foxtalk™
February 1990 Issue

A comprehensive monthly guide for users of FoxBASE+™ and FoxPro™

$10.00 per issue

System Security

By Pat Adams

QUESTIONS ABOUT system security, data security, and system integrity come up more and more frequently in my consulting practice, particularly as systems written in FoxBASE+ and FoxPro become more and more important to the daily work of an organization. These questions could be loosely placed in three categories:

1. Hardware and power related issues to ensure data integrity;

2. Interactive (dot prompt) access to data also utilized in custom applications when the data are on a LAN;

3. Password protection for custom systems.

Unless all three areas are addressed properly, the potential exists that data can be lost or corrupted, indexes are not updated properly as data are changed interactively, and unauthorized accesses to data occur.

Hardware and Power Related Issues

The most serious problems that can occur in a system are hardware problems and power problems that result in data being trashed, corrupted, or lost due to hardware failure such as disk crashes, power spikes, power browns, or complete loss of power. Obviously, daily backups should be done on data-sensitive systems; they are imperative on LANs. On less sensitive systems, backups should be done weekly at the minimum. With proper backup, if problems do occur a minimum of information will be lost.

My recommendation, by the way, is to maintain at least three generations of backups just in case problems are encountered during a restore; this can lead to a frequently encountered problem. All too often I encounter situations in which the client religiously performs backups without ever testing the restore capabilities of the backup. No testing that is, until problems arise and a restore is necessary. Only at that time does the client discover that the backup system has problems that result in the inability to complete the restore, or realize that he has been running the backup incorrectly, and, as a result, the restore cannot be made.

Simply making regular backups is not sufficient. When a new backup system (hardware and/or software) is put in place, take the time to do several backups and then attempt to restore from those backups immediately to ascertain whether the system

(continues on page 3)

— Inside —

foxtalk™

A comprehensive monthly guide for users of FoxBASE+™

foxtalk (ISSN#1042-6302) is published monthly (12 times per year) by Pinnacle Publishing, Inc., 28621 Pacific Highway South, Federal Way, WA 98003. Cost of domestic subscriptions: 12 issues, $99.00; 24 issues, $180.00. Outside USA: 12 issues, $119.00; 24 issues, $210.00. Single copy price: $10.00; outside USA, $12.00. Individual source code disks: $10.00; 12 source code disks, $60.00. All funds must be in U.S. currency. Back issues are available upon request, for the same price as a single copy. Second class postage pending at Auburn, WA. POSTMASTER: Send address changes to **fox**talk PO Box 8099, Federal Way, WA 98003.

Address all editorial correspondence, requests for special permission, or bulk orders to The Editor, **fox**talk PO Box 8099, Federal Way, WA 98003. Phone: 1 800 231-1293 or 206/941-2300.

Questions of a technical nature can be directed to Glenn A. Hart, Editor, care of Pinnacle Publishing, Inc., PO Box 8099, Federal Way, WA 98003.

Publisher, David M. Johnson
Editor, Glenn A. Hart
Managing Editor, Jan B. Seymour
Senior In-House Editor, Brent P. Smith

Volume II, Issue 2

Editorial

It's said that when you reach my age (no, I *won't* tell you what it is, but the bloom is significantly off the rose!), birthdays shouldn't matter as much as they did in one's youth. Well, I may not have quite the blowout parties as in the past, but I still find birthdays, both mine and, especially, my wife and children's, festive and meaningful occasions.

So I take great pleasure in **fox**talk reaching its first birthday. As with any new enterprise, there have been both ups and downs, but the ups have been greatly in the majority. Subscriptions have more than doubled, and **fox**talk remains Pinnacle's largest journal by far. I'm not surprised, really, given the enthusiasm and professionalism that marks both Fox Software and its community of users.

We've tried to bring you the best code, the most useful ideas, the most impressive programs possible, while juggling the needs of an audience with a broad spectrum of skills and interests. I'd like to think that for the most part we've succeeded, and your letters (and subscriptions!) seem to support this assessment. We can always improve, though, and I'm tremendously encouraged at the number of new contributors joining the **fox**talk lineup. We plan to have more articles on the Mac, begin discussing template languages, and otherwise broaden our coverage of both Fox and third-party products. We really want (and *need*) your contributions! Those monthly deadlines come around with frightening regularity (one of the things I've learned this year is that it's *much* easier to be a writer than an Editor!), and the more input we have the better.

This will be an exciting year for Foxies, with FoxPro 1.1, with its many new features and EXE compiler on the horizon, and, perhaps, new graphic and text mode platforms as well. We'll be covering it all, helping you to stay on top of the stimulating developments ahead.

Thanks again for your vote of confidence in **fox**talk. May our second year be even more fantastic than the first!

On to This Month's Goodies...

• This month's Pat Adams spectacular once again shares Pat's invaluable experience in the real world of major league clients. I've long held that the computer industry is guilty of almost scandalous negligence in not emphasizing backups–both data systems like tape cartridges, optical or the emerging DAT tapes, and power systems like uninterruptable power supplies. I had always thought dealers were in business to make money, and high profit items like these should be a worthwhile

(continues on page 24)

System Security Cont.

is working well. As the backup system continues to be utilized, do periodic restores to make sure everything is still in good working order. On sensitive LAN systems, I recommend this interim testing be done each month.

