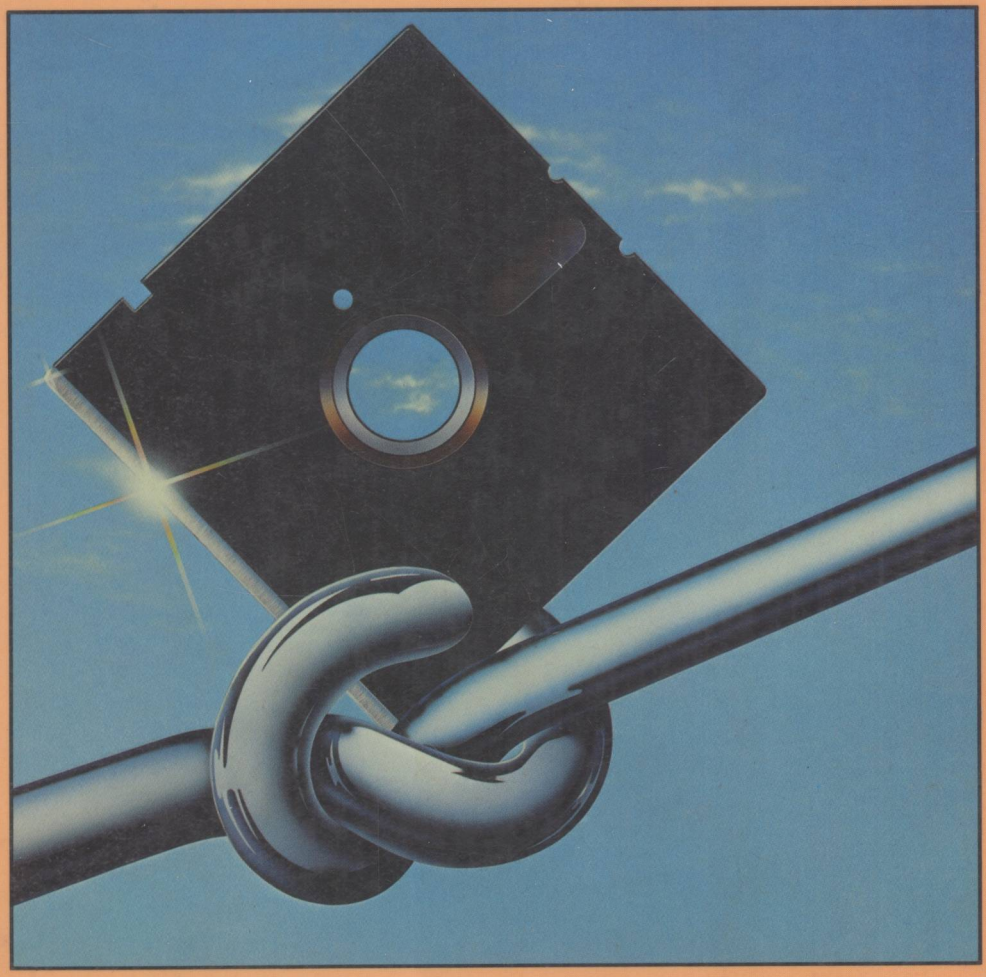
SYBEX

Advanced Techniques in dBase II™



Alan Simpson

Advanced Techniques in dBASE II™

ALAN SIMPSON

SYBEX®

Berkeley • Paris • Düsseldorf • London

Cover art by Nicolae Razumieff
Book design by Lisa Amon

ISBN 0-89588-228-0
Printed by Haddon Craftsmen
Manufactured in the United States of America
10 9 8 7 6 5 4 3

Advanced Techniques in dBASE II

ACKNOWLEDGMENTS

Thanks to Carole Alden, my editor/mentor from SYBEX, for her countless contributions to the development of this book. Also, thanks to Valerie Robbins, word processing; Jeremy Elliott, technical review; Lisa Amon, book design; Donna Scanlon, typesetting; and Dawn Amsberry, Brenda Walker, and Sarah Seaver, proofreading.

