPT2: with the timeout option and redirect output to the network printer:

```
. SET PRINTER TO LPT2
```

Note that printing of one or more files in the network printer spool does not begin until the SET PRINTER command is issued. You must include a printer or device name to reset the printer to the default (currently defined network printer) or to another destination.

If you want to send the report to a *local device* on a Novell network, and you have spooled as shown above, simply redirect the output to the local printer port, as follows:

```
. SET PRINTER TO LPT1
```

The SET PRINTER command establishes a DOS device as the destination for printer output. < destination > can be the parallel line printers (LPT1 or PRN, LPT2, LPT3) or the COM devices (COM1, COM2). Note that it is not necessary to re-enter the command to reset the local device.

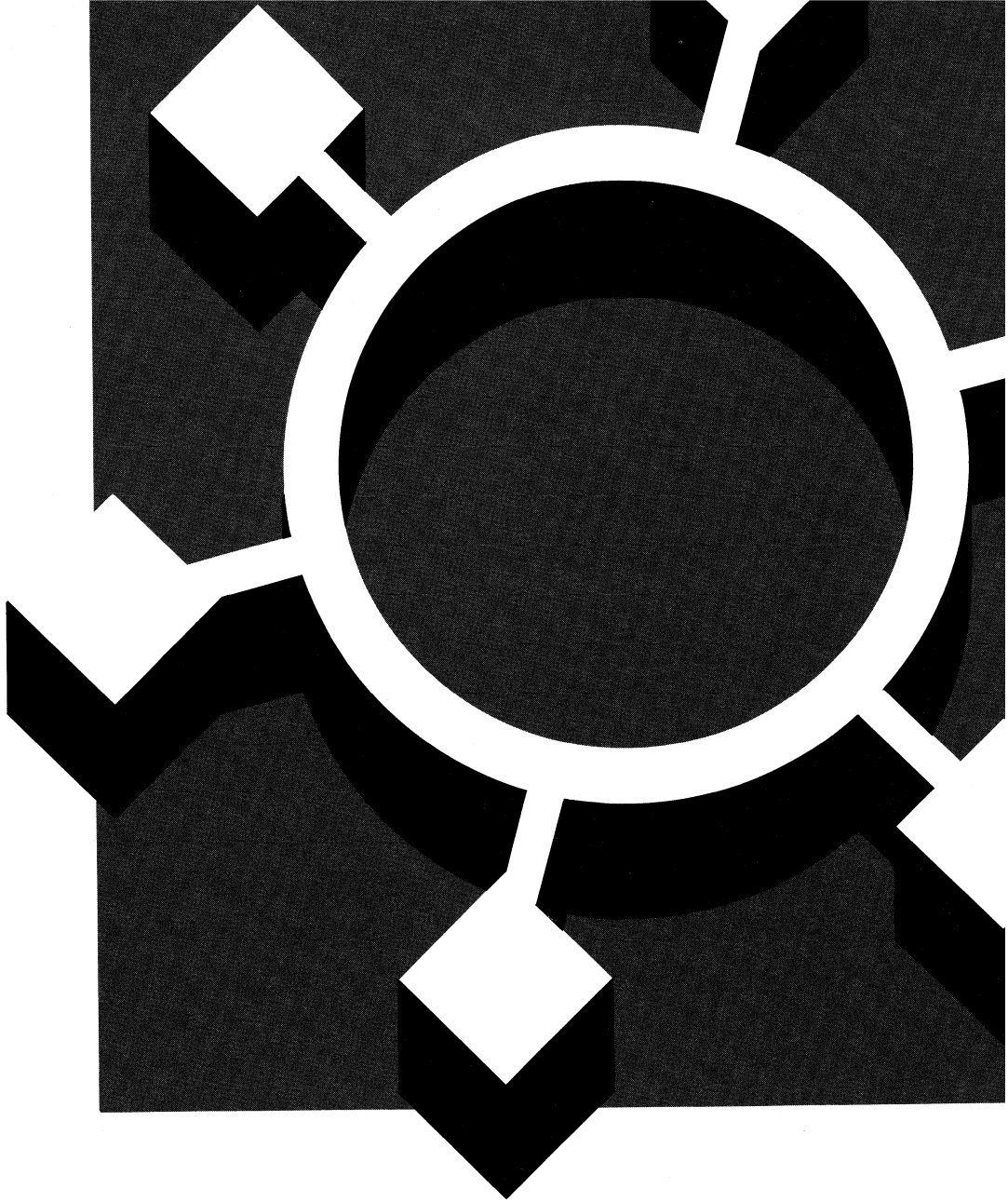
Using the TI option with the Novell SPOOL or CAPTURE command will close the spool file and send it to the printer automatically.

To send the report to the network printer on a 3Com 3 + network, enter:

```
F> 3P LINK LPT2: = < printer name >
```

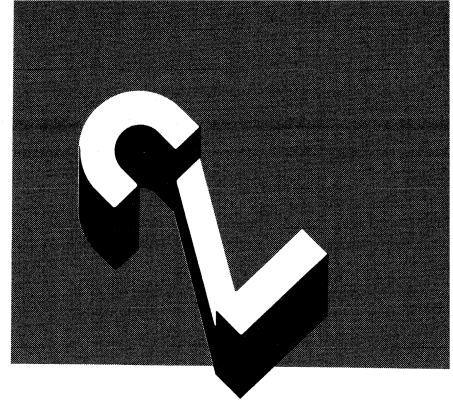
< printer name > is the network-defined printer name. The printer must be defined as the network printer used by < computer name > .

Network Programmer



i 1 2 3 4 5 A Gl In

dBASE Network Programming Concepts



About This Chapter

This chapter introduces the following dBASE IV programming tools and concepts used in a network programming environment:

- Data protection problems that can occur in a local area network and the automatic file and record locking solutions provided by dBASE IV.
- The two ways to open a file, exclusive or shared, and how each affects locking.
- Read-only and read/write file access attributes.
- Explicit locking tools that are supported in dBASE IV. These include the FLOCK(), RLOCK(), and LOCK() functions and the UNLOCK command.
- Error trapping procedures that use the ON ERROR command to trap errors, the ERROR() and MESSAGE() functions to return error values, and the RETRY command to control program execution.
- The refresh capability, which automatically updates database information to show changes made by other network users.
- Typical procedures used to test for the presence of a network, add a record to a database file, and edit a record already in a database file.
- Security tools — the ACCESS() function and the LOGOUT command.
- dBASE IV commands used in local area network (LAN) environments, including SET and USE EXCLUSIVE.
- Moving single-user programs to a multi-user environment.

This chapter also serves as an introduction to Chapter 4, which describes the commands and functions used in a LAN environment.

Data Protection

LAN systems face potential problems that do not occur in a single-user system because, in a LAN environment, users have access to shared data and program files.

One potential problem, called *collision*, could occur if more than one user attempted to edit or add data to a database at the same time. If collision were allowed to happen, either no user update would be successful and there would be no change to the data, or only one user would be successful, while all the other users would believe that they had gained access to and changed the data. In a worst-case situation, the database file might be damaged or destroyed.

Another problem, called *deadlock*, can happen when two users contend for each other's already locked files or records. For example, a deadlock occurs if user A needs to lock the file or record previously locked by user B, and user B attempts to lock the file or record already locked by user A. Figure 2-1 illustrates a deadlock situation in which two users attempt to cross-update a pair of records in the same file.

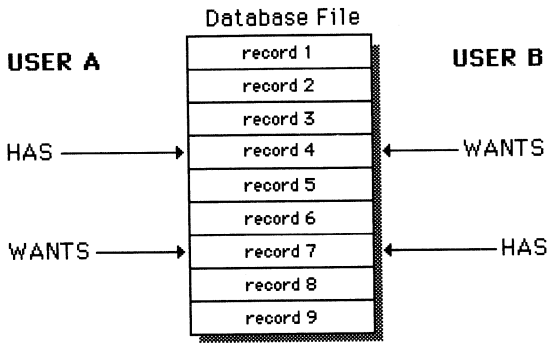


Figure 2-1 Deadlock

In a deadlock situation, a standoff occurs: neither user can access all the files or records necessary to complete a serial transaction, and, instead of exiting, each user repeatedly attempts to reaccess an unavailable file or record. The transactions of both users enter into a standoff, each waiting for the lock held by the other to be released. The only way to correct the deadlock is for one of the users to recognize the situation and back out of it.

A deadlock can also occur if two users contend for each other's files. This happens when multiple users are serially accessing each other's files and each is using a program sequence that includes the following commands:

```
SET EXCLUSIVE ON
SELECT
USE
```

In dBASE IV, the automatic retry facility helps users to recognize when they might be in a deadlock situation and to release their lock attempts. After the number of retries specified in the SET REPROCESS TO command is exhausted, the system displays a prompt box and the user can choose to cancel the lock attempt and end a possible deadlock.

Multi-user dBASE IV uses locking as a *concurrency control* mechanism, to ensure that concurrent transactions accessing the same database file at the same time do not interfere with each other's operation.

Three problems can adversely affect concurrent transactions and produce incorrect results: *lost update, uncommitted dependency*, and *inconsistent analysis.*

A lost update occurs if two users access and update the same record at approximately the same time. Both users are unaware that another user has accessed the record. The first user that updates the record loses out because the second user changes the values entered by the first.

An uncommitted dependency occurs if a user is allowed to update a record that has been updated by a transaction but has not been committed. If a roll-back occurs on the transaction, the user updating the record before it has been committed may be updating inaccurate data. See Chapter 3, "Transaction Processing," for a description of transaction processing and recovery.

Inconsistent analysis occurs if the first user calculates (adds, subtracts, averages, and so on) a series of database values and a second user changes any of those values before the first user completes the calculation. The first user ends up with an incorrect calculation because of the second user's interference.

Any LAN system must protect against attempts at simultaneous data access. It must allow multiple modifications to database files without losing data or corrupting indexes. Locking guarantees a user that other users will not interfere with the database file or record being updated. dBASE IV provides two ways to protect data against simultaneous access attempts: locking of shared files and locking of shared records. File and record locking are automatic. Whether locking is required at all depends on the *file open mode*; that is, whether the file is opened for *exclusive* or *shared* access.

Opening a File

A file is opened in one of two modes, exclusive or shared.

If a file is opened in exclusive mode:

- Only one requesting user can access the file at a time
- There is no need to lock the file or any record in the file

If a file is opened in shared mode:

- Multiple users on the network can access the file
- Files and records should be locked before updating

The file open mode is established and maintained from dBASE IV, which presets or defaults the file open mode every time a file is opened. The user can change how the file is opened with the SET EXCLUSIVE command.

NOTE
A file can't be shared unless it is in a shared directory or the user has rights to the directory. Shared directories are established at the network level. (See the appropriate appendix in Network Installation.*)*

Default File Open Modes

Table 2-1 summarizes by command how dBASE IV presets file open modes. dBASE IV defaults to exclusive use so that your files are not shared with other users. In a LAN environment, files are shared after the execution of a SET EXCLUSIVE OFF command in the Config.db file. Remember that:

- If you use the SET EXCLUSIVE OFF command to reset the default file open mode to shared, all files opened after you issue the command will be opened for shared access. Locking will be a concern, unless the file mode has been set as read-only at the network level.
- The SET EXCLUSIVE OFF command does not affect files already opened for exclusive use.

Table 2-1 Default file open modes

Command	File Type	How Opened
APPEND FROM	Database (.dbf)*	Shared
COPY STRUCTURE TO	Database (.dbf)*	Exclusive
COPY TO	Database (.dbf)*	Exclusive
CREATE	Database (.dbf)*	Exclusive
CREATE <file 1> FROM <file 2>	Database (.dbf)* Database (.dbf)*	Exclusive <file 1> Shared <file 2>
CREATE LABEL	Labels (.lbl)	Exclusive
CREATE QUERY	Query (.qry)	Exclusive
CREATE REPORT	Report (.frm)	Exclusive
CREATE VIEW	View (.vue)	Exclusive
DO <filename>	Command (.prg)	Shared
INDEX ON	Index (.ndx)* Index (.mdx)*	Exclusive Exclusive
JOIN	Database (.dbf)*	Exclusive
LABEL FORM	Labels (.lbl)	Shared
MODIFY COMMAND <filename>	Command (.prg)	Exclusive
MODIFY COMMAND <filename>	Format (.fmt)	Exclusive
MODIFY COMMAND <filename>	Procedure (.prg)	Exclusive
MODIFY LABEL	Labels (.lbl)	Exclusive
MODIFY QUERY	Query (.qry)	Exclusive
MODIFY REPORT	Report (.frm)	Exclusive
MODIFY VIEW	View (.vue)	Shared

(continued)

Table 2-1 Default file open modes (*continued*)

Command	File Type	How Opened
REPORT FORM	Report (.frm)	Shared
RESTORE	Memory (.mem)	Shared
SAVE	Memory (.mem)	Exclusive
SET ALTERNATE TO	Alternate (.txt)	Exclusive
SET CATALOG TO	Catalog (.cat)	Shared
SET FILTER TO FILE	Query (.qry)	Shared
SET FORMAT TO	Format (.fmt)	Shared
SET INDEX TO	Index (.ndx)*	Exclusive**
SET PROCEDURE TO	Procedure (.prg)	Shared
SET VIEW TO	View (.vue)	Shared
SORT	Database (.dbf)*	Exclusive
TOTAL	Database (.dbf)*	Exclusive
UPDATE FROM	Database (.dbf)*	Shared
USE	Database (.dbf)*	Exclusive**
USE < file > INDEX	Index (.ndx)* Memo (.dbt)*	Exclusive** Exclusive**

* The default open mode for index and memo files is always the same as the default for the database file with which they are associated.

** The default is exclusive if SET EXCLUSIVE is ON.

NOTE
The following special cases should be noted:

- *Files accessed by the Control Center and the CHANGE and EDIT commands are opened as shared files if SET EXCLUSIVE is OFF.*
- *The commands CONVERT, INDEX ON < key expression > TAG < tag name >, where < tag name > is an .mdx file, INSERT [BLANK], MODIFY, PACK, REINDEX, RESET, and ZAP can be used only if the file has been opened for exclusive use. The default for SET EXCLUSIVE is OFF. If you attempt to use one of these commands and the file is not opened for exclusive use, an error message appears. You will need to close the file and reopen it for exclusive use.*
- *The CREATE and SAVE commands automatically set a file for exclusive use.*

Default File Open Attribute Rules

The default opening mode for a file consists of a combination of the file open mode (exclusive or shared) and the file access attribute (read-only or read/write). The "Using dBASE Files" section, later in this chapter, discusses the file access attribute and the full default opening mode.

Once you open a file, some commands will check the lock status of the file and may automatically lock that resource for the course of the operation. These commands are identified later in this chapter in the "Automatic File Locking" section. Other dBASE commands may require that the file be opened for exclusive use. These commands are identified above in the "Opening a File" section.

SET EXCLUSIVE OFF Command

If you plan to access application files simultaneously from more than one workstation, those files must be opened for shared use. If your application shares files, note that:

- The default for the multi-user SET EXCLUSIVE command is OFF, which lets you share the files
- Files opened after you issue the SET EXCLUSIVE ON command are not shared on the network

Refer to Chapter 4 for more information on the SET EXCLUSIVE command.

Locking Features

dBASE IV automatically attempts to lock a file or record when you execute any command that modifies a file or record. If you are using a CHANGE or EDIT command, dBASE IV attempts a lock when you enter information that updates a record; that is, when you press any key that changes the record. Automatic locking ensures that each file or record you change is guarded from change by other network users while you are making your update.

Automatic file and record locking do *not* require you to explicitly lock or unlock the file or record you are updating. You do not need an FLOCK() or RLOCK() function to create a lock, and you do not need an UNLOCK command to release a lock. The "Automatic File Locking" section, below, lists the commands that automatically lock a file before the command is executed. The "Automatic Record Locking" section, below, lists the commands that automatically lock a record.

If the file or record you want to update is already locked by another network user, the system displays a prompt box and you can retry or abort the lock attempt. The LKSYS() function allows you to determine what network user has locked a file or record, plus the date and time it was locked.

Automatic file and record locking make it easy to move single-user applications to a multi-user environment. When you do this, it is not necessary to program a lock before each command that updates a file or record and an unlock after each update.

While you may want to manually lock a file or record to explicitly reserve it in some situations, you should treat manual locking in dBASE IV as an exception you will use rarely, if at all. Manual locking is necessary, however, if you plan to run your application under dBASE III PLUS™.

The following locking discussion is for programmers whose applications use shared files.

dBASE IV Locking Levels

Locking happens at four levels:

1. Automatic file locking. At this level, users running dBASE IV on a network are unaware of the file locking features that dBASE IV automatically provides.
2. Automatic record locking. At this level, users running dBASE IV on a network are unaware of the record locking features that dBASE IV automatically provides.
3. Explicit file locking. By using file locking commands and functions, users can claim and relinquish specified files.
4. Explicit record locking. At this level, a user locks a record in a file, thereby preventing others from accessing that record only.

Each locking level is more powerful and flexible than its predecessor, and requires more understanding about data sharing.

NOTE

- *If a shared file is opened only for reading, it need not be locked. There is no danger of problems arising from multiple access to shared data, unless there is an update attempt.*
- *The system automatically locks related files. The user need not consider locking index files or other related dBASE files.*
- *The system automatically locks related records.*
- *Locks permit any user to read the locked file or record, even if it is being updated.*

Automatic File Locking

When a file is not being used exclusively or is not locked, dBASE IV locks the file in use automatically before the following commands are executed:

APPEND FROM	DELETE < scope >	REPLACE < scope >
AVERAGE	INDEX	REPORT
CALCULATE	JOIN	SORT
COPY	LABEL	SUM
COPY STRUCTURE	PROTECT	TOTAL
COUNT	RECALL < scope >	UPDATE

If the file can't be locked, a prompt box appears. You can then retry or cancel the lock attempt. If an automatic file lock is used, dBASE IV automatically unlocks the file when command execution is completed.

NOTE
Specifying <RECORD #> as the scope for the DELETE, RECALL, and REPLACE commands does an automatic record lock instead of a file lock.

You can use the SET LOCK command to enable or disable automatic locking for a subset of the commands listed above. To enable automatic locking for the following commands, you must issue SET LOCK ON before you use any one of them.

AVERAGE	INDEX ON <key expression>	SORT
CALCULATE	TO <.ndx filename>	SUM
COPY	JOIN	TOTAL
COPY STRUCTURE	LABEL	
COUNT	REPORT	

The above commands are read-only for the current database file.

WARNING
These commands will work without locking, but data integrity is not guaranteed if automatic locking is disabled. If you use SET LOCK OFF, there will be greater concurrency and a greater risk of possible inconsistent analysis.

Some of these commands have two phases, reading and writing. For these commands, SET LOCK works only on the reading phase. When you are writing to the file, it is automatically opened for exclusive use. These commands are:

COPY	JOIN
COPY STRUCTURE	SORT
INDEX ON <key> TO <file.ndx>	TOTAL

Automatic Record Locking

In multi-user dBASE IV, all commands that update a record lock the record automatically before making the update. This is true for any *serial database transaction* in which records are incrementally updated instead of fully changed. If the records in a file are to be fully changed rather than incrementally updated, record locking is not an issue. A serial database transaction is one where the following sequence of events occurs:

1. The record to be changed must be locked for the user who wishes to change the data. After a record is locked, the data to be changed must be displayed. This step ensures that the user sees the most current data at the time it is locked. At that point, no other user may modify the data.
2. The modification is made.
3. The modified record is saved, and then is unlocked.

When a file is being shared, dBASE IV automatically attempts to lock the record before the following commands are executed:

APPEND [BLANK]	RECALL
DELETE	REPLACE

dBASE IV attempts to lock a record automatically when you press any key that modifies the record, that is, any key that changes the record. If the record is already locked by another user, the *automatic retry* facility automatically attempts to lock the record the number of times you specified in a SET REPROCESS command. If the system exhausts all its attempts to lock the record, it displays a prompt box, which allows you to continue your attempt to lock the record or cancel your request. If you press any navigation key that moves to another record, the locked record is automatically unlocked.

You may still want to use explicit record locking to solve lost update or uncommitted dependency problems.

Explicit File Locking

If you use explicit file locking, you must lock the file before using a command that works on an entire file. dBASE IV supports explicit file locking with the FLOCK() function for compatibility with dBASE III PLUS. Although explicit file locking is not required because of dBASE IV's automatic file locking facility, explicit file locking can be used when shared data needs to be locked, but only one user at a time needs to update the data. Explicit file locking requires a user or application program to manually lock a file before it is updated, and then manually release it. Because the entire file is locked, there is no need to lock records.

Explicit file locking can be useful when updates can affect many records or index files. It also ensures in advance that a required set of files is available. If you use the FLOCK() function to explicitly lock a file, you must use the UNLOCK command to release the locked file. Other users can still read a file that you have explicitly locked.