All too often this type of testing is overlooked in the press of daily work pressures; however, loss of a company's accounting records, orders, or other important information can bring company business to a complete halt.

The recent San Francisco earthquake pointed out not only the importance of good backup systems, but also the importance of storing periodic backups off-site. It does no good to back up data religiously if the backup is stored at the same location as the system itself, and a fire, earthquake, flood, or other disaster destroys the site.

Mainframe system operators have followed this off-site storage rule for years, and in the case of such disasters have been able to get their systems back up and running quickly. PC and PC-LAN systems should be treated in the same manner.

Another component of mainframe and mini systems that is all too often overlooked in PC and PC-LAN systems is power. All mainframe and most mini systems have some variety of uninterruptable power supplies (UPS); however, the majority of PC systems do not. Are the data on PC and PC-LAN systems less important than data mainframes? I think not.

When most people think of power problems they usually think of power spikes or complete power blackouts. Blackouts, however, are infrequent. Power dips or brownouts, on the other hand, are very frequent and can also be very subtle. Because of the nature of the .DBF file format, the buffering in FoxBASE/FoxPro, and DOS buffers, power brownouts can play havoc with your systems and data. This kind of problem is more frequent in the hot summer months when air conditioners create a tremendous drain on the municipal power systems, but they do occur year round.

Every summer I receive many calls from clients, friends, and complete strangers who are mystified by upper and lower ASCII characters such as smiling faces, hearts, and musical notes — not to mention erroneous end of file markers — suddenly appearing in their data files. Even though they may manually correct the offending records, the situation reappears later in other records. I call this the summer power brownout blues. The callers are often unhappy to learn that the problems are caused directly by power brownouts and that the only solution is to purchase a true UPS.

During the summer months some cities purposely run the power lower than the standard 110. For example, two years ago in New York City, monitoring and recording meters installed by one client proved that the City was continually running the power at 105 instead of 110, although the City did not inform anyone of this. During the summer of 1989, New York City officials announced that, if the need arose, they would be running the power at a lower level to avoid blackouts. Such lowered power sources not only have the potential to corrupt data, but also to weaken chips and other components in PCs.

Another frequent occurrence is one that many do not realize can happen — insufficient power on a particular circuit. The conventional wisdom says that when a circuit is overloaded, a fuse will blow or a circuit breaker will shut off; as is often the case, conventional wisdom is wrong here. It is quite possible that a circuit can become marginally overloaded when too many electrical appliances are plugged into the circuit; at this marginal level a fuse will not be blown but, rather, insufficient or brown levels of electricity will be delivered to one or more of the appliances in the circuit. For computers the result is erratic behavior, and for .DBF files and indexes the result can be corruption.

It is absolutely necessary that every server on a LAN have its own true UPS so these power problems can be avoided. In my own office, every time I order a new computer I automatically include a true UPS as part of the hardware components. Even though I have had two power lines installed specifically for computer and peripherals in my office, I have found that, over the years, as PCs become more powerful, and as printers — such as the Hewlett-Packard LaserJets — consume more power, they put a heavier and heavier demand on power lines. Without the true UPSs, I have experienced occasional glitches.

You may be wondering what the difference is between a UPS and a true UPS, other than dollars. Most UPS systems are battery stand-by UPSs that will engage in milliseconds if the power goes out or if the power level becomes too low. When these UPSs do engage, they begin supplying power from their batteries and thus provide time to close a system down without damage. Unfortunately, even the very small amount of time required for this switch over of power can be sufficient to corrupt data in .DBF files or indexes. Furthermore, battery stand-by UPSs do not provide total power conditioning, because they will only kick in when the power becomes drastically low. A true UPS, on the other hand, pulls its power from the municipal system, utilizes that power to constantly recharge its batteries, and feeds power to the computer from its batteries, thus providing

(continues)

total power conditioning 100% of the time. Both power brownouts and power spikes are avoided, and, of course, in the event of a power blackout, power is available while the system is shut down properly.

True UPSs used to be quite expensive, with prices beginning around $2,000.00 and up, and the units were quite bulky. Luckily, the past two years have seen some excellent advances both in the technology and in price. I use a true UPS called the Liebert PC-ET that looks rather like a slightly oversized Kensington Masterpiece Power Pad. It will fit between the CPU and the monitor. Devices are plugged into the rear of the UPS and the front contains five on/off switches, one for each plug, as well as a master on/off switch for the entire UPS. Three of the ports are totally UPS-protected (Liebert has been purchased by Emerson Electric and the same product is also available under the Emerson label). Prices vary on this UPS, but normally range from $575.00 to $850.00. It is more than sufficient to handle a heavily loaded fast 386 CPU and a power-hungry high-end color monitor such as a NEC Multisync or a Sony HG. Liebert/Emerson are now selling other types of true UPS devices at slightly lower prices.

Another important issue in hardware integrity is that of redundancy. Is it necessary to duplicate the PC or the server in case the system goes down? Consider what would happen to the organization if the hardware goes down for any reason and cannot be brought back up quickly. If it is determined that business operations would be seriously impacted, then a stand-by PC for emergency use is in order for single-user systems and a mirrored server system or duplicate LAN server is appropriate for LAN systems. The cost-justification here is very straightforward: compare what it would cost the company if the computer system is down for a day, a week, or a month with the price of the duplicate hardware.