INTRODUCTION

If you have a great idea for a programming project in mind, and you want to turn it into a dBASE II software reality, then this book is written for you. In these pages you will learn how to turn great ideas into dBASE II programs that accomplish exactly what you specify with speed and efficiency.

It is not necessary that you understand every dBASE II command prior to reading this book. Some of the complex commands will be discussed in detail, particularly those not fully explained in the dBASE II User's Manual. If you do come across an unfamiliar command in one of these programs look it up in the manual to get more information.

The dBASE command is always the first word in a program line. For example, if you see the line:

```
UPDATE ON CODE FROM TRANS ADD AMOUNT REPLACE DATE
```

you can look up the UPDATE command in the dBASE II manual for more information. The first word in a dBASE sentence is the main dBASE command of the entire sentence.

dBASE II *functions* usually precede a field or memory variable name surrounded by parentheses. For example, in the line:

```
STORE STR(X,5) TO Y
```

STORE is the command, and STR is a function which changes the variable X to a character string (in this example, a character string that is five digits long). If this or any other function is unfamiliar, take the time to look it up in the dBASE manual; you'll save yourself a great deal of time in the long run.

Also, keep in mind that this book is a tutorial text, and is designed to be read as such. It starts with relatively easy programs and builds upon acquired skills to more complex programs. Therefore, if you skip

chapters one through nine, you're likely to run into quite a few surprises in Chapter 10. You should take the time to read this book from cover to cover.

To write a custom dBASE II software system, you must have several skills. First, you need to know the dBASE language. My introductory book, *Understanding dBASE II*, was designed to teach you the basic commands of dBASE II. Second, you need to know how to program. The goal of this book is to teach you how to program in dBASE II. We'll approach this goal by drawing upon three primary techniques:

1. *Software design techniques.* Designing a software system involves many steps: writing down the basic goals of the project, deciding what you need to store on the database, or on several databases, and breaking the big job down into small discrete steps, each of which is relatively easy to program. Then your programming task begins: you will write a series of small, simple *routines* which, when combined into a single *system*, work together to accomplish your goal. In this book, we'll follow these steps for creating command files.
2. *Techniques for maximum performance.* A program that accomplishes a given task in an hour may be a good program, but a program that accomplishes the same task in five minutes is preferable. Maximizing the performance of a software system is a key issue in this book. Chapter 2 discusses many techniques for maximizing the speed of dBASE programs.
3. *Fundamental algorithms.* Another programming technique is creating *algorithms*, separate routines that solve common problems, and can later be used in a variety of programming situations. For example, in Chapter 18, you will find a program that writes checks for a bookkeeping system. If you were to develop a payroll system, or accounts payable system, you could use the same check-writing program, rather than build a new one from scratch. The basic logic, or *algorithm*, for check writing is the same for all types of systems.

Why Program in dBASE II?

dBASE II is so rich in commands that you can accomplish many tasks without any programming effort whatsoever. But there are at

least three very good reasons why you should consider writing programs in the dBASE II language:

1. *Expand the capabilities of dBASE II.* Although dBASE has a variety of commands that you can type on the screen to perform a given task, these commands are far from being exhaustive. For example, if you wish to print mailing labels from your database, you will find no command to do so. You must write a dBASE command file to accomplish this task. Therefore, to get the maximum mileage from dBASE II, you will need to write some command files.

2. *Develop user-friendly systems.* By now, you've probably learned how to create and manage data on a database. Undoubtedly, you've had to invest some time and effort to do so. However, chances are that once you set up the database system, you'll want someone else to manage it so you can move on to other dBASE projects. That person may not be quite so willing to invest the time necessary to learn the dBASE II language. Therefore, you can write a series of command files that allow a novice to manage your data through a series of menus from which he or she can select options. This frees you to move on to other dBASE projects.

3. *Increase programmer performance.* If you are a professional programmer, you are probably aware of how much time it takes to develop a custom software system. However, most custom software systems must perform the same basic tasks: the user should be able to add data to a database, sort, search, and edit it, and print formatted reports from it, and perhaps add numbers to it. In lower-level languages, such as C or Pascal or COBOL or FORTRAN, the time required to write programs to perform these tasks is usually pretty hefty. dBASE can perform most of these tasks with a single command, which can cut programming time considerably.