Explicit Record Locking

For compatibility with dBASE III PLUS, you can still use LOCK() or RLOCK() to explicitly lock a record. Another user can read a locked record, regardless of whether it has been automatically locked or explicitly locked. If you use either the LOCK() or RLOCK() function to lock a record, you must use the UNLOCK command to release the locked record.

The Explicit Locking Process

In dBASE IV, locking prevents collision. A locking function allows dBASE IV to determine whether a file or record is locked by another user. If not locked, the function returns a logical true (.T.) value and places a lock on the file or record, protecting access for the user who just tested the status of the record.

You can implement a lock through a locking function: FLOCK() for files and RLOCK() (or its synonym, the LOCK() function) for records. These locking functions:

- Test for a lock
- Lock an unlocked file or record
- Return logical values that result from their test

NOTE
The locking functions:

- *Lock only an unlocked file or record.*
- *Operate differently than most functions in dBASE IV, because locking functions can execute an action (locking) as well as return a value (.T. or .F.).*

Testing for a Lock

A lock function checks the status of the file or record lock. The file or record is either available (not yet locked), or not available (already locked). If the file or record is not locked when the function tests its lock status, the lock is placed, preventing update to the file or record by other users. If the file or record is locked when a lock request arrives, the locking function cannot be executed, and the locking request must be made again later. See the "Locked Files" and "Locked Records" sections in Chapter 1.

Locking a File or Record

At the instant a lock function determines that a file or record is not locked, the user or application program gains control of the file or record. The lock placed on the file or record informs subsequent lock requests that the file or record is not available until the current user is finished with it.

The record lock affects one or more records and their associated records. All other records in a file are available to other users.

Returning a Logical Value

A lock function returns a logical value of true (.T.) or false (.F.). It returns true when the file or record being tested is not currently locked by any other user. If multiple users simultaneously test for a lock, only the first test received gets a true value. A lock attempted by anyone else after that will test false and has no effect until the file or record is unlocked.

Unlocking an Explicitly Locked File or Record

Release an explicitly locked file or record by using the UNLOCK command (described in Chapter 4), by locking another file or record, or by closing a database file with a CLOSE, USE, CLEAR ALL, or QUIT command. Then, the next lock function on the file or record will test true and the lock will be placed for the new user, who will gain control of the file or record.

Using Explicit Locking

You can use explicit file or record locking interactively or implement them within an application program.

Using an Explicit Lock Function Interactively

At the dot prompt, you can type ? FLOCK() or ? RLOCK() (or alternatively, LOCK()), followed by a ↵. dBASE IV responds with **.T.** if the file or record is locked successfully. The file or record is now locked. If another user attempts to place a lock on the file or record, dBASE IV responds with **.F.**, indicating that the file or record is already locked.

Note that it is not necessary to issue a locking function before using a command that automatically locks a file or record. See the "Automatic File Locking" and "Automatic Record Locking" sections above.

For example, if you want to replace two records in a database, you might want to lock the records before issuing the REPLACE command. The following example uses the RLOCK() function and UNLOCK command to show how the lock can be implemented interactively:

```
. USE employee
. ? RLOCK("5,7","employee")
.T.
. GOTO 5
EMPLOYEE: Record No 5
. REPLACE Name WITH "NELSON"
1 record replaced
. GOTO 7
EMPLOYEE: Record No 7
. REPLACE Name WITH "LAMBERT"
1 record replaced
. UNLOCK
```

The sample application included on the Samples Disks, and described at the end of this chapter, includes examples of using the REPLACE command with locks in a program file.

When in full-screen mode, you may choose to lock the currently viewed record from the keyboard by simply pressing **Ctrl-O**. Press **Ctrl-O** again to unlock the record.

For the full-screen operation on shared access files, the record lock status is displayed in the network section (the fourth section) of the status bar as **Record unlocked** or **Record locked**.

Explicit Locking in Application Programs

An expression can be added to a lock function to qualify or expand it. This enhances its usefulness to dBASE IV network programmers. An expression using FLOCK(), RLOCK(), or LOCK() is a logical expression with a value of either true (.T.) or false (.F.). For example:

```
DO WHILE .NOT. RLOCK()
   DO TIMER
   IF TIMES_UP
      ? CHR(7)
      DO SHOW_MSG WITH "TIME-OUT ON WAITING FOR RECORD TO UNLOCK..."
      ? CHR(7)
      ON KEY LOOP
      DO SHOW_MSG WITH "PRESS ESC KEY TO EXIT OR SPACE KEY TO CONTINUE WAITING..."
   ENDIF
ENDDO
```

For more extensive examples of file and record locking, refer to the discussions of the FLOCK(), LOCK(), and RLOCK() functions and the UNLOCK command in Chapter 4. The sample network application program, Employee.prg, included on the Samples Disks further illustrates how the RLOCK() function and UNLOCK command can be used.

Using dBASE Files

How dBASE files are used depends on the full opening mode for a file. The full opening mode for a file consists of a combination of the file open attribute (exclusive or shared) and the file access attribute (read-only or read/write).

File Access Attributes

The file access attribute is the permission to read from and write to a file. It determines what a user can do with the file once it is opened. The file access attribute is either *read-only* or *read/write*.

Read-only means that the user can view the contents of the file, but can't change it.

Read/write means that the user can look at and change the file. The file access attribute can be established and maintained at the network level. For database files, file access attributes can also be assigned through the PROTECT command described in Chapter 5.

File Access Attribute Rules

dBASE IV uses the following rules to assign the default file access attribute:

1. If any dBASE file is being created or modified, it is opened for read/write use.
2. If a command, format, label, query, report, or view file is being used only to get information, it is opened for read-only use.

You can override the dBASE default by using the DOS command ATTRIB to set the file access attribute to read-only. For database files, use the PROTECT program to set the access privilege.

Errors and Error Recovery

How errors are processed in dBASE IV depends on whether:

- The user is at the dot prompt
- A full-screen command has been issued
- A command file is being executed, and an ON ERROR command has been issued

At the dot prompt, error processing is first handled by the automatic retry facility. This facility generates the program's error condition when the number of retries specified in the SET REPROCESS TO command is used up and the user selects **CANCEL**.

If a full-screen command has been issued, error processing is handled automatically. Error messages are displayed in an error box.

If an error is not handled automatically, dBASE IV returns an error state when the error occurs. If the error state is not captured through an ON ERROR procedure, dBASE IV displays the error.

ERROR() and MESSAGE()

dBASE IV provides two functions for returning error values: ERROR() and MESSAGE(). The ERROR() function returns the number of an error. The MESSAGE() function returns the error message text. ERROR() and MESSAGE() are used within an error processing procedure. ON ERROR must be active for these functions to receive a value. In a LAN environment, the programmer uses the ERROR() function to trap recoverable error conditions, such as attempts to lock an already locked file or record.

Chapter 4 details the ERROR() and MESSAGE() functions and includes examples of their use.

NOTE
The FLOCK(), RLOCK(), and LOCK() functions do not return an error message or an error message number. These functions return only .T. or .F.

RETRY

The RETRY command restores control to the calling program or procedure at the same line that called the program. RETRY is frequently used within an ON ERROR procedure to re-execute a command following an error. RETRY differs from RETURN because RETRY executes the same line in the calling program, while RETURN executes the next line.

Chapter 4 describes the RETRY command and gives examples.

The ERROR() and MESSAGE() functions and the RETRY command are only used for capturing errors when:

- A command file is being executed
- An ON ERROR command has been issued to trap errors, and
- The error message will not be displayed on the screen

Testing For a Network

You can use the NETWORK() function to test for the presence of a network. This function returns a logical true (.T.) if the multi-user version of dBASE IV is running, or a logical false (.F.) if the single-user version is running. The following example tests for the presence of the multi-user version of dBASE IV and prompts to determine if the user wants to know who is logged in to the network:

```
IF NETWORK()
   ACCEPT "Do you want to see who else is logged in?" TO answer
   IF UPPER(answer)="Y"
      DISPLAY USER
   ENDIF
ENDIF
```

Adding a Record

This section describes one way to write an application program that allows users to add a record to a database. The Library.prg and Employee.prg programs in *Sample Programming Code* illustrate these steps in a complete application program. At the end of each step below, you are referred to the procedure that contains the commands shown in that step.

1. Display a main option menu on the screen that allows a user to select the needed function:

```
DEFINE BAR 1 OF main_mnu PROMPT "— OPTION MENU —" SKIP
DEFINE BAR 2 OF main_mnu PROMPT " Add record"
DEFINE BAR 3 OF main_mnu PROMPT " Edit record"
DEFINE BAR 4 OF main_mnu PROMPT " Delete record"
```

   The user can press the first letter of the menu choice or move the highlight and press ↵. See the Bar_def procedure in the Library.prg program. If the user selects the **Add user** option, the Barpop procedure calls the Add_new procedure.

2. Initialize the memory variables:

```
STORE SPACE(15) TO lastname
STORE SPACE(10) TO firstname
```

See the Init_fld procedure, which is called by Add_new.

3. Display the text that prompts for the data to be entered by the user, and specify the memory variables that will be modified:

```
@ 5, 4  SAY "LAST NAME:" GET m->lastname PICTURE "!XXXXXXXXXXXXX"
@ 5, 33 SAY "FIRST:"     GET m->firstname PICTURE "!XXXXXXXXX"
```

See the Screen procedure in Employee.prg.

4. Wait for and read the user input:

```
READ
```

See the Add_new procedure in the Library.prg program.

5. Append a blank record, which will be replaced in step 6 with the data in memory:

```
APPEND BLANK
```

The APPEND BLANK command uses automatic record locking to ensure that the record will not be changed by any other network user. See the Sav_data procedure in the Library.prg program.

6. Replace the blank record with the data in memory:

```
REPLACE lastname WITH m->lastname, ;
        firstname WITH m->firstname
```

See the Repl_fld procedure in Employee.prg. The REPLACE command uses automatic record locking to ensure that the record will not be changed by any other network user. The record stays locked only for the shortest time necessary, and is automatically unlocked immediately after it is replaced.

Editing a Record

To change an existing record in a database, follow the same sequence as described in "Adding a Record." When the user selects **Edit record** from the main option menu, the Barpop procedure calls the Edit procedure in Library.prg. The Edit procedure uses the CHANGE() function to determine if anyone on the network has changed the record since the user first accessed it, and then reads the user input. Step 5, appending a blank record, is unnecessary.

Updating Viewed Information

If you are using the BROWSE or EDIT command to view database information, other users on your network may be entering changes to the database. The SET REFRESH command can be executed before using the BROWSE or EDIT command to ensure that changes made by other users will be updated on your screen at specified intervals. See the SET REFRESH command in Chapter 4 for detailed information.

Security at the Programming Level

Chapter 5 describes the file and field access security features available to the database administrator through the PROTECT command. The application programmer can use the ACCESS() function and the LOGOUT command to impose security from an application program.

The ACCESS() function can be used to control program branching based on access level. The description of the ACCESS() function in Chapter 4 includes an example of limiting code access by testing the user's access level returned by the ACCESS() function.

The LOGOUT command logs out the current user and implies that a new user is to be logged in. For security purposes, the LOGOUT command clears all memory variables in the history buffer. When the LOGOUT command is processed, the workstation screen clears and a log-in screen displays in which the user enters the log-in name, password, and group name.

Other dBASE Networking Commands

Table 2-2 lists and describes other commands of particular interest in a LAN environment. These commands are detailed in Chapter 4.

Table 2-2 Other networking commands

Command	Description
CONVERT	Permits use of the CHANGE() and LKSYS() functions. Also allows change detection with automatic locking.
DISPLAY/LIST STATUS	In a network environment, displays lock status along with the information displayed in a single-user environment.
DISPLAY/LIST USERS	This command displays the network-assigned names of workstations currently logged in to dBASE IV.
SET	In a network environment, AUTOSAVE, ENCRYPTION, EXCLUSIVE, LOCK, REFRESH, and REPROCESS are added to the list of system parameters that can be displayed and set.
SET AUTOSAVE	This command automatically updates the disk file and directories after I/O operations.
SET ENCRYPTION	This command controls whether copied files are encrypted when created.
SET EXCLUSIVE	This command determines the file open attribute of all database files USEd (after entering the command) during the dBASE IV session.
SET LOCK	This command enables and disables automatic locking for a subset of commands.
SET PRINTER	This command redirects printer output between shared network printers and local printers.
SET REFRESH	This command updates viewed database information if changed by another.
SET REPROCESS	This command sets the number of times dBASE IV tries a network command or function before producing an error message or a logical result for a function in case of network conflicts.
USE EXCLUSIVE	This command identifies a database file and specifies that access is not to be shared.

The Sample Application

The Samples Disks include an order entry application program. The Library.prg program in that application illustrates basic network programming concepts. This program includes procedures that:

- Define a pop-up menu
- Add, change, and delete records
- Use automatic record locking
- Time out after waiting for a record to unlock
- Demonstrate use of the RETRY command

Library.prg contains code that is common to all modules in the order entry application. Networking code is set off by rows of asterisks. The entire application will run on a single-user or a multi-user system, but all networking code is contained in one core module, Library.prg. The other modules in the application do not require, nor do they contain, any network commands.

To run the order entry application:

1. The files from the Samples Disks must reside on your hard disk drive. (If these files are not on your hard disk, request the Samples Disks from your database administrator and copy them to this directory using dBSETUP.)
2. Type **DBASE** and press ↵ while in the \DBASE directory, to access dBASE.
3. Enter the following commands at the dot prompt:

```
. SET PATH TO C:\DBASE\SAMPLES ↵
. DO BUSINESS ↵
```

See the *Sample Programming Code* booklet for a complete listing of Library.prg.

Moving a Single-User Program to a Multi-User Environment

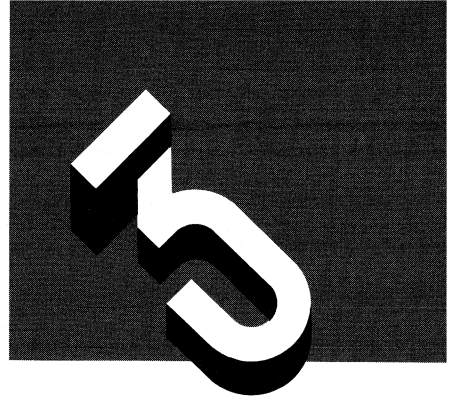
Because of automatic locking, dBASE IV single-user application programs can be moved to a multi-user environment with a minimum of programming effort. Automatic locking places a lock on a file or a record before executing any command that updates a database, and automatically releases the lock after the update. You need not place an explicit lock (with RLOCK(), LOCK(), or FLOCK()) before commands that update a database file, and you need not execute an UNLOCK command after an update.

Before you run a single-user application program on a multi-user system, you must do the following:

- Enter a SET EXCLUSIVE OFF command to make subsequently opened files shareable on the network. This command is required in a multi-user program because the default file-opening mode opens the file for exclusive use.
- Enter multi-user timing mechanisms, where appropriate, to retry a lock on a file or record a specified number of times. See the SET REPROCESS TO command.
- Enter a SET LOCK OFF command, where needed, to disable automatic locking. See the SET LOCK command.
- When you are using a command that requires EXCLUSIVE to be set on, you must SET EXCLUSIVE ON before using the file, as follows:

```
SET EXCLUSIVE ON
USE <filename> ZAP
```

Transaction Processing



dBASE IV provides a transaction processing capability, which creates a transaction log file of all changes to database files and ensures that you can restore database files interrupted while being updated. If recovery becomes necessary, you use the transaction log file to restore the values in your database files to those that existed before the transaction began.

This chapter describes how to do the following:

- Begin and end a transaction
- Recover database files when a transaction is interrupted
- Retry a transaction
- Test for a successfully completed recovery
- Determine if a transaction is in progress
- Test for a successfully completed transaction
- Specify USE commands in a transaction
- Use the transaction log file to its fullest
- Recover database files after a system failure
- Use locking in a transaction
- Use transaction processing at the dot prompt

Transaction processing is meant to be used in multi-user dBASE IV to solve system-interruption problems on a network. The transaction processing commands and functions described in this chapter, however, also work in single-user dBASE IV.

If you use transaction processing in SQL, refer to the "Multi-User and Transaction Programming" section in Chapter 6 of *Using dBASE IV SQL*.

NOTE
If you use transaction processing on a network, you cannot have two users logged in to the network operating system with the same name. The first eight characters of any network log-in name must be unique.

Beginning and Ending a Transaction

A *transaction* is a unit of work which contains a sequence of database operations. These operations change one or more database files. Before the transaction begins and after the transaction ends, your database files are in a *consistent state*, that is, all intended modifications have been made. If the transaction is interrupted, your database files are in an *inconsistent state*, that is, only some of the intended modifications have been made.

Once you start a transaction with the BEGIN TRANSACTION command, all data commands subsequently executed are considered part of the transaction until dBASE IV encounters an END TRANSACTION command. Typically, a transaction contains a sequence of updates. In the following example, all salaries are replaced with a higher salary:

```
BEGIN TRANSACTION
   REPLACE ALL salary WITH salary * 1.1
END TRANSACTION
```

If the above series is interrupted before all replaces are completed, the database remains in an inconsistent state. The END TRANSACTION command signals a successful completion of the transaction and tells the system that the updates can be made permanent. In effect, the END TRANSACTION command *commits* the transaction.

When a user begins a transaction, dBASE IV creates a transaction log file for that user in which it records changes to database files. If recovery becomes necessary, it uses this file to accomplish the recovery. When dBASE IV encounters the END TRANSACTION command, it purges the transaction log file (which is no longer needed). It also releases all locks at the end of a transaction.

Recovering Database Files

Transaction processing ensures that you can undo an interrupted transaction. When you recover or *rollback* a transaction, all updates (including additions and deletions) are reversed and all database files are left in their original consistent state, as if the transaction was never executed. A rollback is necessary if any interruption occurs while a transaction is in process. The interruption might be a power failure, the inability to access a file or record, or any other error that leaves your transaction data in an inconsistent state.

The following series of commands tests for any type of error that occurs during the transaction and starts a recovery if an error occurs:

```
ON ERROR ROLLBACK
BEGIN TRANSACTION
   REPLACE ALL salary WITH salary * 1.1
END TRANSACTION
ON ERROR
```

You will typically use ON ERROR to transfer control to a separate procedure that informs the user when an error occurs and specifies the type of error. Here is an example:

```
SET REPROCESS TO 15
ON ERROR DO err_proc
BEGIN TRANSACTION
   REPLACE ALL salary WITH salary * 1.1
END TRANSACTION
ON ERROR

PROCEDURE err_proc
DO CASE
   CASE ERROR() = 108   && File is in use by another
      @ 21,15 SAY "Your transaction has encountered a file in use by another."
      ROLLBACK
   CASE ERROR() = 109   && Record is in use by another
      @ 21,15 SAY "Your transaction has encountered a record in use by another."
      ROLLBACK
ENDCASE
RETURN
```

The procedure shown above uses the ERROR() function to test for two specific types of errors. A rollback occurs only if one of the errors in the CASE statement occurs.

Retrying a Transaction

A user may want to retry a transaction containing a command that has encountered an error condition. The following error procedure allows the user to retry a transaction or abandon the transaction and start a recovery:

```
PROCEDURE err_proc
choice = " "
@ 21,15 SAY "Your transaction has encountered an error condition."
@ 22,15 SAY "Do you wish to RETRY? (Y/N)";
GET choice PICTURE "!"
READ
@ 21,15 CLEAR TO 22,65
IF choice = "Y"
   RETRY
ELSE
   @ 21,15 SAY "Rolling back your transaction now. Please wait."
   ROLLBACK
ENDIF
RETURN
```

Testing for a Successful Recovery

The ROLLBACK() function can be used to determine whether a recovery was successfully completed. The following error procedure informs the user when an error condition occurs, a rollback starts, a rollback ends, and whether a rollback is successful or unsuccessful:

```
PROCEDURE err_proc
choice = " "
@ 21,15 SAY "Your transaction has encountered an error condition."
@ 22,15 SAY "Do you wish to RETRY? (Y/N)";
GET choice PICTURE "!"
READ
@ 21,15 CLEAR TO 22,65
IF choice = "Y"
   RETRY
ELSE
   @ 21,15 SAY "Rolling back your transaction now. Please wait."
   ROLLBACK
   IF ROLLBACK()
      @ 21,15 SAY "Rollback completed successfully."
   ELSE
      @ 21,15 SAY "Rollback not successful. Restore from backup."
   ENDIF
ENDIF
RETURN
```

Testing for a Transaction in Progress

The ISMARKED() function can be used to determine whether a database file is having transaction updates added to it and is in an inconsistent state:

```
USE customer
IF ISMARKED()
   @ 5,5 SAY "The file you need has an active transaction on it."
   @ 6,5 SAY "Please try again later."
   RETURN
ELSE
   REPLACE ALL salary WITH salary * 1.1
ENDIF
```

Testing for a Completed Transaction

The COMPLETED() function can be used to determine whether a transaction was successfully completed. Here is an example:

```
SET REPROCESS TO 15
ON ERROR DO err_proc
BEGIN TRANSACTION
   REPLACE ALL salary WITH salary * 1.1
END TRANSACTION
ON ERROR
IF COMPLETED()
   @ 21,15 SAY "Transaction successfully completed."
ENDIF
```

Specifying USE Commands in a Transaction

You may execute a USE command inside or outside a transaction. If you execute multiple USE commands inside a transaction, you must have assigned them to separate work areas, as shown in the following example:

```
SET REPROCESS TO 15
ON ERROR DO err_proc
BEGIN TRANSACTION
   USE clients ORDER lastname              && Assigned to work area 1
   USE inventry IN 2 ORDER part_no         && Assigned to work area 2
   SEEK mlastname
   REPLACE orders WITH orders + 1
   IF SEEK(mpart_no, "inventry")
      SELECT 2
      REPLACE quantity WITH quantity - 1
   ENDIF
END TRANSACTION
```

A Complete Example

The following program contains the commands and functions introduced above. Notice the use of the COMPLETED(), ISMARKED(), and ROLLBACK() functions. The COMPLETED() function tests for a successfully completed transaction. The ISMARKED() function determines if a transaction is in progress. The ROLLBACK() function determines whether the rollback was successful. The program displays messages to keep the user informed of significant events.