Don't limit hardware redundancy to just the PC or the server. If you are using a tape system or a WORM system for backup, consider what would happen if that hardware were to go down and could not be brought back up quickly. Without the backup mechanism available, it is not possible to do routine backups, and if the PC or server crashes without proper backups having recently been made, all current work will be lost. What would this cost your organization?

Lan Security Strategies

There are many different avenues that can be utilized to help provide security for general LAN access as well as access to specific applications and data sets on LANs. All too often, however, these strategies become a tug of war between the users and MIS. Users want access to the LAN, applications on the LAN, and data sets on the LAN to be easy; MIS, on the other hand, is charged with ensuring LAN and data security, which, of course, means — from the user's point of view — making access difficult.

The computer literate people in MIS do not view having to enter a user logon name and a password as difficult, but end users often find it confusing and/or tedious and time-consuming. It's relatively easy to inform support staff that such login routines are company policy, but management will often complain about it with a much more forceful voice. Some organizations handle this situation by providing every new LAN user with a written copy of the company policy, written instructions on how to log onto the LAN, and a letter or brochure that explains the importance of such security. The reality is that a LAN that does not require user login and passwords is at risk.

Many LAN administrators do not take advantage of a feature available in higher-end LAN software that permits them to limit any user access to only one workstation or terminal at a time. If this is not done, the usual circumstance is that users will share their user login name and password with others. However, if the user knows that she will not be able to access the LAN if a friend happens to be using her login name and password, the user is much less likely to share that information with others. Another strategy is to require that passwords be changed monthly. In this instance, users should not be able to utilize their previous password nor any password they have used in the past six months to a year.

Once the user has gained access to the LAN, there are a number of other strategies that can be utilized, both globally and on a user by user basis, to limit user access and the potential havoc a user might create on a LAN. The most important item — often overlooked by LAN administrators — is *not* to put FORMAT.COM on the LAN! If, for any reason, there is a specific need to have FORMAT.COM on the LAN, rename it and create a batch file called FORMAT.BAT that will take the user into a set-up where he cannot possibly inadvertently format the server. This strategy also works well on stand-alone PCs, by the way. I use the following set-up on my own systems as well as on client systems:

FORMAT.COM has been renamed to DONOTUSE.COM and placed in a subdirectory named \FMT. Furthermore, the DOS attribute has been changed so that the DONOTUSE.COM file is hidden. A series of DOS batch files, beginning with FORMAT.BAT, are then called

(continues)

whenever a user types FORMAT, FORMAT A: or any similar command. FORMAT.BAT types a text file onto the screen with selections for the user to make about what type of disk he wishes to format and the drive in which the floppy disk is located. Selections in the following examples are A, B, and C, and there are corresponding DOS batch files called A.BAT, B.BAT, and C.BAT that issue the appropriate commands.

```
* FORMAT.BAT
ECHO OFF
CLS
TYPE FTEXT

* FTEXT

          DO YOU WANT TO FORMAT:

          A.  A high-density disk in Drive A:
          B.  A 3-1/2 inch 1.4 mb. disk in Drive B:
          C.  A 3-1/2 inch 720 k disk in Drive B:


             PLEASE ENTER THE LETTER OF YOUR CHOICE
                       AND PRESS RETURN

* A.BAT
cd \fmt
DONOTUSE A:/V
CD \
CLS

* B.BAT
CD \FMT
DONOTUSE B: /V
CD \

* C.BAT
CD \FMT
DONOTUSE b: /n:9 /t:80 /V
CD \
```

There are, of course, many different ways to accomplish this with a single DOS batch file that utilizes the more advanced mini-programming language in DOS. I set this system up a number of years ago and although it is a bit simplistic, it works well. Do remember, however, to change the DOS attributes of all these batch files to read only, to avoid someone accidentally erasing them or modifying them.

Most LAN software permits the LAN administrator to limit access, on a user by user basis, to different volumes, systems, or subdirectories on the LAN, and this is the next line of defense. The initial set-up is tedious, and, the more users on the LAN, the more tedious. However, if Susie Smith has no business utilizing files in a certain area, chances are, sooner or later, she will start looking around just to see what is there. If the user has not been locked out of certain areas, another level of security is lost.

In addition to locking users out of specific areas on a user by user basis, most highend LAN software permits the LAN administrator to assign user "rights" for those areas in which they are enabled. For example, Novell Netware utilizes the GRANT <options> FOR <directory, path> TO <user name, user group> command with the following options:

```
Read
Open
Search
Write
Create
Delete
Modify
Parental
No Rights
All Rights
```

These "rights" are not always what they might seem on the surface. While a user may not have the authorization to create a new file, that same user may be able to open and read the information in the file but not change it, or open the file, read it, and make changes.

In terms specific to FoxBASE/FoxPro, the Novell Read "right" means that, even though the user does not have write authorization for the subdirectory in which the file is located, the user can open a .DBF file and read the information in the file, use the COPY TO command to copy records to another area of the server where they have write authorization, run reports, SUMs, set filters, and other activities that do not involve actually writing to the database record or file. Indexes, on the other hand, require that the user have both read and write authorization to be used — even if no changes are made.