Since dBASE II already has the routines to add new data, sort, search, edit, and print formatted reports, you can write a program that performs these tasks in a relatively short period of time. Furthermore, dBASE II offers the programmer debugging utilities that speed up the programming process even more.

There is another good reason for learning to program in dBASE II. The excellent design of the product, coupled with Ashton-Tate's outstanding marketing effort, has made the program an industry standard. Whether you are an established pro, or an aspiring one who wishes to keep up with the times, you will find lots of advantages in mastering dBASE II.

The Structure of this Book

Chapter 1 discusses general techniques for creating dBASE II command files. This includes using the MODIFY COMMAND, as well as using the WordStar and Screen Editor (SED) programs. Chapter 2 discusses general techniques for maximizing the performance of your dBASE II software.

Chapters 3 through 8 discuss a mailing list system, and cover basic programming techniques such as menus, custom screens, and custom reports. Chapters 9–13 discuss an inventory system and general programming techniques used in managing multiple data files. Chapters 14–19 discuss a bookkeeping system, with more advanced techniques for managing multiple data files. Finally, Chapters 20–21 present a library cross-referencing system, with advanced techniques for presenting menu-driven searches, keyword cross-referencing, and word wrapping.

What Version Should You Use?

To make life simple, all the software systems in this book were written in dBASE II version 2.4. If you are still using an earlier version, you should upgrade to version 2.4 because many of the bugs in earlier versions have been fixed. More capabilities have also been added to version 2.4 to simplify the programming process.

TYPOGRAPHICAL CONVENTIONS

Alt-H	Hold down Alt and H at the same time.
^D	Hold down CTRL and D at the same time.

CHAPTER 1
CREATING COMMAND FILES

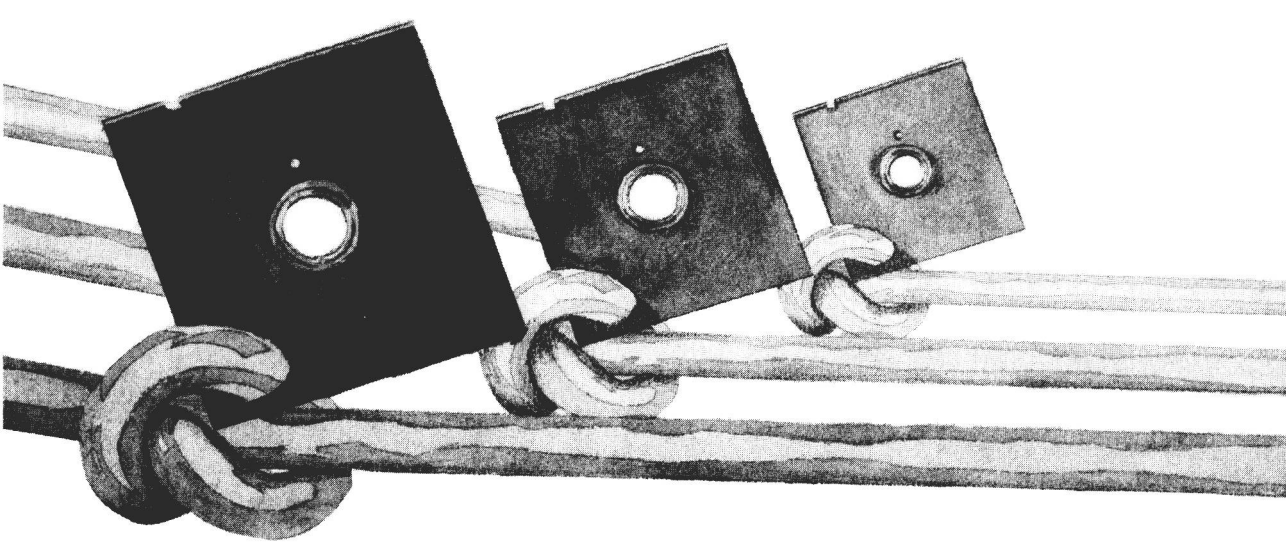


TABLE OF CONTENTS