In the following example, the SEEK command locates an account number and looks into a field in the accounts file to see if this account has specified that it is time to order again. If Yes, a new invoice is added to the invoices file. The program then goes into the inventory file, and looks for the item code as specified in the accounts file (SEEK > item_code). In the inventory file, the example decrements the quantity on hand by the order quantity specified in the order file. If the quantity on hand is less than or equal to zero, the program goes to the vendor file, looks for the vendor's number, and flips a flag that says it is time to order more. It then goes to the invoices file, and replaces the amount with the quantity multiplied by the price. The program then goes back to the account file and replaces the balance with the amount.

This program uses the LKSYS() function. All databases used by this program must be converted through the CONVERT TO 24 command before the LKSYS() function is executed.

```
SET REPROCESS TO 15
ON ERROR DO err_proc
DO invoice
ON ERROR
RETURN
PROCEDURE invoice
BEGIN TRANSACTION
   USE accounts ORDER acct_no
   USE invoices IN 2 ORDER inv_no
   USE inventry IN 3 ORDER item_code
   USE vendors IN 4 ORDER vend_no
   USE shipaddr IN 10 ORDER accr_no NOLOG
   SELECT 1
   Macct_no = acct_no
   SEEK Macct_no
   IF order_time = "Y"
      *
      *
      *
END TRANSACTION
RETURN
PROCEDURE err_proc
choice = " "
DO CASE
   CASE ERROR() = 108
      otheruser = LKSYS(2)
      IF LEN(TRIM(otheruser)) = 0
         otheruser = "another"
      ENDIF
      @ 21,15 SAY "One of the files that you need is in use by " + otheruser
      @ 22,15 SAY "Do you want to try to use it again? (Y/N)" GET choice
      READ
      @ 21,15 CLEAR TO 22,65
      IF UPPER(choice) = "Y"
         ROLLBACK
         RETRY
         DO invoice
      ELSE
         IF COMPLETED()
            @ 21,15 SAY "The file in use by another is not part of a transaction."
            @ 22,15 SAY "Returning to the menu."
            RETURN TO MASTER
         ENDIF
         @ 21,15 SAY "Rolling back your entries. Please wait..."
         ROLLBACK
         IF .NOT. ROLLBACK()
            @ 21,15 SAY "The rollback was not successful. You must restore"
            @ 22,15 SAY "from backup before continuing."
         ENDIF
      ENDIF
      RETURN TO MASTER

   CASE ERROR() = 109
   *** Similar type of program
ENDCASE
RETURN
```

Transaction Log File

A recovery operation is possible because the system maintains a transaction log file of all update operations that occur after a BEGIN TRANSACTION command and before an END TRANSACTION command. A log file contains the value of all updated items before the updates are made. The system uses a transaction log file to restore the database when a ROLLBACK command is executed. A log file contains the record of the last started transaction. A log file does not contain the updates of previous transactions because it is cleared each time a new transaction begins.

The transaction log file is named Translog.log on a single-user system and < workstation name > .log on a multi-user system. The workstation name is the name of the microcomputer that starts the BEGIN TRANSACTION command on the network. On a Novell network, the workstation name is the login ID; on all other networks, it is one of the computer names listed when you enter a DISPLAY USERS command.

When you execute a ROLLBACK command, the transaction functions as a *unit of recovery*. The ROLLBACK command reverses all updates in the transaction, including updates that completed accurately, but did not get written before the system failure.

A transaction is also a *unit of concurrency*, because the transaction log file can be used to reverse concurrent transactions that interfere with each other's operation.

Recovering Database Files After a System Failure

When a system failure occurs, such as that caused by a power loss, the network administrator is normally responsible for recovering the database files for all users that had transactions in progress at the time of the failure. All incomplete transactions must be recovered to ensure that network users will be able to access the database files used by the incomplete transactions.

To recover the database files for all incomplete transactions, follow these steps:

1. Log in to the network as the network administrator.
2. Examine the directories to determine what log files exist.
3. For each user who has a log file, log in to the network as that user, start dBASE IV, and enter the ROLLBACK command at the dot prompt.

Be sure to recover the database file for all users who have a log file.

Finding a log file can be simplified if all log files are stored in the same directory. The < path name > parameter of the BEGIN TRANSACTION command allows you to specify the directory where the log file will be stored; for example:

```
. BEGIN TRANSACTION C:\LOGFILES
```

If all BEGIN TRANSACTION commands specify the same directory, all log files will be stored in one place.

Locking in a Transaction

All file and record locks created in a transaction remain in effect until dBASE IV encounters an END TRANSACTION or ROLLBACK command. The UNLOCK command has no effect in a transaction. Therefore, a transaction should not cover an extended length of time because all locked files and records will not be accessible to other users until the transaction has ended.

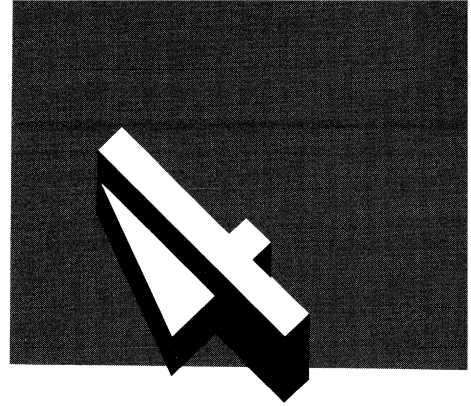
Transaction Processing at the Dot Prompt

Transaction processing commands and functions can be entered at the dot prompt; however, you should be aware of the following considerations:

- Files and records locked during a transaction remain inaccessible for updating by other network users until an END TRANSACTION or ROLLBACK command is executed.
- Error processing works differently at the dot prompt than in an application program.

If you try to update a file or record at the dot prompt, the system attempts to lock the file or record. If the lock attempt fails, the system displays a prompt box saying the file or record is in use and allows you to retry or cancel the attempt. If you choose to cancel the attempt, the system displays a specific message concerning the attempt.

Networking Commands and Functions



About This Chapter

This chapter describes the syntax, defaults, and behavior of each dBASE command and function used in programming for a local area network (LAN) environment.

The chapter has two principal sections: "Using Commands," and "Using Functions." In both sections:

- Some of the commands and functions described are unique to network programming, and some are used in single-user and network programming. If a command or function is used in programming for both single-user and LAN environments, basic features are described in the *Language Reference* manual, and the network programming functionality is described in this section.
- The same syntax notations described in the *Language Reference* manual are used.

Examples clarify the use or syntax of a command or function. When applicable, a list of reference commands and functions that affect the behavior of a command or function is given. Where appropriate, a reference to the *Language Reference* manual is included.

Before reading this chapter, you should be familiar with the commands and functions as described in the *Language Reference* manual.

Using Commands

Each of the commands described in this section falls into a command class.

Table 4-1 lists the various classes.

Table 4-2 summarizes the commands themselves. The table lists commands that have additional syntax or enhanced functionality when used for network programming.

Table 4-1 Command classes

Command Class	Description
Database Security	Introduces security elements at the programming level
Editing of Data	Allows you to edit the data within a database
Locking Control	Ensures data integrity in network applications
Parameter Control	Sets a dBASE system control parameter
Programming	Assists in the control and usage of command files
User Assistance	Provides on-line information
Manipulating Databases	Specifies how .dbf and .ndx files are opened

Table 4-2 Summary of network programming commands

Command	Command Class	Intended Environment	Description
BEGIN TRANSACTION	Editing of Data	Single user and LAN	Initializes the beginning of a transaction
CHANGE/EDIT	Editing of Data	Single user and LAN	Used to alter the contents of a record in the active database file
CONVERT	User Assistance	LAN	Permits use of the SET REFRESH command, the CHANGE() function, and the LKSYS() function
DISPLAY/LIST STATUS	User Assistance	Single user and LAN	Provides information about the current session
DISPLAY/LIST USERS	User Assistance	LAN	Provides information about the currently logged dBASE users
END TRANSACTION	Editing of Data	Single user and LAN	Terminates transaction processing. Used with BEGIN TRANSACTION
LOGOUT	Database Security	Single user and LAN	Forces a user logout and allows a new user to log in

(continued)

Table 4-2 Summary of network programming commands (*continued*)

Command	Command Class	Intended Environment	Description
RESET	Editing of Data	Single user and LAN	Changes the integrity tag in a database file
RETRY	Programming	Single user and LAN	Used to execute a command until it can be completed successfully
ROLLBACK	Manipulating Databases	Single user and LAN	Used with transaction processing to undo changes to a record in a database file
SET	Parameter Control	Single user and LAN	Displays and changes the current values of SET values
SET AUTOSAVE	Database Security	Single user and LAN	Automatically updates the disk file and directories after I/O operations
SET ENCRYPTION	Database Security	LAN	Establishes whether encrypted files are encrypted when copied
SET EXCLUSIVE	Parameter Control	LAN	Determines the file open attribute of all database files subsequently opened during the session
SET LOCK	Locking Control	LAN	Enables and disables automatic locking for read-only commands
SET PRINTER	Parameter Control	Single user and LAN	Redirects printer output to a network or local device
SET REFRESH	Editing of Data	LAN	Updates viewed database information if changed by another user

(*continued*)

Table 4-2 Summary of network programming commands (*continued*)

Command	Command Class	Intended Environment	Description
SET REPROCESS	Parameter Control	LAN	Sets the number of times dBASE IV retries a network command or function before producing an error message
UNLOCK	Locking Control	LAN	Releases record and file locks
USE EXCLUSIVE	Manipulating Databases	LAN	Opens a database file and related files in the selected work area for exclusive use

Before reading about the network programming commands in detail, you should first be aware of how specific commands operate in a LAN environment.

dBASE IV system management governs the commands that update the data stored in a database and associated files. Some of the commands require that the file be opened for exclusive use. Others require that a file or record lock be in place. In the latter case, the system tries automatically to lock the file. This process is not visible to the user, unless the attempt to lock the file fails, in which case the system displays an error message.

Commands Requiring Exclusive Use

You open a file for exclusive use through the SET EXCLUSIVE ON command or the USE <filename> EXCLUSIVE command described in this chapter. The file must have been opened for exclusive use to use the following commands:

```
CONVERT TO
INDEX ON <key> TAG <file.mdx>
INSERT [BLANK]
MODIFY anything
PACK
REINDEX
RESET IN ALIAS
ZAP
```

If a file has not been opened for exclusive use, an attempt to open a file through one of these commands results in the error message **Exclusive open of file is required**.

Files Locked and Released Automatically

When a file is not being used exclusively, commands that work on an entire file first attempt to lock the file before executing the command. If the lock fails, the command is not executed. For the following commands, dBASE IV attempts to lock the file in use before the command is executed:

APPEND FROM
DELETE <scope>
RECALL <scope>
REPLACE <scope>
UPDATE <scope>

When command execution is completed, dBASE IV unlocks the file. If the file can't be locked, dBASE IV displays an error message.

NOTE
Specifying NEXT 1 as the scope for the DELETE, RECALL, and REPLACE commands performs an automatic record lock instead of a file lock.

BEGIN TRANSACTION

BEGIN TRANSACTION starts recording a transaction. It can be used in a program or from the dot prompt.

Syntax

BEGIN TRANSACTION [< path name >]

Usage

Use this command to start a transaction that updates records in one or more database files. An update consists of any of the following: adding a new record, changing the data in a record, or changing the delete status of a record.

A transaction is a unit of work that uses a consistent database, executes a series of updates, and produces another consistent database. While the updates are being made, the database is in an inconsistent state. If any failure occurs during the updating, the transaction can be reversed to the consistent state that existed before the updates started. In this sense, a transaction is also a unit of recovery.

The BEGIN TRANSACTION command creates a transaction log file to track the beginning status of the database file and the changes made during the transaction. The transaction log file is named Translog.log on a single-user system, and < Workstation name > .log on a multi-user system. The workstation name is the name of the microcomputer that starts the BEGIN TRANSACTION command on the network. On a Novell network, the workstation name is the login ID; on all other networks, it is one of the computer names listed when you enter a DISPLAY USERS command.

The transaction log file works with the ROLLBACK command to reverse changes that leave a database in an inconsistent state. Before you issue an END TRANSACTION command, you can reverse the changes made by the transaction. Use the ROLLBACK command to restore all databases to their pre-transaction state.

If the change is satisfactory, or if you have made no changes, issue an END TRANSACTION command. The END TRANSACTION command deletes the transaction log file and clears the integrity tag from the database file.

The BEGIN TRANSACTION and END TRANSACTION commands must be in the same procedure.

A transaction must be complete and the log file deleted before you can start another transaction; transactions cannot be nested. To perform several transactions in succession, bracket each transaction between a BEGIN TRANSACTION and an END TRANSACTION command, as follows:

```
USE Myfile
BEGIN TRANSACTION
   REPLACE ALL Salary WITH Salary * 1.1
END TRANSACTION
LOCATE FOR Last_Name="Cuddigan"
BEGIN TRANSACTION
   EDIT Salary
END TRANSACTION
```

Nested transactions, as shown below, are not permitted.

```
USE Myfile
BEGIN TRANSACTION
   REPLACE ALL Salary WITH Salary * 1.1
   LOCATE FOR Last_Name="Cuddigan"
   BEGIN TRANSACTION
      EDIT Salary
   END TRANSACTION
END TRANSACTION
```

NOTE
dBASE IV commands that result in the creation of a new file are allowed in transactions, but commands that overwrite an existing file are not allowed. Overwriting existing files makes ROLLBACK impossible. If you attempt to use a command that overwrites a file, the following error message appears: ***Cannot execute this command while transaction is in process.***

dBASE IV commands that create new files and that can be used with BEGIN TRANSACTION are:

AVERAGE
COPY [STRUCTURE] TO <newfile> [EXTENDED]
CREATE <newfile> [FROM <filename>]
EXPORT TO <filename>
IMPORT FROM <filename>
INDEX...TO <newfile>
JOIN...TO <newfile>
SET CATALOG TO <newfile>
SORT...TO <newfile>
TOTAL...TO <newfile>

Similarly, dBASE IV commands that close an open file are not allowed during a transaction. These commands are:

CREATE [FROM]
IMPORT
INDEX ON
SET CATALOG
SET INDEX
USE

In addition, the following dBASE IV commands are also not allowed during a transaction:

CLEAR ALL
CLOSE ALL/DATABASE/INDEX
DELETE FILE
ERASE
INSERT
MODIFY STRUCTURE
PACK
RENAME
UNLOCK
ZAP

An attempt to use any of these commands during a transaction results in an error message. The UNLOCK command is a special case. It does not produce an error message and has no effect during a transaction.

NOTE
If you use transaction processing on a network, you cannot have two users logged in to the network operating system with the same name. The first eight characters of any network log-in name must be unique.

See Also

COMPLETED(), END TRANSACTION, ISMARKED(), RESET IN ALIAS, ROLLBACK, ROLLBACK(), SET PATH

CHANGE/EDIT

CHANGE/EDIT are full-screen commands used to alter the contents of a record in the active database file. CHANGE and EDIT are identical commands.

Syntax

CHANGE [< scope >] [FIELDS < field list >] [FOR < condition >]
[WHILE < condition >] [NOFOLLOW] [NOMENU] [NOCLEAR]
[< record number >][NOINIT] [NOAPPEND] [NOEDIT]
[NODELETE]

or

EDIT [< scope >] [FIELDS < field list >] [FOR < condition >]
[WHILE < condition >] [NOFOLLOW] [NOMENU] [NOCLEAR]
[< record number >][NOINIT] [NOAPPEND] [NOEDIT] [NODELETE]

Usage

The CHANGE and EDIT commands will go into APPEND mode if the user attempts to move the cursor past the last record in the file. Also, when called from BROWSE, EDIT will respect all BROWSE command line options with the exception of COMPRESS, LOCK, and WINDOW.

[NOFOLLOW]

The NOFOLLOW option has an effect only on indexed files. If NOFOLLOW is used, editing a key field value repositions the record according to its new key value in the index, but the record that took its original place in the index order becomes the current record.

[NOMENU]

The NOMENU option prevents access to the menu bar.

[NOCLEAR]

When the NOCLEAR option is used, the record will remain on the screen when the user exits EDIT. Also, the memory allocated for the EDIT table and specifications is not cleared. This means that the screen will look exactly as it did when you last exited the EDIT command, including the cursor position.

[NOINIT]

This option is used on the next EDIT command to call up the previously defined EDIT without generating an entirely new structure. If the last EDIT command used the NOCLEAR option, then this command will use the screen specified in the previous EDIT command.

[NOAPPEND]

The NOAPPEND option prevents the user from adding records to the current file when in the EDIT mode.

[NOEDIT]

The NOEDIT option puts the EDIT table in a read-only condition and prevents the user from editing any records when in the EDIT mode. However, additions can be made to database files and records can be marked for deletion and removed permanently from the file with PACK and ZAP.

[NODELETE]

The NODELETE option prevents the user from deleting records (by pressing **Ctrl-U**) of the current file when in the EDIT mode.

< record number > starts the edit on the specified record, but lets you move to other records in the file. You may also use the keyword RECORD, which is one of the options of < scope > . If you use the scope keyword RECORD, however, EDIT is limited to one record, and does not allow you to move to other records in the file. Because EDIT RECORD < record number > limits the edit to the specified record, EDIT RECORD < record number > and EDIT < record number > are not identical.

In a LAN environment, the fourth section of the status bar displays the following file or record information, as appropriate:

Exclusive use
File locked
Read only
Record locked
Record unlocked

The CHANGE and EDIT commands attempt to lock the record and all related records when you press any key that tries to update that record, that is, any key other than a navigation key. If the lock is successful, the system determines if any of the fields have changed since the user accessed the record.

If any information has changed since the record was displayed on the screen, the screen reflects the changes and the system displays the message **Data in record has changed. Press Esc to abandon and any other key to continue**. You can continue or abort. If you press **Esc**, the edit attempt is abandoned; if NOCLEAR is specified, the record stays on the screen and the record is unlocked. Pressing any other key allows you to continue with the edit of the record. Note that you have pressed two keys at this point, both of which count as part of the edit.

If none of the information has changed, you are able to enter information into the record.

Example

```
. USE TEST
. EDIT NOCLEAR
* edit record Ctrl-W
. DISPLAY STRUCTURE
. EDIT NOINIT
* old record comes back
```

CONVERT

CONVERT permits subsequent use of the CHANGE() and LKSYS() functions and the SET REFRESH command. CONVERT also enables automatic change detection for BROWSE and EDIT commands.

Syntax

CONVERT [TO <expN>]

where <expN> is the length of the field. The default is 16, and the field can range in length from 8 to 24 characters. Using MODIFY STRUCTURE, you can alter the length of the command up to a maximum of 24 characters.

Usage

The CONVERT command adds a field to the structure of the active database file. The field name is _DBASELOCK.

If you specify 8 for <expN>, an 8-character _DBASELOCK field is added. You can then use the SET REFRESH command for BROWSE and EDIT, the CHANGE() function to see if a record has been changed, and the LKSYS() function to determine the date and time a record was locked. If you specify 16 for <expN>, you can do all of the above and also use the LKSYS() function to determine the first 8 characters of the user ID that locked a record. If you specify 24, the user ID returned by the LKSYS() function is 16 characters long.