Experienced LAN managers often take advantage of this to provide yet one more level of data security. Database files are placed in one subdirectory while the indexes are located in a different subdirectory. Users who only have a need to view records in a database, create queries or reports, perform searches and other activities that do not require editing of the database files or records, are given Read authorization to the subdirectory in which the .DBF files are located, and both Read and Write authorization to the subdirectory containing the indexes. This lowers the exposure for potential corruption or damage to the database files.

Although I have discussed a few of the details of the Novell LAN software, the majority of the full-featured LANs have similar capabilities. Novell and the other LAN software packages also provide many other capabilities for strengthening LAN security that can be tailored to your particular needs.

(continues)

Standalone PC Security Strategies

One of the most difficult tasks is avoiding unauthorized access to PCs that hold important or sensitive data. In most circumstances, anyone can boot a PC from drive A: with a floppy disk and then log onto the hard disk from there, thus accessing the data. There are systems that combine hardware and software to preclude booting from drive A:. They require the use of a password to access the hard disk upon booting. However, this can, and frequently does, cause considerable problems. A number of my clients tried these systems and found they worked quite well on the surface. The problem was that all too often the management procedures of the systems were sloppy and only one person knew the password. When that person was out of the office or ceased to be employed by the company, the company could not gain access to its own data. And in one instance, the custodian of the password forgot it! These systems are so secure that attempts to gain access to the hard disk often result in the information on the disk being corrupted. Additionally, some of these systems preclude the use of certain types of backups, thus creating even more problems.

Initially, the concept of such systems is appealing, but I recommend you not utilize them because of the serious problems I have observed when they are actually placed in use. Granted, the problems are people-generated, not security system-generated. However, regardless of the cause, the end result is that the company cannot access its data.

In my opinion, the single best first line of defense for a standalone PC is the physical lock and key system present in the 286 and 386 class machines. These machines normally come with two sets of keys, so be sure that the department administrator or a similarly responsible person has one key to facilitate access in the event the person normally responsible for starting the machine on a daily basis is not available.

Physically locking a machine still does not prevent problems that arise when a PC is stolen, nor does it prevent a particularly enterprising person from picking the lock on the PC or opening up the CPU and stealing the hard disk itself. Removable disk media such as Bernoulli cartridges and the Plus Development Passports, however, can be taken out of the PC each evening and stored away in a safe place to avoid such problems. Some security-cleared areas in which I have worked mandate this type of security, by the way.

There are many methods to foil the casual intruder and those who are not quite as knowledgeable as they might think. Two of my favorites are using the upper ASCII characters to name subdirectories and changing the DOS attribute to hidden.

To include upper ASCII characters such as a blank space, [CHR(255)], or a dash, CHR(196), on screen, or CHR(205), which will give the screen appearance of an equals sign, or others, simply make sure Num Lock is turned on, hold down the Alt key, and type in the ASCII value. This technique can be used to name a subdirectory and provide a small measure of security from the casual browser. Of course, if utilities such as Norton Utilities, X-Tree, and other similar products are utilized, they can easily access subdirectories named in this manner, but the technique will foil the casual browser who can often do serious damage.

.DBF Files and the DOS Attributes

Over the past several years I have averaged approximately one consultation per month on security issues. In all but one of those meetings, the clients were unaware that the DOS attribute of .DBF files can be changed to hidden and the .DBF file will still be accessible via the FoxBASE/FoxPro USE command. This will not fool the knowledgeable "cracker" and utilities such as those mentioned above will show the presence of the files. However, it does provide a measure of security and also avoids the accidental erasure of the file(s). It's particularly helpful in telecommunications situations since most hard disk utilities will not function on a remote basis.

Password and User log Security Systems

Password and user login/out security systems written in FoxBASE+ and FoxPro can be built into any custom system to provide an additional measure of security. There are many different ways to design, structure, and code such systems, but for the purposes of this article, we'll take a look at a fairly simply one running on a Novell LAN. The system consists of a SECURITY database that contains the user's LAN login name, his full name, a field called RIGHTS that contains a code for the user's authorization level, and the user's password. The user may change her own password on demand via a selection from the main menu of the system. The user is required to provide the correct password before access to the main menu of the system — or any other portion — can be obtained. The password is upper and lower case sensitive. Three tries are permitted before the user is bounced out of the ballpark. The system automatically retrieves the user's Novell login name with the Fox GETENV()

(continues)

function and compares it against the password to ensure a second level of security.

In addition, there is a USERLOG file that automatically records the date and time the user logged in (the Novell login name is again utilized) as well as the date and time the user logged out from the system. A user is only permitted to have one active session at a time. This provides a tertiary level of security and also has some additional side benefits that LAN administrators particularly appreciate. If a user attempts to leave the system by simply turning off his machine or by doing a warm boot, the user does not get logged out of the system. Because the user's record in the USERLOG file is still open, the next time the user attempts to use the system he is greeted with a message to call the LAN administrator. The LAN administrator can then query the user about why he has an open log and discuss the problems caused by incorrect egress from the system. Because such activity inconveniences the user, users become less likely to leave the system incorrectly, and therefore, the chance of the LAN being brought down by a user reboot or power down is lessened.