If you specify 8 for <expN>, the system does not save the user ID, date, and time, but you can use the SET REFRESH command and the CHANGE() function. Specifying 8 produces a smaller file and prohibits the display of user IDs.

When you enter the CONVERT command, the current database file is copied to a .cvt file before the conversion, and a new .dbf file is created.

The previous SET FIELDS and SET RELATION that are related to the database file are cancelled when you use the CONVERT command.

Example

```
. USE employee
. CONVERT
```

See Also

CHANGE(), LKSYS(), MODIFY STRUCTURE

DISPLAY/LIST STATUS

DISPLAY STATUS and LIST STATUS provide information about the current dBASE IV session.

Syntax

LIST STATUS [TO PRINTER/TO FILE <filename>]
DISPLAY STATUS [TO PRINTER/TO FILE <filename>]

Usage

This command is the same in single-user and LAN environments. In a LAN environment, however, the lock status for each open database file and a listing of the currently locked records in each database file (if any) are added to the single-user information displayed.

The single-user information displayed is:

- Database name
- Work area number
- Alias
- Database relations
- Open index filenames
- Index key for each open index file
- Current file search path
- Default disk drive
- Current settings for most ON/OFF SET commands (in a LAN environment, the display includes settings for ENCRYPTION and EXCLUSIVE)
- The setting for DEVICE (SCREEN or PRINT)
- Left margin setting
- Function key and programmable key assignments

If TO PRINTER is specified, the STATUS display is directed to the currently defined printer. The currently defined printer is either the selected network printer, or a local printer defined through the SET PRINTER command (described later in this chapter).

If TO FILE is specified, the STATUS display is directed to the specified file.

DISPLAY/LIST STATUS

Programming Note

LIST/DISPLAY STATUS can be useful when debugging an application program; for example, ON ESCAPE DISPLAY STATUS.

Example

This is a sample STATUS display as it might appear in a LAN environment (several screens have been combined for illustration purposes):

```
. DISPLAY STATUS

Currently Selected Database:
Select area:  1, Database in Use:  H:MYFILE.DBF    Alias:  MYFILE
          Memo file:   H:MYFILE.DBT
          Lock list:       2,      4,       6,      8  locked


Select area:  2, Database in Use:  H:ACCTG.DBF   Alias:  ACCTG
          Memo file:   H:ACCTG.DBT
          Lock list:   database locked


File search path:
Default disk drive:  H:
Print destination:  PRN:
Margin =     0
Refresh count =    0
Reprocess count =    0
Number of files open =    4
Current work area:  1

ALTERNATE  - OFF    DELIMITERS  - OFF    FULLPATH   - OFF    SAFETY      - OFF
AUTOSAVE   - OFF    DESIGN      - ON     HEADING    - OFF    SCOREBOARD  - ON
BELL       - ON     DEVELOP     - ON     HELP       - ON     SPACE       - ON
CARRY      - OFF    DEVICE      - SCRN   HISTORY    - ON     SQL         - ON
CATALOG    - OFF    ECHO        - OFF    INSTRUCT   - ON     STATUS      - OFF
CENTURY    - OFF    ENCRYPTION  - OFF    INTENSITY  - ON     STEP        - OFF
CONFIRM    - OFF    ESCAPE      - ON     LOCK       - ON     TALK        - OFF
CONSOLE    - ON     EXACT       - ON     NEAR       - ON     TITLE       - ON
DEBUG      - OFF    EXCLUSIVE   - OFF    PAUSE      - ON     TRAP        - ON
DELETED    - OFF    FIELDS      - OFF    PRINT      - OFF    UNIQUE      - OFF
```

```
Programmable function keys:
F2          - assist;
F3          - list;
F4          - dir;
F5          - display structure;
F6          - display status;
F7          - display memory;
F8          - display;
F9          - append;
F10         - edit;
CTRL-F1     -
CTRL-F2     -
CTRL-F3     -
CTRL-F4     -
CTRL-F5     -
CTRL-F6     -
CTRL-F7     -
CTRL-F8     -
CTRL-F9     -
CTRL-F10    -
SHIFT-F1    -
SHIFT-F2    -
SHIFT-F3    -
SHIFT-F4    -
SHIFT-F5    -
SHIFT-F6    -
```

DISPLAY/LIST USERS

DISPLAY/LIST USERS identifies the workstations currently logged in to dBASE IV.

Syntax

DISPLAY USERS
LIST USERS

Usage

DISPLAY/LIST USERS displays the network-assigned computer machine (workstation) names of currently logged dBASE network users.

WARNING
This command, or the network equivalent of it, should always be issued to determine whether any users are using dBASE IV before dBASE IV is uninstalled.

Example

The following is a sample USERS display:

```
. DISPLAY USERS

Computer name
-------------

 WKSTN1
 WKSTN4
 WKSTN3
>WKSTN2
 WKSTN7
```

The character > marks the currently logged user.

END TRANSACTION

END TRANSACTION terminates a transaction to change a record in a database file.

Syntax

END TRANSACTION

Usage

This command commits the active transaction, and deletes the transaction log file. The integrity tag placed in the header of the database file involved in the transaction is removed.

You should not use this command until you are certain that the transaction has done what you wanted it to do. Once you issue an END TRANSACTION command and the transaction log file is erased, the only way to restore database files to their pre-transaction state is to restore backup files.

The BEGIN TRANSACTION and END TRANSACTION commands must be in the same procedure.

You must use this command to terminate a transaction before starting another. dBASE IV does not allow transaction nesting. If an END TRANSACTION command is not executed, the transaction log file is closed when the program exits.

This command releases all locks created during a transaction.

Example

See the BEGIN TRANSACTION command.

See Also

BEGIN TRANSACTION, COMPLETED(), ISMARKED(), RESET, ROLLBACK, ROLLBACK()

LOGOUT

LOGOUT logs out the current user and allows a new user to log in.

Syntax

LOGOUT

Usage

This command enables you to control user sign-in and sign-out procedures. It forces a logout and prompts for a login.

When the LOGOUT command is processed, the workstation screen clears and a log-in screen appears in which the user enters a group name, log-in name, and password. The PROTECT command establishes log-in verification functions and sets the user access level.

LOGOUT closes all open database files, their associated files, and program files.

You can also use this command if you need to log in to a different group to access a different set of database files.

Special Cases

If PROTECT has not been used, no Dbsystem.db file is created and the LOGOUT command returns to the dot prompt.

See Also

Chapter 5, "dBASE Security"

RESET resets the integrity tag in a database file.

Syntax

RESET [IN <alias>]

where <alias> is an alias name (which may be the same as its filename), a work area letter, a work area number, or an indirect reference.

Usage

The *integrity tag* marks files that have a transaction in progress. This tag stays on a file until the transaction is completed or a successful ROLLBACK has occurred. If an <alias> is not specified, the current database file is reset.

You may need to remove this tag with the RESET command if you do not want to ROLLBACK a database file, or if a successful ROLLBACK is not possible.

After issuing a RESET command, you can access the file for another transaction.

RESET should not be used in a program. Use this command at the dot prompt, when necessary, to correct an unusual situation.

This command requires exclusive use of the database file.

Example

```
. RESET IN WORK3
```

See Also

BEGIN TRANSACTION, END TRANSACTION, ISMARKED(), ROLLBACK, ROLLBACK(), UNLOCK

RETRY

RETRY closes the program or procedure file in which the command was issued. It then restores control to the program or procedure that called it, at the same line that called the program.

Syntax

RETRY

Usage

This command returns to the program that called the present program. Then it re-executes the line that called the program. RETRY is normally used in an ON ERROR procedure.

The RETRY command is used in single-user and network programming to execute a dBASE program repeatedly until a task is completed. In network programming, you use the RETRY command in error recovery situations to retry execution of a command until it can be completed successfully.

Programming Notes

RETRY differs from RETURN in that RETRY executes the same line in the calling program, while RETURN executes the next line.

Examples

See the example in the description of the ERROR() function and the more extended example in Chapter 2.

See Also

ERROR(), MESSAGE(), SET REPROCESSS

The section "Errors and Error Recovery" in Chapter 2

ROLLBACK

ROLLBACK restores all changes made to records in a database file since a BEGIN TRANSACTION command was issued.

Syntax

ROLLBACK [< database filename >]

Usage

This command restores database and index files to the state they were in before the current transaction began. It closes and deletes all database and index files that may have been created during the transaction.

While a transaction is active, you can use ROLLBACK only on the database files that are currently active in the transaction. You cannot use a database filename argument while a transaction is active.

ROLLBACK compares the transaction log file with the active database contents, to reverse the active transaction and restore the database file to the condition it was in at the beginning of the transaction.

Once a transaction is terminated, you can use the ROLLBACK command only with a database filename argument. In this case, all ROLLBACK does is clear the file from being flagged as "in use." You will have to restore the file to an unchanged state manually from backup copies or hard copy records.

A transaction log file contains the pre- and post-transaction contents of each record that is altered. The ROLLBACK command refers to the transaction log file to reverse transactions. ROLLBACK can fail under two circumstances:

1. If there are inconsistencies between a record's pre- and post-transaction contents, or
2. If the transaction log file is not readable.

At the end of a successful ROLLBACK procedure, the database file is restored to its pre-transaction status. The transaction log file is reset to the beginning and remains open waiting for a new transaction.

If you enter a SUSPEND command during a transaction, the transaction remains active. Commands that you enter during the suspended mode are considered part of the transaction, and it is acceptable to enter a ROLLBACK command during the suspended mode. If a ROLLBACK command is entered during a suspended mode, you are not allowed to RESUME the suspended program, because it would restart in the middle of a transaction. After a ROLLBACK, the RESUME command executes the next line of code following the END TRANSACTION command.

Examples

To restore the database to the beginning status during a transaction and verify that the ROLLBACK was successful:

```
. ROLLBACK
10 Records checked
```

After a failed transaction, to do a manual ROLLBACK on a database file:

```
. ROLLBACK Name
```

Also see the example in BEGIN TRANSACTION.

See Also

BEGIN TRANSACTION, COMPLETED(), END TRANSACTION, ISMARKED(), RESET, ROLLBACK()

SET is a full-screen, menu-driven command that displays SET commands and their current values. It also allows you to change the value for any setting.

Syntax

SET

Usage

This command allows you to view and change:

- ON/OFF SET commands, including those for network programming
- Screen attributes
- Function key assignments
- Default drive and search path
- Alternate, format, and index files
- Left margin
- Decimal places

In a LAN environment, ENCRYPTION and EXCLUSIVE are added to the list of SET commands that can be set ON/OFF.

Examples

The **Options** submenu on the **SET** menu bar that appears in a LAN environment contains the following settable commands:

ALTERNATE	DELIMITERS	HISTORY	SAFETY
AUTOSAVE	DEVICE	HOURS	SCOREBOARD
BELL	DOHISTORY	INSTRUCT	SPACE
CARRY	ENCRYPTION	INTENSITY	STATUS
CATALOG	ESCAPE	LOCK	TALK
CENTURY	EXACT	MARGIN	TITLE
CONFIRM	EXCLUSIVE	MEMOWIDTH	TRAP
CURRENCY	FIELDS	MENU	UNIQUE
DATE	FIXED	NEAR	
DECIMALS	HEADING	PRECISION	
DELETED	HELP	PRINT	

If you select **Settings** from the Control Center **Tools** menu, a subset of the above option list appears.

SET AUTOSAVE

SET AUTOSAVE activates or deactivates the feature that updates the disk file and directories after you update a single record or a group of records.

Syntax

SET AUTOSAVE on/OFF

Defaults

SET AUTOSAVE defaults to OFF.

Usage

The SET AUTOSAVE ON command reduces the chance of data loss in the event of a power loss or other catastrophic event. However, use of this command slows performance.

If AUTOSAVE is set ON, then every time a command that updates records completes its operation, the directory and file allocation table are updated to disk.

Commands that update groups of database records, such as REPLACE, SORT, and TOTAL, perform the AUTOSAVE operation after they have finished processing the records. Data entry commands (APPEND, INSERT) and editing commands (BROWSE, EDIT, MODIFY COMMAND/FILE) perform this operation after each record is updated.

Example

```
SET AUTOSAVE ON
```

See Also

Chapter 6, "Customizing dBASE IV," in the *Language Reference* manual

SET ENCRYPTION

SET ENCRYPTION establishes whether encrypted files are encrypted when copied.

Syntax

SET ENCRYPTION on/OFF

Defaults

SET ENCRYPTION is normally OFF.

Usage

The SET ENCRYPTION command determines whether copied files (that is, files created through the COPY, JOIN, and TOTAL commands) are created as encrypted files. The command avoids the overhead involved in encrypting/decrypting files when it's not necessary. Note that the original file must be encrypted.

Memo (.dbt) files associated with the database file are encrypted.

An encrypted file contains data enciphered into another form to hide the contents of the original file. An encrypted file can only be read after the encryption has been deciphered or copied to another file in decrypted form.

If you enter a valid user name, password, and group name at the log-in screen, dBASE IV will open an encrypted file.

With SET ENCRYPTION OFF, a file can be copied in a decrypted form (see the "Examples" section):

- The authorization/access levels of the user limit the information that will be copied. That is, a copy operation will be limited by any applicable file and field access level controls, and by field access privileges as well.
- The COPY TO options, EXPORT, FIELDS, FOR, WHILE, TYPE DELIMITED, WITH BLANK, DIF, SDF, SYLK, and WKS, can be specified only if SET ENCRYPTION is OFF.
- You need to do this if you wish to EXPORT the file or use it with single-user dBASE IV.

NOTE
Encryption works only with PROTECT. If you do not enter dBASE IV through the log-in screen, you will not be able to use encrypted files.

SET ENCRYPTION

Tips

All encrypted files used in an application must have the same group name.

Encrypted files cannot be JOINed with unencrypted files. Make both files either encrypted or unencrypted before JOINing them.

Encrypted files must be copied to files in decrypted form before they can be used with the COPY STRUCTURE EXTENDED command. (The examples below illustrate how to create a decrypted version of an encrypted file.) After issuing the command, you can encrypt the new file with PROTECT by assigning it an access level.

Files created through the CREATE command are encrypted through the PROTECT command by assigning the files an access level.

Examples

You cannot use the MODIFY STRUCTURE command (see the *Language Reference* manual) to modify encrypted files. Use the following commands to create a decrypted version first:

```
USE <filename 1>                    && Identifies an encrypted
SET ENCRYPTION OFF                  && database file or db text file

COPY TO <filename 2>                && Creates a decrypted
USE <filename 2>                    && version of the file

MODIFY STRUCTURE <filename 2>
COPY TO <filename 1>
CLOSE ALL
DELETE <filename 2>
PROTECT
```

The following example sets up an on-line environment in which a user is prompted to enter the name of an encrypted file to be made available as a decrypted file with the COPY command. The example assumes the user has successfully logged in through PROTECT and belongs to a group that can decipher the named file.

```
In_file=SPACE(8)
DO WHILE LEN(In_file)>0
  ACCEPT "Enter filename to convert" TO In_file
  IF LEN(In_file)=0
     EXIT
  ENDIF
  In_file=TRIM(In_file)+".dbf"
  ACCEPT "Enter file to copy to" TO Out_file
  IF .NOT. FILE("&In_file")
     LOOP
  ENDIF
  SET ENCRYPTION OFF
  USE &In_file
  COPY TO &Out_file
ENDDO
```

See Also

Chapter 5 for more information on encryption

SET EXCLUSIVE

SET EXCLUSIVE determines the file open attribute of all database files USEd (after entering the command) during the dBASE IV session.

Syntax

SET EXCLUSIVE on/OFF

Defaults

The default mode for dBASE IV is SET EXCLUSIVE OFF. By system default, all opened files can be shared. As long as there is no need for simultaneous access by multiple users, there is no concern about locking database file resources.

Usage

When set to ON, all files opened after entering the command are for the sole use of the network user. No other users are able to open the database file. Until the user closes a file, no other network user has any access to the file.

When set to OFF, all files USEd after entering the command are opened for shared use. If the file is to be accessed simultaneously by multiple users, SET EXCLUSIVE should be set OFF.

Tips

If you open a .dbf file with SET EXCLUSIVE OFF, the file open attribute is shared and the file access attribute is read/write. If you wish the file to be read-only, use the DOS command ATTRIB to set the read-only flag (on a Novell network, use the FLAG command).

However, if you open a .dbf file with SET EXCLUSIVE ON, the file open attribute is exclusive and the file access attribute is read/write. Use the DOS command ATTRIB or PROTECT to change the file access (on a Novell network, use the FLAG command).

Examples

Assume that dBASE IV is automatically called with SET EXCLUSIVE ON. The following sequence of commands opens the EVRY_ONE database files for shared use and the MY_FILE database files for exclusive use:

```
SET EXCLUSIVE OFF              && Change the file open
                               && mode to shared.

SELECT 1                       && Select work area 1, and
USE Evry_one INDEX Evry_One    && open the Evry_one
                               && files for shared use.

SELECT 2                       && Select work area 2.
SET EXCLUSIVE ON               && Change the file open
                               && mode to exclusive.

USE My_file INDEX My_file      && Place the My_file
                               && files in work area 2.
```

See Also

USE EXCLUSIVE

Chapter 2 for a discussion of the file open and file access attributes

The DOS command ATTRIB (on a Novell network, the FLAG command)

SET LOCK

SET LOCK enables and disables automatic locking for the following commands: AVERAGE, SUM, CALCULATE, COPY [STRUCTURE], COUNT, INDEX, JOIN, REPORT, LABEL, TOTAL, and SORT.

Syntax

SET LOCK ON/off

Defaults

The default for the SET LOCK command is ON. This default can be changed in the Config.db file by using the command:

LOCK = OFF

Usage

The SET LOCK command affects only a subset of the commands that are automatically locked, specifically those commands that do not modify data values. Although these commands will work without locking, data integrity is not guaranteed. If you use SET LOCK OFF, there will be greater concurrency, but also a greater risk of possible inconsistent analysis.

Some of the commands which work with SET LOCK have two phases: reading and writing. SET LOCK affects only the reading phase. When writing, the file is opened for exclusive use (for example, COPY [STRUCTURE], INDEX, JOIN, TOTAL, and SORT).

SET PRINTER

SET PRINTER redirects dBASE IV printer output to a network or local device. You use this command to redirect print output to a local printing device or a shared network printer.

Syntax

Form 1: Sending Output to a Network Printer

IBM PC, IBM Token-Ring, or Ungermann-Bass networks

```
. SET PRINTER TO \\<computer name>\<printer name>=<destination>
```

or

Novell

Use the Novell SPOOL or CAPTURE command prior to starting dBASE IV:

```
F > SPOOL L = 2 TI = 5
F > DBASE
```

Spool LPT2: with the timeout option and redirect to the network printer:

```
. SET PRINTER TO LPT2
```

Form 2: Sending Output to a Local Device

IBM PC, IBM Token-Ring, or Ungermann-Bass networks

```
. SET PRINTER TO <destination>
```

or

Novell

```
F > SPOOL L = 2 TI = 5
F > DBASE
```

```
. SET PRINTER TO LPT1
```

Form 3: Emptying the Print Spool File; Resetting the Default Destination

```
. SET PRINTER TO <printer or device name>
```

Defaults

By default, output is spooled to the printer assigned to the DOS PRN device.

SET PRINTER

Usage

The SET PRINTER command changes the destination of printed output.

Form 1 of the command spools printer output to a network printer.

\\ < computer name > \ < printer name > is used to identify an IBM network printer. SPOOLER is used to identify a Novell network printer.

< computer name > is a network-assigned server name. It must be a unique identifier for the server, and therefore can't be a group name.

< printer name > is a network-assigned printer name. Note that < computer name > and < printer name > are assigned via the network shell.

< destination > identifies the installed printer: LPT1, LPT2, or LPT3, as appropriate. LPT must be specified. The installed printer can be the shared network printer defined at the logged user's workstation, or it can be a shared printer at another remote workstation. For example:

```
. SET PRINTER TO \SERVERPSON=LPT1
```

Form 2 of the command establishes a DOS device as the destination for printer output. < destination > can be the parallel line printers (LPT1 or PRN, LPT2, LPT3) or the COM devices (COM1, COM2). SET PRINTER TO LPTx/COMx initiates spooled output to the assigned DOS device. For example:

```
. SET PRINTER TO LPT2
```

Form 3 of the command empties the print spooling file and resets the printer destination to its default (the currently defined shared network printer) or to another destination. For example:

```
. SET PRINTER TO LPT2
```

Special Cases

If the local printer has been redirected, use Form 2 of the command to send printer output to the local device.

Programming Notes

Printing of spooled files begins only when the SET PRINTER command is issued (see the *Language Reference* manual). Many files can be collected for printing before print files begin to print. Once printing begins, it can be cancelled only at the network shell level. You can use the RUN command to return to the network shell from within dBASE IV, but only if your computer is configured with enough memory.

Examples

The following commands spool the output of report CENSUS to the network printer (parallel printer number 1) attached to the server named ASERVER:

```
. SET PRINTER TO \\ASERVER\PRINTER=LPT1
. REPORT FORM CENSUS TO PRINT
. SET PRINTER TO LPT1
```

The first command redirects LPT1 to the network printer; the second command sends the output to the spooler; the third command sends the output to the printer and resets the network printer.

The following commands will set up a local printer as the printer output destination. Note that you must disconnect the network printer via the RUN/! command before resetting the printer device to the local printer. For example, on a Novell network:

```
. RUN ENDSPOOL
. SET PRINTER TO <destination>
. REPORT FORM CENSUS TO PRINT
```

NOTE
If your workstation has more than one parallel port, you can use one to access the network printer and another to access a local printer. With this method, you do not need to specify the RUN/! command.

SET REFRESH

SET REFRESH updates the display of the database in BROWSE/EDIT at a specified interval to show changes made by other users.

Syntax

SET REFRESH TO <expN>

where the value of <expN> is from 1 to 3,600.

Defaults

The default is 0, which means the display is not updated.

Usage

The SET REFRESH command only works in BROWSE, EDIT, and CHANGE.

This command must either be set in Config.db using REFRESH = , or must be executed before you start the BROWSE/EDIT session. It is in effect for the current work session or until it encounters another SET REFRESH command.

The command works differently in BROWSE than in EDIT. In BROWSE, when the refresh count reaches zero, dBASE IV checks the information on all records in the network that are currently viewed in the BROWSE table and replaces the old screen with a new screen containing updated information.

Examples

The following example allows you to view the Employee database with the knowledge that the data will be updated and shown on your screen every five seconds, if anyone changes the data:

```
USE employee
CONVERT
SET REFRESH 5
```

The converted file now has a .dbt extension and is already in use. Alternatively, you can use the SET REFRESH command while using the EDIT command instead of BROWSE. If you use the SET REFRESH command in EDIT, you must have the status line on.

See Also

BROWSE, CONVERT, EDIT

SET REPROCESS

SET REPROCESS sets the number of times dBASE IV retries a command or function that causes a network error before producing an error message.

Syntax

SET REPROCESS TO <expN>

Usage

Use this command in a local area network to change the command execution process. A record or a file may be in use on the network, and several retries may be required to access it.

You can request from 1 to 32,000 retries with <expN>, or you can specify −1 to continuously retry a command until execution occurs. If you specify −1 and another network user has locked a file or record you need to access, the system will appear to hang. In such a situation, the user who has locked the file or record must release it.

If you set the numeric value to 0, with no ON ERROR command, dBASE IV displays a **Please Wait, another user has locked this record or file** while it tries an infinite number of times to lock the record or file. You can use the **Esc** key to interrupt the program from retrying endlessly if you prefer.

If you do not use SET REPROCESS, the system does not display the name of the user. The system says that the file or record is in use by another.

Examples

This example shows that after the first 20 tries, the record was not successfully locked. Selecting the **Retry** option from the box resulted in a successful lock, and the function returned a logical true (.T.):

```
SET REPROCESS TO 20              && 20 retries will be made, if a file is locked
USE EMPLOYEE
* for a simple example 1:
* to edit record 32
GO 32
* for a simple example 2:
* update records for a particular employee
REPLACE salary WITH salary * 1.1 FOR empl_num = "1001"
```

See Also

RLOCK()/LOCK(), UNLOCK, USE, USE EXCLUSIVE

UNLOCK

UNLOCK releases record and file locks so that other users can access the data.

Syntax

UNLOCK [ALL/IN < alias >]

where < alias > is an alias name (which may be the same as its filename), a work area letter, a work area number, or an indirect reference.

Usage

Once locked, a record or file can't be modified by another user until the UNLOCK command has been executed to unlock the record or file.

The UNLOCK command unlocks the current record or the current file in the selected work area. UNLOCK ALL releases all current locks in all work areas. The current lock is the last lock function issued in the selected work area.

This command releases the last record lock in the selected work area if RLOCK() or LOCK() is the last lock function issued in the selected work area. This command releases the last file lock if FLOCK() is the last lock function issued in the selected work area.

If you lock a record or file while it is actively related to other open files, then all the files in the relation are automatically locked. Unlocking the record or file automatically unlocks all the related files.

If you specify an incorrect < alias > , the error message **Alias not found** appears.

The UNLOCK command has no effect if a transaction is in process. The END TRANSACTION command releases all locks created during the transaction.

Examples

The following example illustrates using the RLOCK() function to lock a record and the UNLOCK command to unlock it. (The "Examples" section for the FLOCK() function includes a program use of the UNLOCK command.)

```
. USE employee
. ? RLOCK("5,7","employee")
.T.
. GOTO 5
EMPLOYEE: Record No 5
. REPLACE Name WITH "NELSON"
1 record replaced
. GOTO 7
EMPLOYEE: Record No 7
. REPLACE Name WITH "LAMBERT"
1 record replaced
. UNLOCK
```

See Also

FLOCK(), RLOCK()/LOCK()

The section "Using an Explicit Lock Function Interactively" in Chapter 2

USE EXCLUSIVE

USE EXCLUSIVE opens a database and related files in the selected work area for exclusive use.

Syntax

USE [< database filename >] [INDEX < .ndx or .mdx file list >]
EXCLUSIVE [ALIAS < alias >]

where < alias > is an alias name (which may be the same as its filename), a work area letter, a work area number, or an indirect reference.

Defaults

USE, without any parameters, closes the active database and index files in the currently selected work area.

Unless otherwise specified, dBASE IV assumes a .dbf extension for the file name, a .dbt extension for the memo file, and an .ndx extension for the index files.

Usage

The USE EXCLUSIVE command opens a file with a file open attribute of exclusive and a file access attribute of read/write. If the file specified is already open, it is closed and reopened in exclusive mode.

The reserved word EXCLUSIVE can be specified either at the end of the command or before or after the filename parameter.

Record Pointer

If an index file is not in use when the USE EXCLUSIVE command is issued, the record pointer is positioned at record 1 of the database file. If an index file is in use, the record pointer is at the logical beginning-of-file.

Special Cases

The following commands require exclusive use of files:

INSERT [BLANK]
MODIFY any option
PACK
REINDEX
RESET
ZAP

Examples

The following example shows how the EXCLUSIVE parameter can be added to the USE command to select a file for exclusive use:

```
SET EXCLUSIVE OFF
SELECT A
USE Myfile
SELECT B
USE New_file EXCLUSIVE
```

Myfile is opened in share, read/write mode. New_file is opened in exclusive, read/write mode. The following code is equivalent:

```
SET EXCLUSIVE OFF
SELECT A
USE Myfile
SET EXCLUSIVE ON
SELECT B
USE New_file
```

See Also

SET EXCLUSIVE

Using Functions

Functions perform specialized operations that augment and enhance the dBASE commands. Functions always return a value and must be used as expressions or in expressions following a command verb.

Table 4-3 lists the functions unique to or enhanced for network programming. The table lists the functions alphabetically, briefly describes each one, and shows the type of the value returned by the function.

Table 4-3 Network programming function summary

Function Name	Description	Output Data Type
ACCESS()	Returns a user access level	N
CHANGE()	Determines whether a record has been changed *	L
COMPLETED()	Determines if a transaction has been completed	L
ERROR()	Returns the error number	N
FLOCK()	Attempts to lock a database file and returns a success value *	L
ISMARKED()	Determines if a transaction is in progress	L
LKSYS()	Determines who has locked a record or file and the date and time it was locked *	C
MESSAGE()	Returns an error message character string	C
NETWORK()	Determines if the network version of dBASE IV is running	L
RLOCK()/LOCK()	Attempts to lock a record and return a success value *	L
ROLLBACK()	Determines whether the last ROLLBACK command was successful	L
USER()	Returns the name of the user logged in to a protected system	C

* Intended for use in network programming only. Functions that do not have an asterisk following their descriptions are available in single-user and network programming.

ACCESS()

ACCESS() returns the access level of the last logged-in user, as it was set in PROTECT.

Syntax

ACCESS()

Usage

The ACCESS() function allows you to build security in a network application program. The access level returned by this function can be used as a test to control program command sequence.

Programming Notes

Using the ACCESS() function, you can easily determine if a user has authority to proceed with a module in a program. The following example tests the user access level to determine if it is less than 3:

```
IF ACCESS() < 3
```

You may not want anyone to be able to modify access controls implemented through the ACCESS() function. Use the COMPILE command to create a .dbo file, save the .prg file to a diskette, and then delete the .prg file from the hard disk.

Special Cases

If dBASE IV does not find the Dbsystem .db file during startup, it does not present the log-in screen to the user. ACCESS() returns a zero if the user has not entered dBASE IV through the log-in screen. A user with an access level of zero cannot access encrypted files. To prevent ACCESS() from returning a zero, it is recommended that you keep the Dbsystem.db file in the same directory as dBASE IV, and always enter through the log-in screen.

If you write programs that use encrypted files, check the user's access level early in the program. If ACCESS() returns a zero, your program might prompt the user to log in again, or to contact the Network Administrator for assistance.

In a single-user environment, ACCESS() always returns a zero.

ACCESS()

Example

In the following example, the user's ability to execute the code is determined by testing the user's access level with the ACCESS() function:

```
DO WHILE .T.
? "1) Enter chart of accounts"
*
*     && More menu options
*
ACCEPT "Enter choice " TO Choice
   DO CASE
      CASE Choice="1"
         IF ACCESS()<3
            DO Program
         ELSE
            ? "ACCESS NOT ALLOWED"
            WAIT
            LOOP
         ENDIF
      CASE choice="2"
         *
         *
         *
   ENDCASE
ENDDO
```

See Also

FLOCK(), PROTECT(), RLOCK(), UNLOCK(), USER()

CHANGE() allows you to determine whether a record has been changed since it was opened.

Syntax

CHANGE([<alias>])

where <alias> is an alias name (which may be the same as its filename), a work area letter, a work area number, or an indirect reference.

Usage

The CHANGE() function checks the field added by the CONVERT command, to see whether the current record has been changed. The CHANGE() function returns a logical true (.T.) if the current record has been changed, or a logical false (.F.) if the current record has not been changed.

You must issue the CONVERT command before you use CHANGE(). If you attempt to use CHANGE() before issuing the CONVERT command, the system returns .F.

Example

```
USE employee
DO WHILE .NOT. EOF()
   STORE LASTNAME TO M_LASTNAME
   STORE SALARY TO M_SALARY
   @ 5,5 SAY "LASTNAME"
   @ 6,5 SAY "SALARY"
   @ 6,5 SAY "PRESS ANY KEY TO VIEW NEXT RECORD"
   @ 5,16 GET M_LASTNAME
   @ 6,16 GET M_SALARY
   CLEAR GETS
   I=0
   DO WHILE I<>0
      I=INKEY()
      IF CHANGE()
         GO RECNO()
         STORE LASTNAME TO M_LASTNAME
         STORE SALARY TO M_SALARY
         @ 5,16 GET M_LASTNAME
         @ 6,16 GET M_SALARY
         CLEAR GETS
      ENDIF
   ENDDO
   SKIP
ENDDO
```

CHANGE()

See Also

CONVERT, LKSYS(), RLOCK()

COMPLETED()

COMPLETED() allows you to determine whether a transaction has ended.

Syntax

COMPLETED()

Usage

The COMPLETED() function returns a logical true (.T.) if a transaction is not in process or a logical false (.F.) if a BEGIN TRANSACTION command has been executed. The value of the function is reset to a logical true when an END TRANSACTION command is executed or when a ROLLBACK is completed.

Example

The following example checks to see if a transaction has finished and displays a ****** TRANSACTION SUCCESSFULLY COMPLETED ****** message.

```
USE employee TAG status
BEGIN TRANSACTION
REPLACE ALL salary WITH salary * 1.1 FOR department = "SALES"
END TRANSACTION
IF COMPLETED()
   * when true, indicates a completed transaction session
   clear
   @ 10,20 SAY "**** TRANSACTION SUCCESSFULLY COMPLETED ****"
   @ 12,20 SAY "**** SALARIES HAVE BEEN INCREASED ****"
ENDIF
```

See Also

BEGIN TRANSACTION, END TRANSACTION, RESET, ROLLBACK, ROLLBACK()

ERROR()

ERROR() returns the number of an error.

Syntax

ERROR()

Usage

The ERROR() function returns the error code of the condition that caused an error. ERROR() is used within an error processing routine; it can only be used when ON ERROR is being used to trap errors. In network programming, you use the ERROR() function to trap recoverable error conditions, such as attempts to lock an already locked file or record. The value returned by ERROR() is numeric.

Programming Notes

ON ERROR must be active for this function to receive a value. If RETRY or RETURN is used within the ON ERROR procedure, the ERROR() value is replaced by zeroes when either of these commands is processed.

Tips

The MESSAGE() function obtains the dBASE error message.

The CANCEL command terminates execution of an application program when the error is not recoverable. In debug mode, the SUSPEND command terminates execution.

Example

The following example uses the value returned by the ERROR() function to establish the condition for execution of commands and passes the value of ERROR() to a procedure:

```
*********************************************************
* Error Program                                         *
*********************************************************

DO CASE
CASE ERROR() = 108                  && File is in use by another error
   Time = 1
   DO WHILE Time < 100              && Delay loop
      Time = Time + 1
   ENDDO
   IF Trys > 10                     && Trys is a PUBLIC variable
      Trys = Trys + 1
      RETRY                         && Retry opening file
   ELSE
      ? "File is locked - can't proceed"
      WAIT
      RETURN TO MASTER
   ENDIF
CASE ERROR()=25
   *
   *
   *
ENDCASE
```

See Also

MESSAGE(), RETRY

CANCEL, DO CASE, ON ERROR, RETURN, and SUSPEND in the *Language Reference* manual

The section "Errors and Error Recovery" in Chapter 2

The complete list of error messages and codes in the *Language Reference* manual (Appendix A)

FLOCK()

FLOCK() attempts to lock a database file and returns a .T. if successful.

Syntax

FLOCK([< alias >])

where < alias > is an alias name (which may be the same as its filename), a work area letter, a work area number, or an indirect reference.

Usage

The FLOCK() function is a programming feature to prevent file update collision. A collision occurs when two users try to update a file at the same time. To avoid the collision, the first user to get access must lock the file with the FLOCK() function.

If the file lock attempt is successful, dBASE IV opens the file for the user's private use, and FLOCK() returns a logical true (.T.). The file remains locked until:

- It is unlocked by the UNLOCK command, or
- An operation closes the file, or
- The user quits dBASE IV

A locked file can't be modified by another workstation until the lock is released. However, the lock is a shared lock and users at other workstations can view the file.

If the file is already locked by another user, FLOCK() returns a logical false (.F.).

FLOCK() is effective for operations that can change many records in a file and can change many index files.

Use the FLOCK() function to lock all the records in a file for whole file type operations. Use the RLOCK() or LOCK() function to lock one record in a file for more efficient multi-user file access.

If you lock a file while it is actively related to other open files, then all the files in the relation are automatically locked. Unlocking any one of the related files automatically unlocks the other files.

Tips

File locking is most useful for whole file update operations, such as customized report programs or batch type operations. Generally, FLOCK() locking should not be used if the end user intervenes during the transaction cycle (that is, if there is an @ <row>, <col> GET/READ sequence in the update sequence).

Examples

Assume that Trial_bal.dbf contains trial balances and Transact.dbf contains transaction debits and credits. The following dBASE program uses the FLOCK() function to lock Trial_bal.dbf while it is being updated:

```
*************************************************************
*Trial_bal is opened as a shared file with read/write access.*
*Transact is opened as an exclusive use file.               *
*************************************************************

SET EXCLUSIVE OFF
SELECT A
USE Trial_bal INDEX Accts
SELECT B
USE Transact EXCLUSIVE
PUBLIC Time

*************************************************************
* Perform a wait loop; that is, wait until FLOCK() is TRUE   *
* but on the 250th test, if FLOCK() is still not TRUE, error *
* out.                                                       *
*************************************************************

SELECT A
Time=1
DO WHILE .NOT. FLOCK() .AND. Time<250
   Time=Time+1
ENDDO
IF .NOT. FLOCK() && The lock will be tried one more time with this command.
   DO Time_out
   RETURN
ENDIF

*************************************************************
* Now the file Trial_bal is locked and updated. *
*************************************************************
```

FLOCK()

```
SELECT B
GO TOP
DO WHILE .NOT. EOF()
   STORE ACCOUNT TO Mkey
   SELECT A
   SEEK Mkey
      IF .NOT. EOF()
         REPLACE Ac_debit WITH Ac_debit + Trns_dbt,;
                 Ac_credit WITH Ac_credit + Trns_cdt
      ENDIF
   SELECT B
   SKIP
ENDDO
SELECT A
UNLOCK
```

See Also

RLOCK()/LOCK(), UNLOCK

ISMARKED()

ISMARKED() checks the current work area or the optional alias to determine if a database file is having transaction updates added to it and is in an inconsistent state.

Syntax

ISMARKED([< alias >])

where < alias > is an alias name (which may be the same as its filename), a work area letter, a work area number, or an indirect reference.

Usage

This function indicates whether a database is in a constant state or in a state of change. If the database is marked for change, this function returns a logical true (.T.). If the database file is not marked, this function returns a logical false (.F.).

Example

The following example checks to see if a transaction is in progress and either delays the replacement of variables when a transaction is in progress or replaces the variables if one is not in progress.

```
USE customer
IF ISMARKED()
    @ 5,5 SAY "The file you need has an active transaction on it."
    @ 6,5 SAY "Please try again later."
    RETURN
ELSE
    REPLACE ALL salary WITH salary * 1.1
ENDIF
```

See Also

BEGIN TRANSACTION, COMPLETED(), END TRANSACTION, RESET, ROLLBACK, ROLLBACK()

LKSYS()

LKSYS() determines who has locked a record or a file and the date and time it was locked.

Syntax

LKSYS(< expN >)

Usage

The CONVERT command must be used prior to executing this function. The LKSYS() function converts the values in the _DBASELOCK field from hexadecimal to ASCII, and returns a value for the specified argument.

There are three arguments in this function:

expN = 0 returns the time when the lock was placed.
expN = 1 returns the date when the lock was placed.
expN = 2 returns the log-in name of the user who placed the lock.

The field itself cannot be listed or displayed, but the LKSYS() function will return values stored in the field.

Examples

The following example returns the user ID of the user who locked the record:

```
user_lock = LKSYS(2)
@ 22,5 SAY "The data record is LOCKED ..."
@ 23,5 SAY "Last locked by:"
@ 23,21 SAY user_lock
```

If a user named BILLC locked a record, the variable user_lock would display his user ID, as follows:

```
The data record is LOCKED ...
Last locked by: BILLC
```

The following example displays the time the current record was locked:

```
. DISPLAY LKSYS(0)
```

See Also

CHANGE(), CONVERT

MESSAGE()

MESSAGE() returns an error message character string.

Syntax

MESSAGE()

Usage

The MESSAGE() function allows the network programmer access to system error message strings. The program can display the error messages on the screen, or can store them to a variable.

ON ERROR must be active for this function to receive a value. If RETRY or RETURN is used within the ON ERROR procedure, the MESSAGE() string becomes unavailable after either command is processed.

Examples

The following error procedure retries to lock a file if error 108 occurs or to lock a record if error 109 occurs. It also provides for the system displaying the text of the error message, if any other error is encountered.

```
DO CASE
CASE ERROR()=108
   RETRY
CASE ERROR()=109
   RETRY
OTHERWISE
   @ 21,15 SAY MESSAGE()
ENDCASE
```

See Also

ERROR()

Network-specific error messages (Appendix A of this manual)

The complete list of error messages and codes in the *Language Reference* manual (Appendix A)

NETWORK()

NETWORK() indicates whether the network version of dBASE IV is running.

Syntax

NETWORK()

Usage

This function returns a logical true (.T.) if the multi-user version of dBASE IV is running, and a logical false (.F.) if the single-user version is running.

Example

```
IF NETWORK()
   ACCEPT "Do you want to see who else is logged in?" TO answer
   IF UPPER(answer)="Y"
      DISPLAY USER
   ENDIF
ENDIF
```

RLOCK()/LOCK()

RLOCK()/LOCK() is used to lock multiple records. The two function names are interchangeable.

Syntax

RLOCK([< expC list > , < alias >]/[< alias >])

or

LOCK([< expC list > , < alias >]/[< alias >])

where < alias > is an alias name (which may be the same as its filename), a work area letter, a work area number, or an indirect reference.

Usage

RLOCK() or LOCK() allows programs to coordinate multiple transaction sequences. A lock on a record prevents different transactions from gaining simultaneous access to the same record. However, the lock is a shared lock and users at other workstations can view the file.