This system is purposely written without a menu selection for the LAN administrator or anyone else to access the USERLOG file, and the USERLOG file is placed in a subdirectory to which only the LAN administrator has full rights. In other words, it is intentionally structured so the LAN administrator — and only the LAN administrator — must go in interactively (i.e., from the dot prompt or command window) and edit the appropriate record in the USERLOG file before the user can access the system once again. While this system structure was created at the user's request, it could be easily modified to include a special menu selection for less sophisticated LAN administrators.

If you would like to test this system without a Novell LAN, simply comment out the GETENV() statement and initialize a public memvar with a username in it for the xuser memvar in its place.

The file structures for the SECURITY.DBF and USERLOG.DBF are:

```
Structure for database: SECURITY.DBF

Field  Field Name  Type       Width    Dec

    1  USERNAME    Character      8
    2  PASSWORD    Character      8
    3  FIRST       Character     12
    4  MI          Character      1
    5  LAST        Character     20
    6  DEPT        Character     15
    7  RIGHTS      Character      1
** Total **                      66
```

```
Structure for database: USERLOG.DBF

Field  Field Name  Type       Width    Dec

    1  USERNAME    Character      8
    2  DATE_IN     Date           8
    3  TIME_IN     Character      8
    4  DATE_OUT    Date           8
    5  TIME_OUT    Character      8
** Total **                      41
```

```
*:*********************************************************
*:
*:        Program: TOPFILE.PRG
*:
*:         System: SECURITY PASSWORD & LOG SYSTEM
*:         Author: Pat Adams
*:      Copyright (c) 1990, DB Unlimited
*:  Last modified: 02/06/90     14:18
*:
*:          Calls: ERRORCK.PRG
*:               : BADBOY2.PRG
*:               : BADBOY.PRG
*:               : TRYAGAIN.PRG
*:               : YOUR_OUT.PRG
*:               : LOOKING.PRG
*:               : BONKERS.PRG
*:
*:           Uses: USERLOG.DBF
*:               : SECURITY.DBF
*:
*:        Indexes: USERLOG.IDX
*:               : SECURITY.IDX
*:
*:      Documented 02/06/90 at 14:36  SNAP!  version 4.02
*:*********************************************************
* TOPFILE.PRG
*
*     This is the top or first file called for the system.
*     It gets the user name from the Novell LAN with the
*     GETENV() function, requires the user to enter his/her
*     password and compares to ensure these match.  User is
*     allowed three attempts to enter the password correctly.
*
*     If the password is not entered correctly or the passwor
*     does not match the user's LAN name the user is bounced
*     back to DOS.
*
*     If the user name and password do match, the USERLOG
*     file is checked to determine if the user has a currentl
*     active log-in record.  If so, the user is advised that
*     records indicate they have a current log-in session fro
*     the time and date recorded in the USERLOG file.  User
*     must contact the LAN administrator to have that open
*     log-in session canceled.
*
*     This system is written for FoxBASE+ 2.1 or higher or
*     FoxPro M/U 1.0 or higher and Novell LANs.
*
*     Author: Pat Adams, DB Unlimited     (718) 469-4032
*   Date: 2/3/90
*   Copyright 1990 DB Unlimited
*
*     All rights reserved
*
* * * * * * *
*
SET ESCAPE OFF
SET STATUS OFF
```

(continues)

```
SET SCOREBOARD OFF
SET EXACT ON
SET EXCLUSIVE OFF
SET TALK OFF
SET DELETED ON
?? SYS(2002)
* ================================
* Set up error handling system.  This
* will catch any problems that might
* result from the necessary files or
* indexes being unavailable.
* ================================
ON ERROR DO errorck WITH ERROR(), MESSAGE(), MESSAGE(1), ;
    SYS(16), SYS(2001, "CONSOLE")
* ================================
* Initialize public memory variable
* for user name to be utilized
* throughout the system.
* ================================
PUBLIC xuser
STORE SPACE(8) TO xuser
SELECT 2
USE \userlog\userlog INDEX \userlog\userlog
* ============================
* Index is UPPER(username)
* ============================
SELECT 1
USE security INDEX security
* ==========================
* Index is UPPER(password)
* ==========================
STORE ALLTRIM(GETENV("USER")) TO xuser
* ====================================
* The above line of code gets the Novell user
* login name and stores it to a memory variable.
* The ALLTRIM function is specific to FoxPro but
* can be simulated in FoxBASE+ by a UDF or the
* following line of code can be substituted for
* FoxBASE+ applications:
*
*     STORE LTRIM(TRIM(GETENV("USER"))) TO xuser
* ====================================

IF LEN(xuser) < 8
    * ================================
    * The checks require exact matches.
    * Therefore, if the user name length
    * is less than 8, pad it with blank
    * spaces to ensure an exact match.
    *
    * Note that the assumption here is the
    * LAN software will not permit a user
    * name of greater than 8 characters.
    * ================================
    xuser = xuser + SPACE(8 - (LEN(xuser)))
ENDIF LEN(xuser) < 8

SET COLOR TO W/N
CLEAR
STORE 0 TO kount

DO WHILE kount < 3
    kount = kount + 1
    SET COLOR TO GR+/B, B/B, N
    * ================================
    * Initialize color settings so that the
    * 'GET' will be blue on blue and the
    * user input will not be visible on the
    * screen.
    * ================================
    @ 7,22,13,55 BOX"╔═╗║ ║╚═╝"
    SET COLOR TO W+/B
    @ 8,23 CLEAR TO 12,54
    @ 8,26 SAY "PLEASE ENTER YOUR PASSWORD"
    SET COLOR TO W/B
    @ 10,25 SAY "(Upper and lower case count)"
    STORE SPACE(8) TO xpass
    @ 12,26 GET xpass
    READ

    IF kount <= 3
        * ============================
        * User has three attempts to enter the
        * correct password.  Check to see if
        * password matches that on file for user.
        * ============================
        SELECT security
        SEEK xpass

        IF FOUND()
            * ======================
            * If the password is found compare
            * against the user name
            * ======================
            IF security->username = m->xpass
                * ============================
                *$ Correct password has been entered
                *  and matched with user name.  Permit
                *  user to enter the system.
                * ============================
                EXIT
            ELSE      && Else for security->username = m->mpass
                IF kount = 3
                    * ======================
                    * If user name does not match the
                    * password and user has exhausted
                    * all tries, then bounce them out to DOS.
                    *
                    *  This routine can be modified to put
                    *  the user name, date, and time of the
                  *  attempted security breach into a database
                    *  for use by the LAN administrator.
                    * ======================
                    DO badboy2
                    CLOSE DATA
                    ?? SYS(2002,1)
                    QUIT
                ELSE                    && Else for kount = 3
                    * ======================
                    *$ Inform user password does not
                    *  match the user name and provide
                    *  another try.
                    * ======================
                    DO badboy
                ENDIF kount = 3
            ENDIF security->username = m->xpass
        ELSE                    && IF FOUND() for xpass
            IF kount < 3
                * ======================
                *$ Inform user password is not correct
                *  and provide another  try.
                * ======================
                DO tryagain WITH SUBSTR(kount,1,0)
            ELSE
                * ======================
                *$ Inform user they have exhausted
                *$ the password attempts and bounce
                *  them out to DOS.
                * ======================
                DO your_out
                CLOSE DATA
                ?? SYS(2002,1)
                QUIT
```

(continues)

```
            ENDIF kount < 3
        ENDIF found()
    ENDIF kount <= 3
ENDDO while kount < 3

* ================================
*$ Now check to see if user has an open log
*  record in the USERLOG.DBF file
* ================================
DO looking
SELECT userlog
SEEK xuser

IF FOUND()
    LOCATE WHILE UPPER(username) = xuser FOR EMPTY(date_out)

    IF FOUND()
        * ====================
        *$ User has an open USERLOG.DBF record, which
        * means they logged out incorrectly.  Advise
        * user and bounce them back to DOS.
        * ====================
        DO bonkers
        ?? SYS(2002,1)
        CLOSE DATA
        QUIT
    ENDIF found()
ENDIF found()

CLOSE DATA
* ================================
* User has correctly entered password and
* password matches the user name in the LAN.
* Permit user to enter the system.
* ================================
DO mainmenu
* END Topfile.Prg


*:*********************************************************
*:
*:        Program: BADBOY.PRG
*:
*:         System: SECURITY PASSWORD & LOG SYSTEM
*:         Author: Pat Adams
*:      Copyright (c) 1990, DB Unlimited
*:  Last modified: 02/06/90     13:56
*:
*:      Called by: TOPFILE.PRG
*:
*:      Documented 02/06/90 at 14:37   SNAP!  version 4.02
*:*********************************************************
* BADBOY.PRG
*
*     Inform user the password entered does not match
*     his/her LAN user name.
*
*     Author: Pat Adams, DB Unlimited     (718) 469-4032
*   Date: 2/5/90
*   Copyright 1990 DB Unlimited
*
*     All rights reserved.
*
* * * * * * * *
*
SET COLOR TO BR+*/N
@ 9,35,17,68 BOX"┌─┐│┘─└│"
SET COLOR TO BG+/N
@10,36 CLEAR TO 16,67
@12,37 SAY "The password you entered does"
@13,37 SAY "not match your LAN user name."
SET COLOR TO BR+*/N
@10,46 CLEAR TO 10,55
@10,47 SAY "BAD NEWS!"
SET COLOR TO W/N
@16,37 CLEAR TO 16,65
@16,37 SAY "Press any key to try again..."
WAIT ""
RETURN
* END Badboy.Prg


*:*********************************************************
*:
*:        Program: BADBOY2.PRG
*:
*:         System: SECURITY PASSWORD & LOG SYSTEM
*:         Author: Pat Adams
*:      Copyright (c) 1990, DB Unlimited
*:  Last modified: 02/06/90     14:08
*:
*:      Called by: TOPFILE.PRG
*:
*:      Documented 02/06/90 at 14:37   SNAP!  version 4.02
*:*********************************************************
* BADBOY2.PRG
*
*     Password entered by user still does not match user
*     LAN name and this is the third and last try.  Inform
*     user to contact LAN administrator immediately and
*     then bounce user to DOS.
*
*     Author: Pat Adams, DB Unlimited     (718) 469-4032
*   Date: 2/5/90
*   Copyright 1990 DB Unlimited
*
*     All rights reserved.
* * * * * * *
*
SET COLOR TO R+*/N
@10, 9,19,41 BOX"┌─┐│┘─└│"
SET COLOR TO W+/N
@11,10 CLEAR TO 18,40
@11,12 SAY "The password you entered"
@12,12 SAY "does not match your LAN"
@13,12 SAY "user name and that was your"
@14,12 SAY "third and last try.  You"
@15,12 SAY "can"+SPACE(5)+"access this system!"
@17,12 SAY "Please call your LAN"
@18,12 SAY "administrator immediately!"
SET COLOR TO R+*/N
@15,16 CLEAR TO 15,18
@15,16 SAY "NOT"
?? CHR(7) + CHR(7) + CHR(7) + CHR(7)
xwait = INKEY(8)
RETURN
* END badboy2.prg


*:*********************************************************
*:
*:        Program: TRYAGAIN.PRG
*:
*:         System: SECURITY PASSWORD & LOG SYSTEM
*:         Author: Pat Adams
*:      Copyright (c) 1990, DB Unlimited
*:  Last modified: 02/06/90     13:40
*:
*:      Called by: TOPFILE.PRG
*:
*:      Documented 02/06/90 at 14:37   SNAP!  version 4.02
*:*********************************************************
```