Record locking is a control mechanism that prevents multiple, serial database transactions from interfering with each other. The process for a serial database transaction requires three operations: read data, modify data, write data.

The RLOCK()/LOCK() function enables multiple transaction sequence control without the need to lock all the information in a file. It enables locking of only the data to be changed. The other records of the shared file are left free for use by other users.

The RLOCK()/LOCK() function locks the records whose numbers are in the numeric expression list. In addition, all other records that are related to the records in the list are locked. Records are related if you use the SET RELATION command. The maximum number of records that can be locked by RLOCK()/LOCK() at any one time is 50. If you attempt to LOCK more than 50 records, the system returns the error message **Lock table is full**.

If you do not use a record number list, then the current record in the specified alias is used.

If all the records in the list and all other records related to them can be locked, the RLOCK()/LOCK() function returns a logical true (.T.) and locks all of the records. Otherwise, the function returns a logical false (.F.), indicating that one of the records is already locked.

The RLOCK() function is additive; it does not unlock the previous locked records. A locked record is not unlocked until you issue an UNLOCK command or an FLOCK() function.

Examples

The RLOCK()/LOCK() function attempts to lock the specified records in the active database file if a numeric expression list is included and an alias name is not specified:

```
. ? RLOCK("3,5,8")
```

If you specify an alias name without specifying a numeric expression list, an attempt is made to lock the current record number in the alias file:

```
. ? RLOCK("Alias_file")
```

If you specify both a numeric expression list and an alias name, an attempt is made to lock the specified records in the alias file:

```
. ? RLOCK("3,5,8","Alias_file")
```

The following example uses the RLOCK( ) function without a specified expression list or alias name:

```
DO WHILE .NOT. RLOCK()
   DO TIMER
   IF TIMES_UP
      ? CHR(7)
      DO SHOW_MSG WITH "TIME-OUT ON WAITING FOR RECORD TO UNLOCK..."
      ? CHR(7)
      ON KEY LOOP
      DO SHOW_MSG WITH "PRESS ESC KEY TO EXIT OR SPACE KEY TO CONTINUE WAITING..."
   ENDIF
ENDDO
```

See Also

FLOCK(), SET RELATION, UNLOCK

ROLLBACK() determines whether the last ROLLBACK command was successful.

Syntax

ROLLBACK()

Usage

This function returns a logical true (.T.) if the last ROLLBACK command was successful. If the ROLLBACK command was not successful, this function remains false until there is a successful ROLLBACK, or until you exit dBASE IV.

Example

The following program (from the sample applications) is an example of how the BEGIN TRANSACTION, ROLLBACK, and END TRANSACTION commands can be used to create a transaction log file and perform a recovery using that file.

```
USE employee TAG status
BEGIN TRANSACTION
REPLACE ALL salary WITH salary * 1.055 FOR department = "SALES"
END TRANSACTION
IF completed()
   * when true, indicates a completed transaction session
   clear
   @ 10,20 SAY "**** TRANSACTION SUCCESSFULLY COMPLETED ****"
   @ 12,20 SAY "**** SALARIES HAVE BEEN INCREASED ****"
ELSE
   * when false, indicates problems in the transaction session
   clear
   ? chr(7)
   @ 10,20 SAY "**** TRANSACTION NOT SUCCESSFUL ****"
   @ 12,20 SAY "**** ROLLBACK COMMENCING ... ****"
   ON ERROR DO err_proc
   ROLLBACK employee
   IF rollback()
      * when true, indicates a successful rollback to initial state
      @ 16,20 SAY "**** ROLLBACK COMPLETE. ****"
      @ 18,20 SAY "**** CHECK YOUR DATA BEFORE ****"
      @ 20,20 SAY "**** RUNNING THIS PROGRAM AGAIN... ****"
```

```
   ELSE
      * when false, indicates an unsuccessful rollback
      ? chr(7)
      @ 16,20 SAY "**** ROLLBACK FAILED.......... ****"
      @ 18,20 SAY "**** CHECK YOUR DATA BEFORE ****"
      @ 20,20 SAY "**** RUNNING THIS PROGRAM AGAIN... ****"
      WAIT
      IF ISMARKED(employee)
         * integrity flag remains in database-incomplete rollback
         ? chr(7)
         @ 18,20
         @ 20,20 SAY "**** DATABASE STILL CONTAINS INTEGRITY FLAG. ****"
         ACCEPT "RESET DATABASE FROM ROLLBACK ? (Y/N)" TO reset_db
         IF UPPER(reset_db) = "Y"
              RESET IN employee
              @ 22,20 SAY "DATABASE RESET -- MANUALLY RECOVER..."
              @ 23,20 SAY "CHECK DATA AND RERUN THIS PROGRAM..."
         ELSE
              ? chr(7)
              @ 22,20 SAY "DATABASE NOT RESET -- MANUALLY RECOVER..."
              @ 23,20 SAY "CHECK DATA AND RERUN THIS PROGRAM..."
         ENDIF
         WAIT
      ENDIF
   ENDIF
ENDIF
********************** END OF TRANSACT.PRG ************************
```

See Also

BEGIN TRANSACTION, COMPLETED(), END TRANSACTION, ERROR(), ISMARKED(), RESET, ROLLBACK

The complete list of error messages and codes in the *Language Reference* manual (Appendix A)

USER()

USER() indicates the user name of the user logged in to a protected system.

Syntax

USER()

Usage

This function returns the name of the user logged in at a workstation. The system must be protected for a name to be returned. If the system is not protected, this function returns a null string.

Example

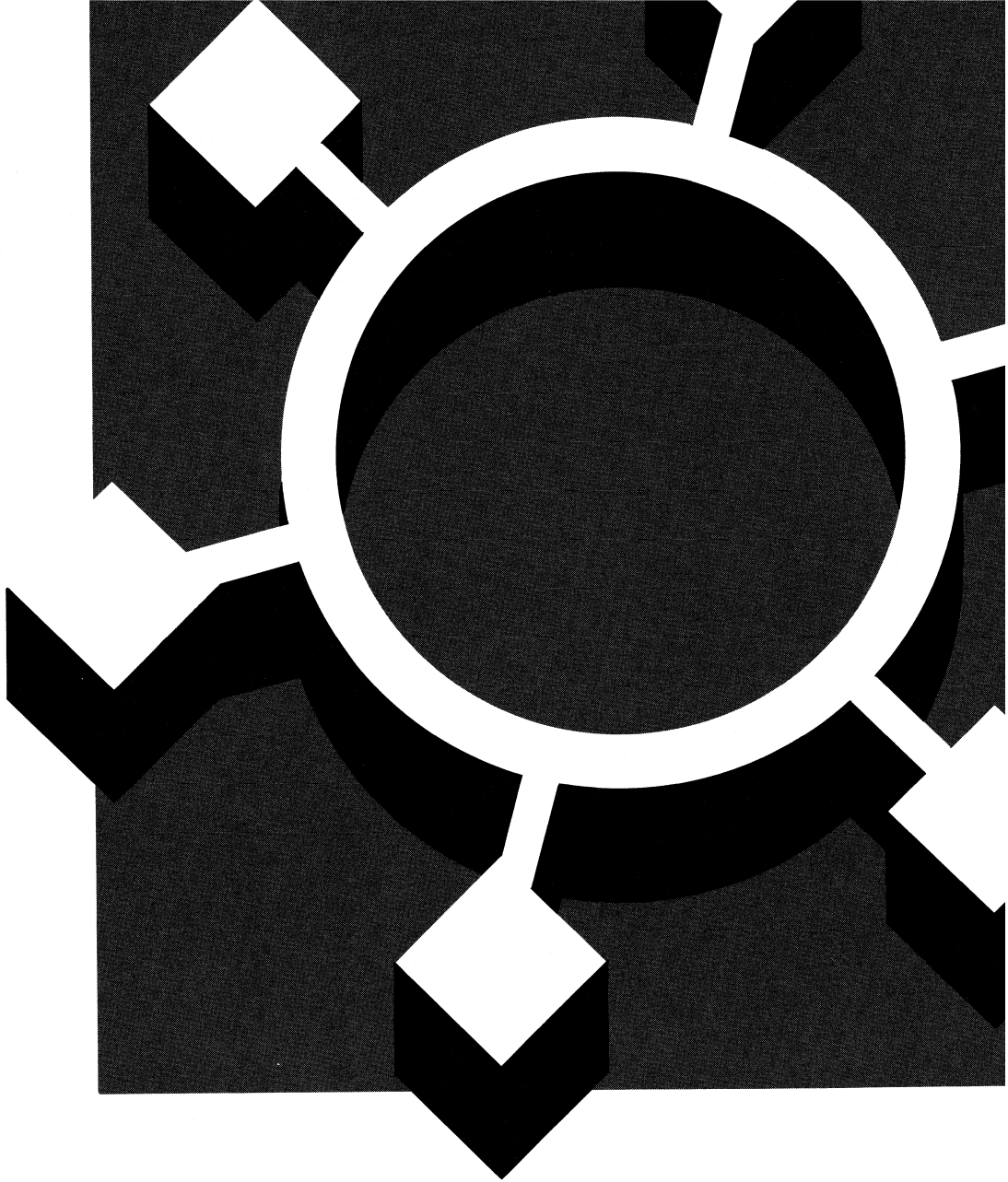
```
. ? USER()
FRED
```

See Also

PROTECT

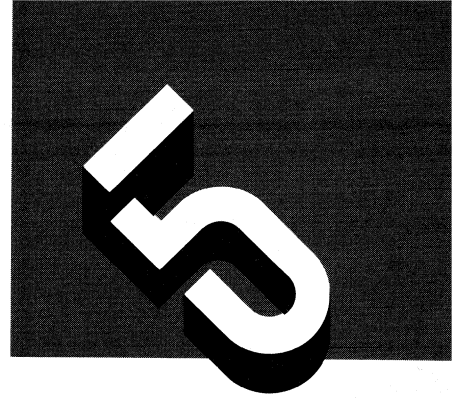
Network Administrator

5. dBASE Security



i 1 2 3 4 5 A Gl In

dBASE Security



About This Chapter

This chapter, which describes how to use the PROTECT security command, discusses the following:

- PROTECT security concepts, including levels of security
- Using PROTECT menus to set up a security system
- System password files
- Operational considerations that affect network security

NOTE
SQL users should refer to Using dBASE IV SQL *for information about securing data in the SQL environment.*

About dBASE Security

The PROTECT command is used to create and maintain security on a dBASE IV system. PROTECT is a menu-driven command issued inside dBASE IV by the database administrator who is responsible for data security. PROTECT may be used on a single microcomputer or in a local area network environment.

PROTECT is optional: you don't have to use it. Once you have used PROTECT, however, the security system will always control access to database files.

PROTECT includes three distinct types of database protection:

- *Log-in security*, which prevents access to dBASE IV by unauthorized personnel
- *File and field access security*, which allows you to define what files, and fields within files, each user can access
- *Data encryption*, which enciphers dBASE files so that unauthorized users cannot read them

Table 5-1 summarizes the database security types, how to implement each security type, and the results of security implementation.

Table 5-1 dBASE security summary

Security Type	You Define:	You Get:
Log-in	User name and password	Control over access to dBASE IV
File and Field Access	Access levels	Control over access to data files and fields in data files
Data Encryption	User and file group	Automatic encryption and decryption of data

The security types listed in Table 5-1 and described more fully in this chapter constitute levels of security. Log-in security is the first security level. Once a security system is in place, users cannot access dBASE IV until they pass log-in security. Access control is the next security level. Access control determines what a user can do both with a database file and data in the file, and can be used to control processing of application code. Data encryption scrambles the database so that unauthorized users cannot read the information in the file.

You must implement the security types in the order shown above. You cannot establish file and field access control without first creating log-in security. Similarly, you must create both log-in security and file and field access control in order to have data encryption. However, it is not necessary to implement all three levels of security. Many database administrators elect to implement just log-in security.

If you decide to use PROTECT, you must create a *user profile* for each operator. See "Creating User Profiles," later in this chapter.

Log-in Security

PROTECT allows you to create a password-protected system. If password protection is in force:

- No user can gain access to dBASE IV on that system unless the user enters a valid login. The login consists of three items: a group name, a log-in name, and a password.
- PROTECT displays the user log-in screen whenever you access dBASE IV. All paths into the database system initiate the log-in process.

Access Level Security

You control access to database files and fields within those files by assigning *user access levels* that determine the user's file access and field access *privileges*. The file access privileges and the field access privileges for a file are called its *privilege scheme*.

User access levels are numbered 1 through 8. Assigning a low number gives the user greater access privileges. Assigning a higher number limits the user's access.

You establish an access level for each user in the user's profile, and additional access levels for file and field privileges in the file privilege scheme.

After you log in, dBASE IV determines what access level you have with a file by matching the user access level with the file's privilege scheme.

You can assign any number of users to each access level, but only one access level to the same user in the same group. A user's ability to access a particular file is a function of both group membership and access level. However, only access level determines what the user can do with the file once it is accessed.

File Access Privileges

You establish privileges for a database file by assigning access levels to the operations that a user can do on it. You can assign access levels, in any combination, to read, update, extend, and delete privileges. These privileges grant users the ability to:

- View records in a database file (read privilege)
- Change database file record contents (update privilege)
- Append new records to a database file (extend privilege)
- Delete records from a database file (delete privilege)

If you do not use PROTECT to create a privilege scheme for a database file, all users can read and write to all fields in the file. When you use PROTECT to create a file privilege scheme, all four file privileges are granted initially until you change them. You change the default by specifying the most restricted level that is to have each privilege.

Field Access Privileges

At the field level, you can control what operations each user is allowed. You can grant full (FULL), read-only (R/O), or no access (NONE) privilege to each field in a database file *for each access level.* The field privileges allow users to:

- Read and write the field in the database file (FULL privilege). This is the initial default.
- Read but not write the field (R/O privilege).
- Neither read nor write the field (NONE privilege).

  When NONE is selected, a user is blocked from writing to fields and even from seeing fields you do not want displayed.

Data Encryption

Data encryption scrambles data so that it can't be read until it is unscrambled. An encrypted file contains data that has been translated from source data to another form that makes its contents unreadable. If your database system is PROTECTed, dBASE IV automatically encrypts and decrypts database files and their associated index and memo files.

The User and File Group

When you create each user's security profile, you assign each user to a group. The user's group must be specified at login. As part of each file's privilege scheme, you assign the file to a group. A file can be assigned to only one group. If the user group and file group do not match, the user cannot access the file.

NOTE
A user can belong to more than one group. However, each group that a user belongs to must be logged into separately. If a user needs to access files from two different groups, the user must log in twice, specifying a different group name at each login.

Typically, each group is associated with a set of files. By associating each application with its own group, you use the group to control data access. Further, by using access levels within the group, you can give different users different kinds of access to the application program files.

System Password Files

PROTECT creates and maintains the Dbsystem.db and Dbsystem.sql system password files, which contain records for each user defined through PROTECT. Dbsystem.db stores dBASE IV user profiles, which include the user's log-in name, account name, password, group name, and access level. Dbsystem.sql stores user log-in names and passwords for use by SQL. When a user enters the dBASE command at a network workstation, dBASE IV looks for the applicable system file. If it is found, the log-in process is initiated. If it is not found, there is no log-in process.

Dbsystem.db and Dbsystem.sql are maintained as encrypted files that can't be decrypted. Only a database administrator can print this information. (See the "Printing Security Information" section later in this chapter.)

Creating a Security System

To create a PROTECTed database system:

1. Initiate the PROTECT command.
2. Define the database administrator's password.
3. Define user profiles.
4. Define file privilege levels and field privileges.
5. Save the security information.

The following sections describe these steps in detail.

Initiating PROTECT

You initiate PROTECT from within dBASE IV after all database files have been closed. To start PROTECT at the dot prompt, enter **PROTECT** and press ↵. You can also select **Protect data** from the Control Center through the **Tools** menu.

The Database Administrator Password

When you enter the PROTECT command, the dBASE IV Password Security System log-in screen appears, as shown in Figure 5-1.

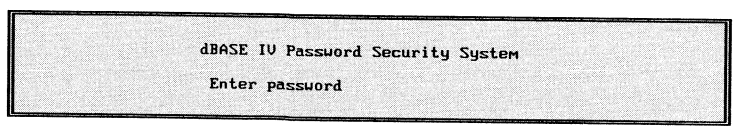


Figure 5-1 dBASE IV Password Security System log-in screen

Enter a password of up to 16 alphanumeric characters on the log-in screen. You can enter alphabetic characters in upper or lower case. The password is not displayed on the screen. For maximum log-in security (and maximum encryption protection), you should specify the full 16 characters allowed.

The first time you use PROTECT, the system prompts you **Please reenter password to confirm**. You must enter your password again. Thereafter, each time you use PROTECT, you enter the password only once. The system gives you three chances to enter the password correctly before PROTECT terminates.

WARNING
Remembering the administrator password is essential. You can access the security system only if you can supply the password. Once established, the security system can be changed only if the administrator password is supplied when PROTECT is called. Keep a hard copy of the database administrator password in a secured area. There is no way to retrieve this password from the system.

If you pass security, the system displays the **Users** menu, through which you create user profiles.

Creating User Profiles

The **Users** menu, shown in Figure 5-2, creates or modifies a user profile in the Dbsystem.db file.

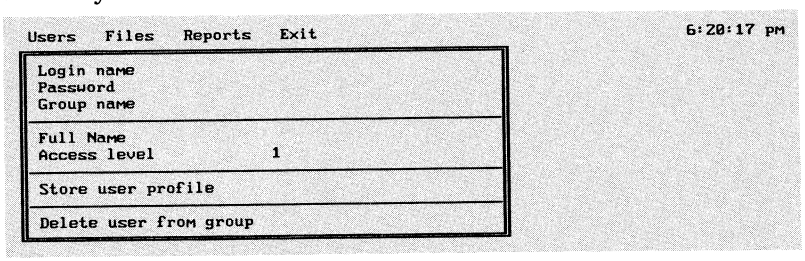


Figure 5-2 Users menu

You use the **Users** menu to:

- Add, change, and delete user profiles
- Establish and change the access level for each user

To select an option in the menu, move the highlight (with ↑ or ↓) and press ↵, or press the first letter of the menu choice.

Adding a New User Profile

Follow the steps below to add a user profile to the Dbsystem.db file:

1. Access the **Users** menu. The cursor is at the first field, **Login name**. Press ↵.
2. Enter a user log-in name (1-8 alphanumeric characters). Press ↵. This entry is converted to upper case.
3. The **Password** option is highlighted. Press ↵. Enter a user password (1-16 alphanumeric characters). Press ↵.
4. The **Group name** option is highlighted. Press ↵. Enter a group name (1-8 alphanumeric characters). Press ↵. This entry is converted to upper case.
5. The **Full name** option is highlighted. Press ↵. Enter the full name (1-24 alphanumeric characters), if desired. Press ↵.
6. The **Access level** option is highlighted. Press ↵. Select a user access level (a number from 1 through 8).
7. The **Store user profile** option is highlighted. Press ↵ to store the user profile.

You must specify a value for log-in name, password, and group name, or a user profile will not be created.

Note that:

- If you intend to use SQL, you must add the super user log-in name SQLDBA, which is granted privileges to all operations in SQL mode. The SQL GRANT and REVOKE commands control file and field access privileges using log-in names assigned by PROTECT.
- The full name is the only optional item in a user profile. Since this item is not used in validating a login, you can use it any way you want. Frequently, the full name is used to add a more complete user identification. Alphabetic characters you enter in the **Full name** option will not be converted to upper-case characters.
- You may find it useful to organize users and files into groups that reflect application use (see "The User and File Group" section above).
- The user group name will be matched with the file group name to enable file access.
- Within each group, the user is assigned an access level. This level will be matched with file access levels established with the **Files** menu to determine what access level the user has for each database file. It will also determine the type of access the user has to each file and, within each file, to each field. See the "Establishing File Privilege Levels" section below for a discussion of access levels and how they should be assigned.

When you have entered all the items, you must select **Store user profile** to save the new profile. You can then do one of the following:

- Return to the first option on the menu (**Login name**), and begin entering another user profile
- Move to another menu by pressing ← or →

At any time during the definition or modification of a user profile, you can terminate the process. The user profile information is not saved until you store it.

Changing a User Profile

When you enter the first three items on the **Users** menu (**Login name**, **Password**, and **Group name**), PROTECT checks to see if the user profile has already been defined. If it has, the rest of the menu items are completed with their current values. You can then change any of the values, but it is recommended that you do not change the group name.

To change a user profile, follow the steps below:

1. Open the **Users** menu.
2. Enter the log-in name, password, and group name of the user profile you want to change.
3. Enter Y in response to the **User already exists, do you want to edit?** prompt.
4. Press ↑ to highlight the **Login name** field or the **Password** field. Press ↵. Do not change the group name.