(continues)

```
* TRYAGAIN.PRG
*
*     Message screen to advise user s/he has not entered
*     the correct password and should try again.
*
*     The expected parameter is the character value
*     for the number of attempts utilized.
*
*     Author: Pat Adams, DB Unlimited     (718) 469-4032
*   Date: 2/5/90
*   Copyright 1990 DB Unlimited
*
*     All rights reserved.
*
* * * * * * *
*
PARA xused
xremain = SUBSTR((3 - VAL(xused)),1,0)
SET COLOR TO R+*/W
@11,39,21,67 BOX "┌─┐│┘─└│"
SET COLOR TO B/W
@12,40 CLEAR TO 20,66
@14,41 SAY "That wasn't the correct"
@15,41 SAY "password.  This was your"
@16,41 SAY "&xused attempt; you have &xremain"
@17,41 SAY "more chances to enter the"
@18,41 SAY "correct password."
SET COLOR TO R*/W
@12,51 CLEAR TO 12,56
@12,51 SAY "SORRY!"
SET COLOR TO B+/W
@20,41 CLEAR TO 20,56
@20,41 SAY "Press any key..."
WAIT ""
RETURN
* END Tryagain.Prg


*:*********************************************************
*:
*:        Program: LOOKING.PRG
*:
*:         System: SECURITY PASSWORD & LOG SYSTEM
*:         Author: Pat Adams
*:      Copyright (c) 1990, DB Unlimited
*:  Last modified: 02/06/90     14:22
*:
*:      Called by: TOPFILE.PRG
*:
*:      Documented 02/06/90 at 14:37   SNAP!  version 4.02
*:*********************************************************
* LOOKING.PRG
*
*     Pop up screen to inform user a search is
*     being performed on the USERLOG.DBF file.
*
*     Author: Pat Adams, DB Unlimited     (718) 469-4032
*   Date: 2/5/90
*   Copyright 1990 DB Unlimited
*
*     All rights reserved.
*
* * * * * *
*
SET COLOR TO W/N
CLEAR
SET COLOR TO BR+/N
@ 6,22,12,55 BOX"┌─┐│┘─└│ "
SET COLOR TO BG+/N
@ 7,23 SAY SPACE(32)
@ 8,23 SAY "  Checking user log file."+SPACE(7)
@ 9,23 SAY SPACE(32)
@10,23 SAY "  Sorry to keep you waiting...  "
@11,23 SAY SPACE(32)
xwait = INKEY(2)
RETURN
* END Looking.Prg


*:*********************************************************
*:
*:        Program: BONKERS.PRG
*:
*:         System: SECURITY PASSWORD & LOG SYSTEM
*:         Author: Pat Adams
*:      Copyright (c) 1990, DB Unlimited
*:  Last modified: 02/06/90     14:32
*:
*:      Called by: TOPFILE.PRG
*:
*:      Documented 02/06/90 at 14:37   SNAP!  version 4.02
*:*********************************************************
* BONKERS.PRG
*
*     The USERLOG.DBF shows a record still open for
*     this user, which probably means s/he "exited"
*     the system improperly by doing a warm reboot
*     or turning off the power.  Inform user they will
*     need to contact the LAN administrator to have the
*     log record closed and can not access the system
*     until this has been accomplished.  This provides
*     a good way to discourage users from rebooting or
*     powering down and provides the LAN administrator
*     with the opportunity to discuss the situation with
*     repeat offenders.
*
*     Author: Pat Adams, DB Unlimited     (718) 469-4032
*   Date: 2/5/90
*   Copyright 1990 DB Unlimited
*
*     All rights reserved
*
* * * * * * *
*
SET COLOR TO W/N
CLEAR
SET COLOR TO BG+/N
@ 5,17,18,61 BOX "┌─┐│┘─└│"
SET COLOR TO BR+/N
@ 6,20,17,58 BOX"┌─┐│┘─└│ "
SET COLOR TO W+/N
@ 7,21 CLEAR TO 16,57
@ 7,23 SAY "The record indicates you did not"
@ 8,23 SAY "log off correctly last time you"
@ 9,23 SAY "used the system and your user"
@10,23 SAY "log is still open.   You must"
@11,23 SAY "contact your LAN administrator"
@12,23 SAY "and request that your log be"
@13,23 SAY "closed before you can access the"
@14,23 SAY "system."
SET COLOR TO W/N
@16,23 CLEAR TO 16,39
@16,23 SAY "Press any key..."
WAIT ""
RETURN
* END Bonkers.Prg


*:*********************************************************
*:
*:        Program: YOUR_OUT.PRG
*:
```