If you edit the group name, there will be no way to access files associated with the original name. You also should not delete the group. If you delete a group name before all files associated with the group are copied out in a decrypted form, no one can access the files.

Remember that you will need to enter the user profile changes and, later, save them. (See the "Adding a New User Profile" section above.)

Deleting a User Profile

The last option on the **Users** menu is used to delete a user profile. Follow the steps below:

1. Open the **Users** menu.
2. Enter the log-in name, the password, and the group name.
3. Move the highlight to **Delete user from group**.
4. Press ↵.

File Privilege Schemes

The **Files** menu creates or modifies file privilege schemes in which file and field privileges are assigned to match user access levels. The file privilege schemes are saved in the database file structure. Press ← or → to move to the **Files** menu, illustrated in Figure 5-3.

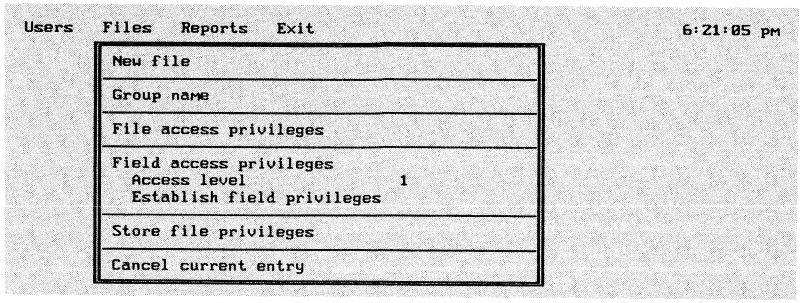


Figure 5-3 Files menu

You use the **Files** menu to:

- Assign a file to a specific group.
- Create and change access levels for file privileges. You can define up to eight levels of privilege for each file.
- Assign field access privileges for each file access level.

To select an option in the menu, use the ↑ or ↓ key to move the highlight to the desired option and press ↵, or press the first letter of the menu choice.

Creating a File Privilege Scheme

Follow the steps below to define file and field privileges for a database file:

1. Access the **Files** menu.
2. Select a file.
3. Assign the file to a specific group.
4. Establish the most restrictive access level for each file privilege.
5. Select an access level for field privilege assignment.
6. Select field privileges for each field at each access level, as required.
7. Store the file and field privilege scheme.

The sections that follow describe these steps in detail.

A file is not encrypted unless you select it (**New file**), store it (**Store file privileges**), and save it (**Exit**). With these choices, you can change the file privilege scheme or accept the preset menu values for the privileges. These values, also called *default values*, are set as follows:

- The access levels for all file privileges are set to 8
- Field privileges are set to FULL

If these access level/privilege associations are not set, the only access controls are those established by the network operating system. See your DOS manual or the appropriate appendix of the *Network Installation* manual to define access controls outside of dBASE.

Table 5-2 summarizes the values you enter in this menu and lists the default values for items on this menu.

Table 5-2 Files menu items summary

Menu Item	Value Type	Value	Initial Default
New file	Menu list item	File selected from file list	The current drive designation
Group name	User-defined character string	1-8 alphanumeric characters	None
Read privilege	Integer	1 through 8	8
Update privilege	Integer	1 through 8	8
Extend privilege	Integer	1 through 8	8
Delete privilege	Integer	1 through 8	8
Access level	Integer	1 through 8	1
Field access privilege (set for each field at each access level)	Enumerated data type	Fields selected from field list	First field in list
		FULL, R/O, NONE	FULL *

* Once field privileges are set for an access level, the field privileges for more restricted levels that are not set default to FULL. (See the "Establishing Field Access Privileges" section below.)

Selecting Database Files

Pressing ↵ to choose the **New file** option displays a file list. The file list illustrated in Figure 5-4 contains the names of all the database files in the system.

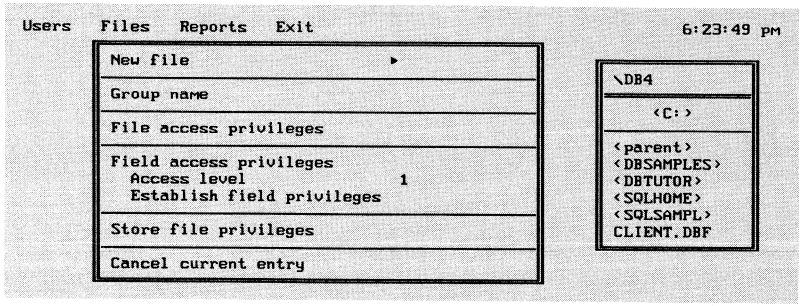


Figure 5-4 File list

Move the highlight to the name desired and press ↵. The highlighted file name will be displayed in the **New file** option.

WARNING
You can create database file privilege schemes for up to nine database files at a time. If you try to set up a tenth file, you will get the error message **Too many files are open**. When you have finished creating the eighth scheme, move to the **Exit** menu and select **Save** or **Exit** to save the database file privilege schemes. If appropriate, you can then move back to the **Files** menu and continue defining database file privilege schemes.

Assigning the File to a Group

A file can be assigned to only one group. The group name will be matched with a user group name to enable data access. Try to organize users and files into groups that reflect application use (for instance, by department or sales area). Through group association you can establish file and user sets to limit application access by user and, within applications, to limit what users can do with applications. (See "The User and File Group" section above.)

Establishing File Privilege Levels

Next, you establish the privilege levels for the selected file. File privileges specify the access rights to a file. When you select **File access privileges**, the system displays the **File access levels** submenu shown in Figure 5-5.

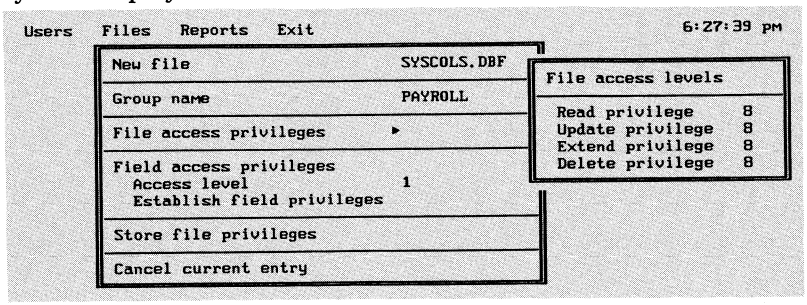


Figure 5-5 File access levels submenu

These access rights cannot override a read-only attribute established for the file at the operating system level.

In building a file privilege scheme, note that:

- Access level 1 has the most privilege, and access level 8 has the least. Level 1 is called the least restrictive access level, while level 8 is called the most restrictive.
- The more privileged levels (1, 2, 3) are typically assigned to the fewest people. To limit access to your data, the more privileges a level has, the fewer users you should assign to that level.

File privileges determine permissions for the file operations listed in Table 5-3.

Table 5-3 File privilege operations

Privilege	Access Granted
DELETE	Delete records from the database
EXTEND	Add records to the file
READ	View the file contents
UPDATE	Edit existing records in the file

For each type of file privilege, you can specify the most restricted access level to have the privilege. All levels less restricted than the specified one will be granted the file privilege; all levels more restricted than the specified one will not be granted the file privilege. For example, if you specify 6 as the READ privilege level for the file Customer.dbf, all users with access levels of 1 through 6 will be able to read the file. However, users with access levels of 7 and 8 will not be able to read the file.

You assign and change file privileges by entering an integer value. To change the value, highlight the item, press ↵, and set the value, or use the ↑ and ↓ keys to change the value.

Establishing a Field Access Level

Move the highlight to **Access level** and enter the access level for which you wish to define field access. The eight file access levels match the user-assigned access levels (1 through 8). Remember that level 1 has the most rights and level 8 has the least.

Establishing Field Access Privileges

Finally, you establish Field Access privileges by moving the highlight to **Establish field privileges** and pressing ↵, or by pressing the first letter of the menu choice. A fields list appears next to the **Files** menu. You can establish one of the field access privileges listed in Table 5-4 for each field in the selected file at the selected access level.

Table 5-4 Field privilege operations

Privilege	Access Granted
FULL	View and modify the field. This is the default.
R/O	View the field only (no update capability).
NONE	No access. The user can neither read nor update the field, and the field appears as if it is removed from the database file.

NOTE
File privileges take precedence over field privileges. For example, if a file privilege is set for READ, but not UPDATE, the only meaningful field privileges are R/O and NONE.

You must restrict file privileges to protect your data against file-oriented commands like DELETE or ZAP. Restricting field privileges to R/O or NONE without restricting file privileges does not protect your data against these commands.

The fields list contains all fields defined in the database file and the current field access privilege assigned to them. Initially, all field access privileges are set to FULL. The procedure to change the field privilege is as follows:

1. Move the highlight to the field for which privilege is to be changed.
2. Use ↵ to switch the preset values between FULL, R/O, and NONE.
3. When the appropriate field privilege is displayed, move to the next field to be changed by pressing ↑ or ↓.

Continue this process until all fields for the access level are set to the appropriate value, and then return to the **Files** menu by pressing ←.

Continue this process as long as necessary to establish field privileges for all user access levels. The procedure is as follows:

- Establish all the field access privileges for one level.
- Then, move back to the **Files** menu (using ←). Set the next access level, and move the highlight back to **Establish field privileges** (by pressing ↵) to assign field privileges at that level.
- This process continues until you have assigned all necessary field privileges.

If you specify a level and change field privileges, the changes you make affect that field only at that access level. All other fields and access levels remain the same.

Storing a File Privilege Scheme

When you are satisfied with all the file and field access privilege settings, you should select **Store file privileges** to store them. Remember that after you have defined or modified nine file privilege schemes, you must save them before you can continue using the **Files** menu. You can then do one of the following:

- Begin entering another file privilege scheme
- Move to another menu by pressing ← or →

NOTE
Do not store file privileges until you are completely finished with the file you are working on.

Cancelling a File Privilege Scheme

At any time during the definition or modification of a file privilege scheme, you can move to the last option on the menu, **Cancel current entry**, press ↵, and delete the definition in progress.

Changing a File Privilege Scheme

When you select a file, PROTECT checks to see if the file privilege scheme has already been defined. If it has, the rest of the menu items are completed with their current values. You can then change any of the values. If the file privilege scheme has already been saved, the message **< filename > .crp already exists, overwrite it? (Y/N)** is displayed. Press Y if you wish to change any values. (This is not affected by the status of SET SAFETY.)

Make your changes. Remember that you will need to store the changes and, later, save them. (See the "Creating a File Privilege Scheme" section above.)

Printing Security Information

Dbsystem.db is maintained as an encrypted file. There is no way for you or any other user to examine the contents of that file once information is stored in it. Therefore, you will probably want to keep a hard copy record of some or all of the information contained in Dbsystem.db. For example, if a user forgets a log-in value (such as group, log-in name, or password), you will want to have that information available.

The **Reports** menu illustrated in Figure 5-6 allows you to display or print security information about users and files.

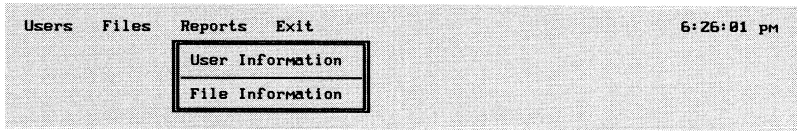


Figure 5-6 Reports menu

The User Information Report lists the names of all system users, their passwords, group names, account names, and levels.

The File Information Report lists the name of the selected dBASE file, the group to which the file is assigned, the file privileges, the name of each field in the file, and the field privileges. The File Information Report cannot be run on a file that has not been saved.

Selecting the **User Information** option displays a prompt box with the message **Send report to the printer? (Y/N)**.

Entering Y sends the output to the printer, and entering N displays the output on the screen.

Selecting the **File Information** option first displays a list of the available .crp files. Select a file from the list. The system displays the prompt **Enter Group Name:**.

Enter the correct group name for the selected database file. The system displays a box with the prompt **Send report to the printer? (Y/N)**.

Entering Y sends the output to the printer, and entering N displays the output on the screen.

Exiting from PROTECT

The **Exit** menu contains three options:

- Save
- Abandon
- Exit

Select **Save** to post all new and updated user profiles and file privilege schemes that have been stored during the current PROTECT session. User profiles are saved in the current Dbsystem.db file. File privilege schemes are saved in the database file structure. You can save user profiles at any point during a PROTECT session. You must save file privilege schemes after you define or change eight of them. Database files are encrypted when the file privilege scheme is saved.

Select **Abandon** to cancel all new and updated user profiles and file privilege schemes not already saved during the current PROTECT session.

Select **Exit** to terminate the current PROTECT session. New and updated user profiles and updated file privilege schemes will be encrypted and saved, if they have not already gone through this process.

Other Considerations

Keep the general considerations discussed in the rest of this chapter in mind as you build and use a PROTECTed database system.

Data Encryption

Be aware of the following:

- When a database file's privilege scheme is saved, PROTECT creates an encrypted version of the database file with a .crp extension. To enable security, you should:

  1. Copy the encrypted file (.crp) over the unencrypted file (.dbf) and change the extension to .dbf, as follows:

     F> COPY <filename.crp> <filename.dbf>

     You may want to copy the unencrypted file to a floppy disk and store the floppy disk in a secure place.

  2. Delete the .crp file, as follows:

     F> ERASE <filename.crp>

  3. Rename the .cpt file so that it has a .dpt extension.

- Index files are only encrypted when you REINDEX or create them with an encrypted database file.
- You can control when copied files are encrypted through the SET ENCRYPTION command.

- The time required to encrypt files depends on the size of your files. Files are encrypted when you select **Save** on the PROTECT **Exit** menu (see above) or exit PROTECT. If you are saving file privilege schemes of large files, it may take some time to exit PROTECT.

You can use the BUILD utility to encode application program files. (Refer to Chapter 18 in *Programming with dBASE IV*.)

You control whether newly created copied files (that is, destination files created by a COPY operation) are to be encrypted through the SET ENCRYPTION command. (See the "Using SET ENCRYPTION" section.)

Using SET ENCRYPTION

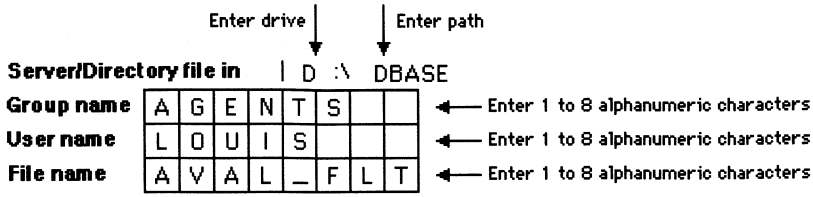
Even after a database system has been PROTECTed, the database administrator and application programmer maintain control over encryption of copied files.

If a database system has been PROTECTed, SET ENCRYPTION is ON by default. If you SET ENCRYPTION OFF, files created with the COPY command will not be encrypted. Refer to Chapter 4 for more information on the SET ENCRYPTION command.

General Security Considerations

Security is only as good as its degree of control and confidentiality. Maintaining the integrity of the security system is your first responsibility. Be sure you consider the points below when establishing your security system.

1. Decide how your users are to be assigned logins. Are they to select their user log-in name, password, and group membership, or will you assign the login? If they are to select values for the login, be sure that they know how to make such a request, how long user names and passwords can be, and what characters can be used in them. Passwords should use the full 16 characters allowed.
2. Once passwords are determined, make sure that you keep your password secret and that your users understand the importance of keeping their user log-in names and passwords secret.
3. If a user wants to create a file and protect its contents, make sure the user knows how to communicate that requirement to the database administrator prior to creating the file. You may wish to use a form to facilitate communication between yourself and your users, and also to have a hard copy of security information. You can use copies of the security request form included at the end of this chapter, or develop your own. A filled-out security request form (Figure 5-7) is also provided for your reference.
4. Develop a strategy for retaining the database administrator password in a secure place so that it can be available should it be forgotten or otherwise needed. Make sure it is well protected from unauthorized access.



File Privileges	◄ Least Restrictive						Most Restrictive ►	
Access Level	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Extend			X					
Delete		X						
Read								X
Update						X		

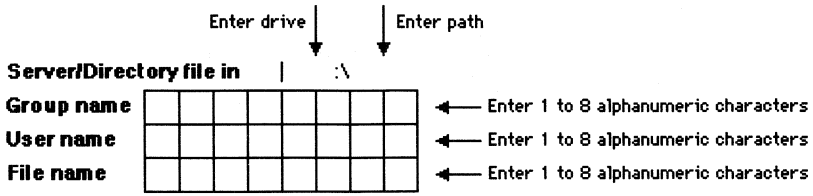
Enter X at Most Restrictive Level

Field Privileges									
		Type of Access							
		Access Level							
Fld. No.	**Field Name**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
1	AFLT_NO						F		
2	ADEP_CITY						F		
3	ADES_CITY						F		
4	ADATE						F		
5	ADEP_TIME						F		
6	AARR_TIME						F		
7	ACLASS						F		
8	ASEAT_AVL						F		
9	AFARE						F		
10									

Enter Field Name

Enter F for Full, N for None, or R for Read only

Figure 5-7 Sample security request form



File Privileges	Least Restrictive							Most Restrictive
Access Level	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Extend								
Delete								
Read								
Update								

Enter X at Most Restrictive Level

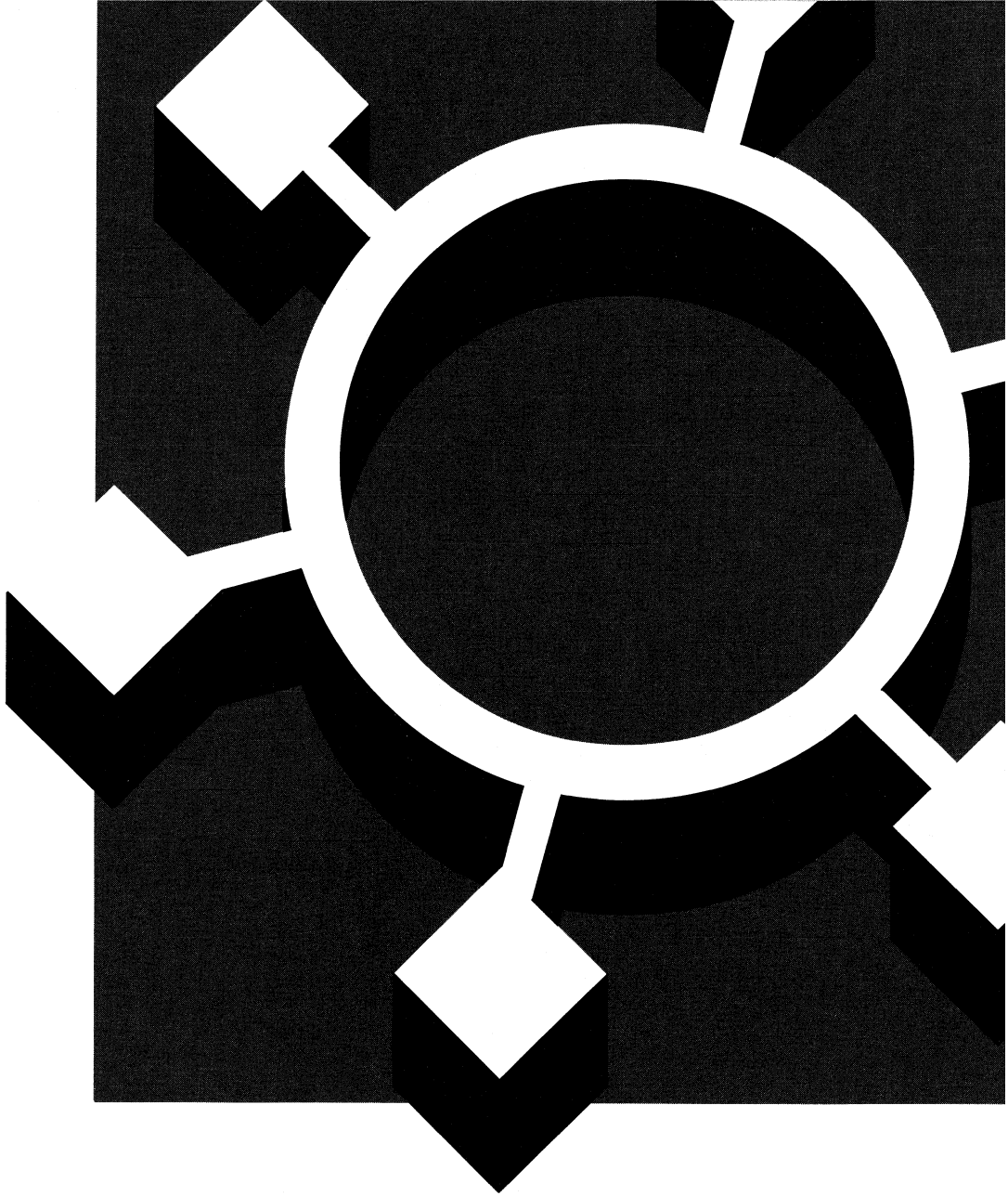
Field Privileges									
Fld. No.	**Field Name**	**Type of Access**							
		Access Level							
		1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
1									
2									
3									
4									
5									
6									
7									
8									
9									
10									

Enter Field Name

Enter F for Full, N for None, or R for Read only

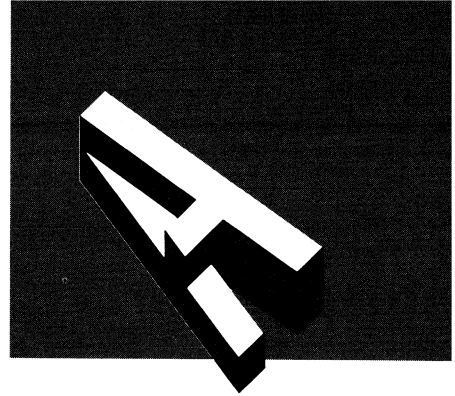
Error Messages

A. Network Error Messages



i 1 2 3 4 5 A Gl In

Network Error Messages



This appendix alphabetically lists error messages that are specific to the dBASE LAN environment. Each message is followed by a description of the cause of the error and, where appropriate, suggested corrective action. The error number associated with each message is shown in square brackets at the end of the message.

For error messages that may occur when you run single-user dBASE IV, refer to Appendix A in the *Language Reference* manual. For error messages that may occur while installing or uninstalling dBASE IV, refer to Appendix A in the *Network Installation* manual.

dBASE IV Error Messages

The following messages may be displayed when you start dBASE IV or after you begin running dBASE IV.

Startup Messages

Check network drive specification
You entered the command to start dBASE IV, but the dBASE IV and DBNETCTL.300 files could not be located on the file server.

You need to specify the correct drive and directory location of files, and set the correct DOS PATH command (depending on the network).

Control file cannot be processed
dBASE IV cannot locate the Dbase.ctl file in the DBNETCTL.300 directory, or the file is damaged.

Control file LOCK failure
An internal error occurred when checking the Dbase.ctl file.

Load failed
Files in the DBNETCTL.300 directory could not be read.

Maximum network users reached (xx) Please try later
The maximum number of users (displayed in parentheses) are already running dBASE IV.

Memory allocation error
An internal error occurred while DOS was trying to execute the Dbase.com program to start up dBASE IV.

Check network requirements for installed memory.

Network drive not supplied
You did not specify the correct network drive with #DF = when you attempted to start dBASE IV, or the DBNETCTL.300 directory files could not be found at the specified location.

Network load failure
This message usually appears due to a corrupted dBASE IV or DBNETCTL.300 directory file.

Try to run dBASE IV again. If you get the same message, you may have to uninstall dBASE IV and then reinstall it.

Unauthorized duplicate
The Dbase.lod file cannot be found in the DBNETCTL.300 directory.

dBASE IV Execution Messages

All database files must be closed before using PROTECT
Close all databases in use before attempting to use PROTECT.

Cannot close database when transaction is in process [185]
You must issue an END TRANSACTION command or attempt a ROLLBACK before you can close a database that is involved in a transaction.

Cannot close index files when transaction is in process [187]
You must issue an END TRANSACTION command or attempt a ROLLBACK before you can close index files involved in a transaction.

Cannot execute this command while transaction is in process [186]
See the BEGIN TRANSACTION command for a list of commands that are not allowed during a transaction.

Cannot nest transactions [198]
Transactions cannot be nested inside other transactions.

Issue an END TRANSACTION command before starting a new transaction.

Cannot write to a read-only file [111]
A file has been opened for read-only access, and an attempt has been made to write to the file.

Cannot write to database due to incomplete transaction [201]
An incomplete transaction has flagged the database as in use.

Issue either an END TRANSACTION or a ROLLBACK command to clear the in use flag from the database file.

Cannot write to transaction log file [188]
The required transaction log file is not open or available.

Issue an END TRANSACTION command to abandon the transaction you attempted.

Database encrypted [131]
An attempt has been made to open an encrypted database without passing through the log-in procedure.

Place Dbsystem.db in the same directory as dBASE IV, and log in.

Environment not correct for rollback [193]
You are attempting to do a recovery when there is no BEGIN TRANSACTION command and a transaction log file does not exist.

Error in reading log file [192]
The transaction log file is corrupted or otherwise cannot be read. A successful ROLLBACK is not possible.

File in use by another [108]
An attempt has been made to open a file that has already been opened for exclusive use by another user.

Try to open the file again later.

File in use by <username>. Retrying lock, press Esc to cancel. [372]
An attempt has been made to open a file that is locked by the user given in <username>. This message appears if you have specified SET REPROCESS ON.

File must be opened in exclusive mode [110]
An attempt has been made to issue an INSERT [BLANK], MODIFY STRUCTURE, PACK, REINDEX, or ZAP command, and the file has not been opened for exclusive use.

Use the SET EXCLUSIVE ON or USE <filename> EXCLUSIVE command, reopen the file, and then reissue the command.

File not in transaction log [194]
You cannot perform a ROLLBACK on a file that was not recorded in a transaction log file.

You must restore this file manually from backup copies.

Invalid printer redirection [124]
The path to a printer destination has not been established, or the printer is not shareable.

Lock table is full
The lock table allows a maximum of 50 locks per session.

Unlock some records and continue.

Log file corrupted [195]
The log file is unuseable. Recovery is not possible.

Log file not found [191]
The log file is missing. Recovery is not possible.

Log record does not match database record [196]
The log file is unuseable.

Network server busy [148]
Your file server cannot process the number of tasks requested on the network. This message indicates that the server may not have executed dBASE tasks or commands correctly.

You should quit dBASE IV and reboot the network. To correct the problem, you need to reduce network traffic.

Password and confirmation mismatch [154]
When entering the administrator password, the confirmation did not match the original password.

Re-enter the password.

Password has not been defined [157]
You have entered an incorrect password.

Try again.

Record in use by another [109]
An attempt has been made to read or lock a record that has already been locked by another user.

Try to access the record again later.

Record in use by < username > . Retrying lock, press Esc to cancel. [373]
An attempt has been made to open a record that is locked by the user given in < username > . This message appears if you have specified SET REPROCESS ON.

Relation record in use by another [142]
A record that is related to the record you want to access is in use.

Try again later.

Rollback database cannot be executed inside a transaction [197]
Move the ROLLBACK command after the END TRANSACTION command. ROLLBACK < database filename > cannot be used inside a transaction. ROLLBACK < database filename > is used after a system failure to roll back one database file at a time.

Unable to LOCK [129]
An attempt has been made to use **Ctrl**-O to lock the current record, which has already been locked by another user.

Try to lock the record again later.

Unable to SKIP [128]
An attempt has been made to use **Ctrl**-C or **Ctrl**-R to skip to a locked record.

Try to access the record again later.

Unauthorized access level [133]
An attempt has been made to access a file or field without the appropriate privilege.

See your database administrator.

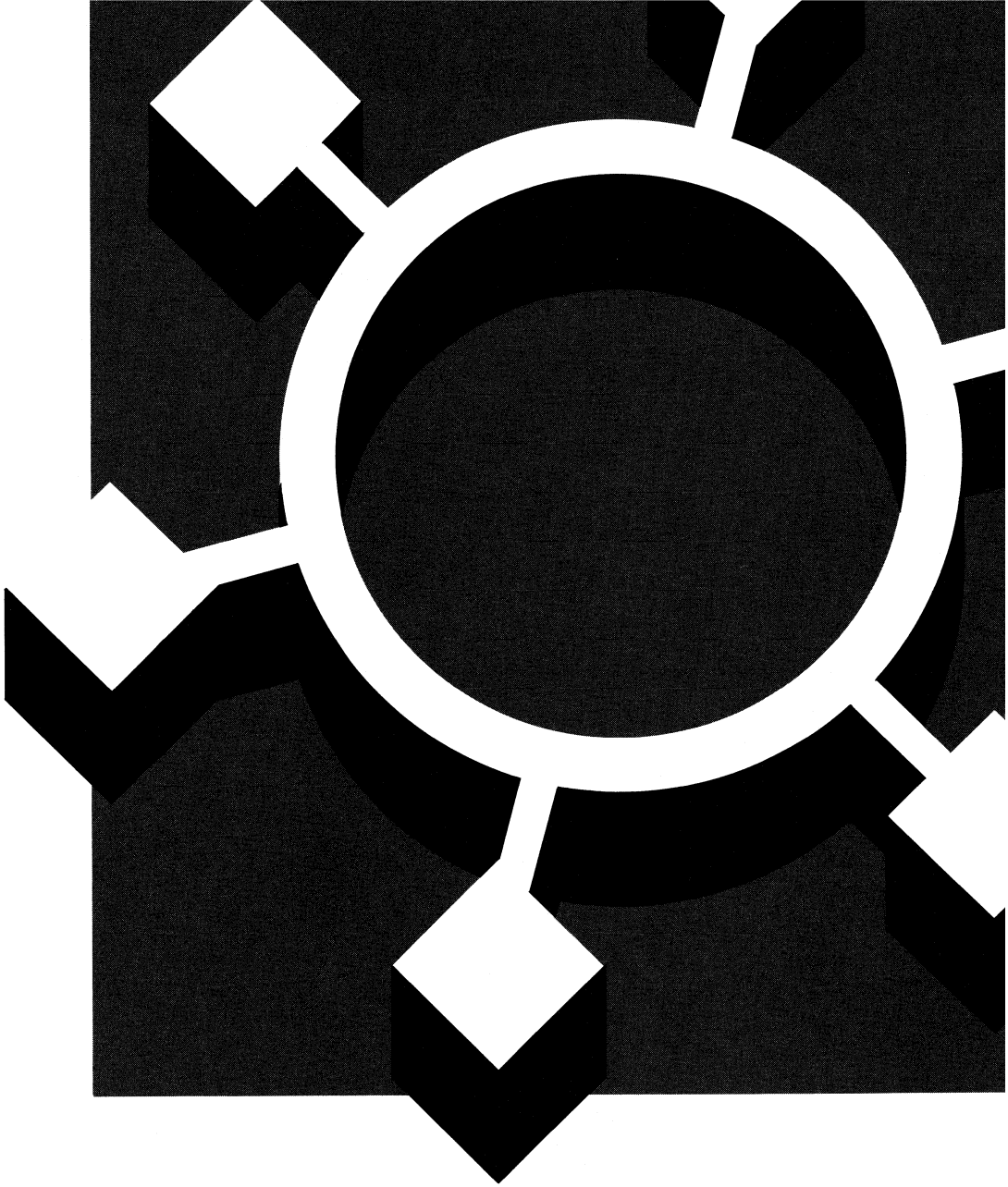
Unauthorized login **[132]**
Three attempts have been made to log in to dBASE IV, none of which were successful. dBASE IV will return you to the default network drive prompt.

This message also appears if the user login you are using is already in use at another workstation.

Unterminated transaction file exists, cannot start new transaction **[183]**
Issue a ROLLBACK command at this point. You cannot end the transaction. You must roll it back. If the rollback fails, ERASE the Translog.log file. If there are two users logged in to the network operating system with the same name, log out one of them. Transaction processing requires that the first eight characters of any network log-in name be unique.

**** WARNING ** Uncompleted transaction found** **[190]**
You are attempting to use a BEGIN TRANSACTON command before first issuing an END TRANSACTION command.

Glossary



i 1 2 3 4 5 A GI In

Glossary

This glossary includes terms that are specific to *Networking With dBASE IV*. For other dBASE IV term definitions, refer to the glossaries in other dBASE IV manuals.

Access The ability to use system resources.

Access level A number that determines user file access in a PROTECTed database system. An access level assigned in the user profile is matched against an access level in a file privilege scheme to determine what a user can do with a file. See *field privilege* and *file privilege*.

Coaxial cable A connecting cable that consists of two insulating layers and two conductors.

Collision An uncontrolled attempt by more than one user to update a database at the same time.

Communication The act of sending and receiving data between workstations.

Configuration The combination of hardware and software used in a network. The hardware consists of the equipment and the way it is connected. The software consists of the programs that operate the system, the network, and applications within the system.

Data integrity A system that protects database files against data loss and prevents corruption of index files. In a LAN environment, the dBASE IV locking functions help ensure data integrity.

Data privacy The limiting of access to information. In dBASE IV, the PROTECT command is used to create and maintain data privacy. PROTECT ensures data privacy by controlling access at the file and field level through file and field privileges, and by controlling data accessibility through data encryption. See *encryption*, *field privilege*, *file privilege*, *password protection*, and *PROTECT*.

Database administrator The person or persons responsible for installing and maintaining dBASE IV on a local area network.

Deadlock An infinite loop caused when serial transaction processing is occurring and a user continuously tries to gain access to an unavailable file.

Decryption The process of deciphering data from an encoded form so that it can be read. See *encryption*.

Download To request and receive a file from another computer.

Encryption The process of enciphering or encoding data so that it can't be read.

Exclusive use A file that, once opened, can be accessed only by one user until the file is closed.

Field privilege A permission that establishes what a user can do with a field in a file. In PROTECT, one of the following field privileges can be assigned: FULL, R/O (read only), NONE. Field privileges are assigned by access level.

File access The ability to work with files. A network user's ability to access dBASE IV files is determined by the file open attribute, the file access attribute, and, if PROTECT has been used to secure a database system, a match between the user's access level and the file privilege scheme.

File access attribute A file status that determines what a user can do with a file. The file access attributes are read-only and read/write. If PROTECT is used to secure a database system, the read/write attribute can be fine-tuned. See *field privilege* and *file privilege*.

File locking A mechanism for preventing different users from gaining simultaneous update access to a shared file, thus ensuring data integrity.

File open mode A file status that determines how a file is opened. If a file is opened for exclusive use, it can be used only by one user until it is closed. If a file is opened for shared use, it can be used by more than one user at a time.

File privilege A permission or set of permissions that establishes what a user can do with an encrypted file. In PROTECT, the file privileges are DELETE, EXTEND, READ, and UPDATE.

File privilege scheme A file access control system used in a PROTECTed database system. A file privilege scheme contains the file privilege levels and the field privilege levels for a file.

File server A microcomputer and its associated hard disk on which the network sharing programs are loaded. The file server controls network communication and the definition and sharing of network resources.

Group An organization of users and of files that is matched to limit file access. Groups are identified by group name.

LAN environment A system or network of computers in which many users can operate simultaneously, sharing network resources.

Local area network (LAN) A collection of microcomputers and peripheral devices connected through cables to communicate and share resources.

Log-in security A system access validation process that limits access to users who can provide a log-in name, password, and group name.

Network Any system that communicates (sends and receives) data through cables. See *local area network*.

Network shell A special software program that sets up and manages a network.

Node See *workstation.*

Password protection A method of limiting log-in access to a network by requiring a user to enter a password. If PROTECT has been used to secure a database system, password protection occurs only if dBASE IV can locate the Dbsystem.db file in the dBASE directory.

Peripheral devices Printers, plotters, and mass storage devices connected to microcomputers.

Print spooler A buffer in the file server that controls network printing output.

Privilege See *field privilege* and *file privilege.*

Profile See *user profile.*

PROTECT A dBASE IV command that is used to create and maintain dBASE IV security, including log-in security, file access, and field access. See *encryption, group,* and *user profile.*

Record locking A software mechanism for preventing different users from gaining simultaneous update access to the same record in a shared file, thus ensuring data integrity.

Resource Any computer, device, or file that is a part of a network.

Serial transaction A database update in which records are read, modified, and saved in a consecutive manner. In a network environment, serial transactions use record locking to prevent collision.

Shared file A file that more than one user at a time can use.

Shell See *network shell.*

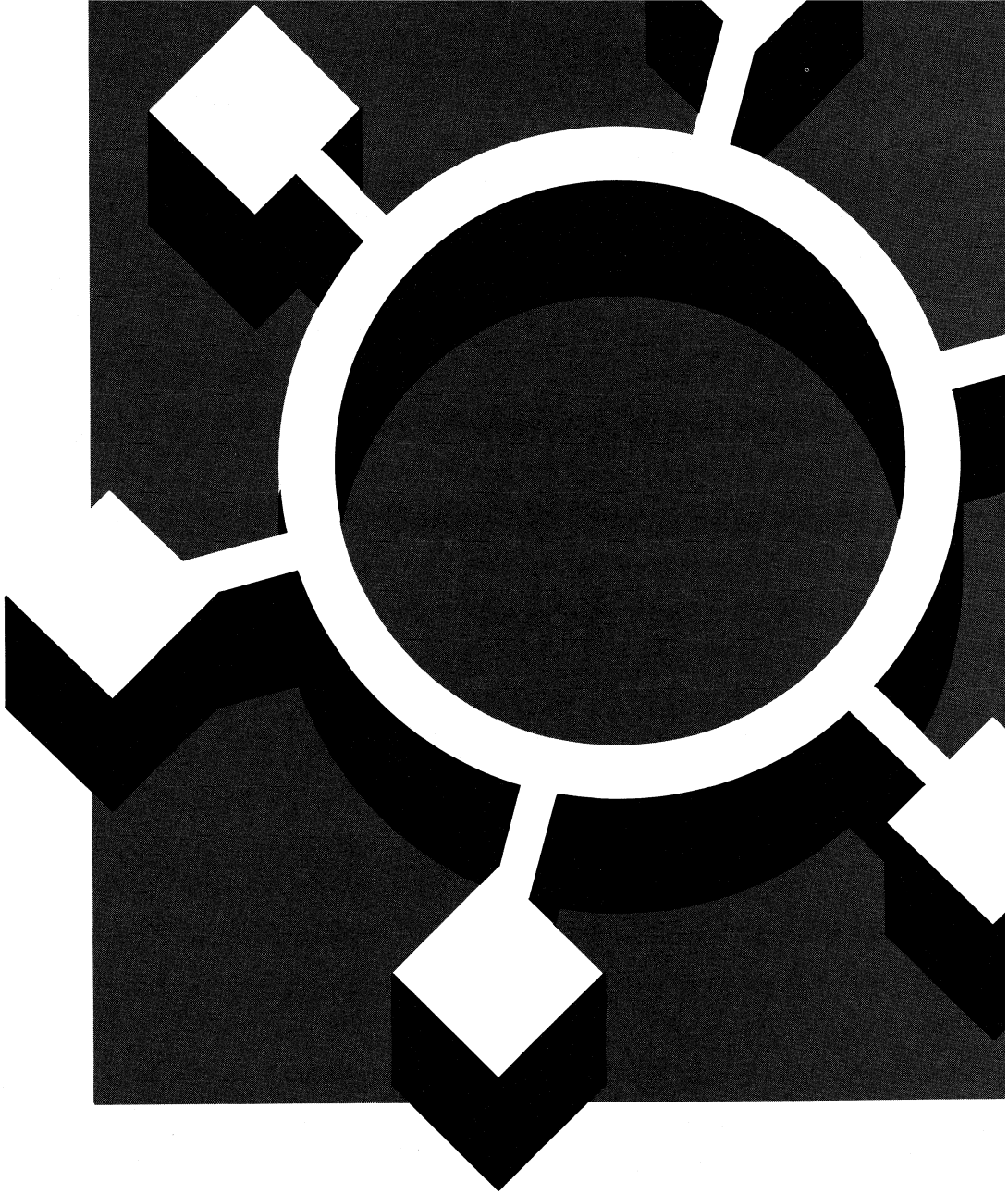
Transaction A unit of work that uses a consistent database, executes a series of updates, and produces another consistent database. While the updates are being made, the database is in an inconsistent state. If any failure occurs during the updating, the transaction can be reversed to the consistent state that existed before the updates started. In this sense, a transaction is also a unit of recovery.

Upload To transmit a file to another computer.

User profile The log-in information maintained for each user in a PROTECTed system. The log-in information consists of the user's name, password, and group name, and is kept in the Dbsystem.db file. In addition, a full name can be maintained as part of the user profile. The full name is not used as part of the log-in information.

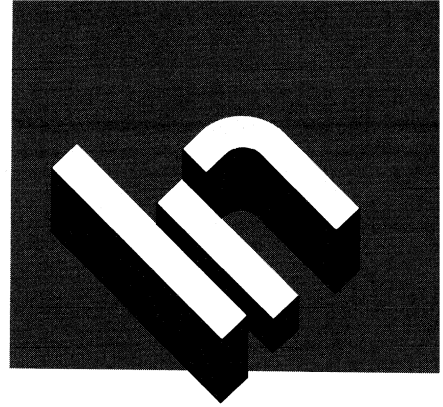
Workstation An individual personal computer connected to another within a network by a communications card and a cable.

Index



i 1 2 3 4 5 A Gl In

Index



D

E

F

S

T

U

W