(continues)

```
*:         System: SECURITY PASSWORD & LOG SYSTEM
*:         Author: Pat Adams
*:      Copyright (c) 1990, DB Unlimited
*:  Last modified: 02/06/90     13:49
*:
*:       Called by: TOPFILE.PRG
*:
*:      Documented 02/06/90 at 14:37   SNAP!  version 4.02
*:*******************************************************
* YOUR_OUT.PRG
*
*     Inform user s/he has used up all three
*     attempts to enter a correct password and
*     can not access the system.
*
*     Author: Pat Adams, DB Unlimited     (718) 469-4032
*   Date: 2/5/90
*   Copyright 1990 DB Unlimited
*
*     All rights reserved.
*
* * * * * * * *
*
SET COLOR TO R+*/N
@12, 6,21,36 BOX"╔═╗║ ╝═╚║"
SET COLOR TO W+/N
@13, 7 CLEAR TO 20,35
@13,10 SAY "The password is still"
@14,10 SAY "incorrect and that was"
@15,10 SAY "your third and last try."
@19,10 SAY "Contact your"
@20,10 SAY "LAN administrator."
SET COLOR TO R+*/N
@17,15 CLEAR TO 17,26
@17,15 SAY "YOU'RE OUT!"
?? CHR(7) + CHR(7) + CHR(7) + CHR(7)
xwait = INKEY(8)
RETURN
* END Your_out.PRG
```

About the Author:

Pat Adams is an independent consultant, headquartered in Brooklyn, New York. She is the author of two books on dBASE III and dBASE III+, and numerous articles on the family of dbase languages. She is also founding treasurer of the International dBASE Users Group, and founder and chair of the NYPC Consultants SIG.

EOF

Micro-Budget Marketing

By David Irwin
with Emile Barrios

O Lord, won't you buy me
A color TV?
"Dialing for Dollars"
Is trying to find me.

— Janis Joplin

Introduction

So you've created this wonderful piece of software and you're ready to unleash it upon the world. Problem is, your marketing budget is smaller than your chance of wining the lottery. What do you do?

Well, you could wait around until "Dialing for Dollars" calls, or you could get involved in the software industry's newest marketing specialty: "Micro-Budget Marketing," or simply Marketing With No Money. Sound like what you need? Then gather up those champagne wishes, caviar dreams, and Spam bankrolls, and come along with me.

The Fundamentals

Micro-Budget Marketing works a lot like the Biblical tale of David and Goliath. You're facing competition in the marketplace that is better funded and better equipped with the standard marketing weapons (flashy display advertising, big trade show booths, public relations blitzes, etc.). Although you're small, you're also fast and willing to work hard in non-traditional ways. Because you have an idea you believe in. Fanatically.

I'm not suggesting you lay in a supply of stones and slings. You do, however, need to be armed with some basic information before you start your Micro-Budget campaign.

If you've been reading this column for awhile, this will all sound a bit familiar. But let's go through it again anyway.

You don't have any time or money to waste, so your marketing program should be as focused as possible. Before you begin, try to pinpoint who your potential users are, and where they are, both geographically and demographically. Then search out the media that reach these potential users: magazines, newsletters, user groups, etc. Don't squander time and effort targeting people and publications that won't be interested in you or your technology.

(continues)

Once you know who you're looking for and how to reach them, you're ready to begin.

Press Contacts and Articles

Getting your products written about in the right publications can be as valuable (or even more valuable) than expensive display advertising. A potential buyer might scan your ad briefly, but if he's interested in your product category, chances are he'll read an article word-for-word.

This doesn't happen overnight. As we've talked about in previous columns, you need to develop an ongoing relationship with editors at the publications you've targeted. A relationship that's more on a personal (rather than a product) level. Try to position yourself as a resource to the editor — an expert on your particular subject that she can quote in various articles. If you can persuade the editor that you know what you're talking about, then she'll probably pay more attention to the product you've created.

It's not a good idea to bombard a particular editor with phone calls asking for a product review or writeup. Nothing will alienate an editor faster than a pestering programmer who won't go away.

Always remember David Irwin's Rule One of Micro-Budget Press Relations: THE EDITOR IS YOUR ALLY. TREAT HIM THAT WAY.

In trying to generate press, ask yourself "How will information about my product benefit the reader of this particular publication?" When pitching an editor about a product review or article, approach it in those terms. Sell the editor on the benefit to his readers. That approach tells the editor that you at least know who his readers are and what they want. That's a good start.

Get ready to fail, because for every product review or article you place, you'll get at least 10 rejections. That doesn't mean your product is worthless, or that your Micro-Budget Marketing program is a flop. Just keep trying, and no matter how rude an editor may be to you, remember Rule One.

User Groups

If you're interested in making personal appearances, then user groups can help you build grassroots support for your product.

User groups come in all shapes and sizes, and you can find them almost anywhere. Some groups are devoted to specific products (dBASE, Clipper, etc.), while others have a more general charter. These larger groups often have subgroups or Special Interest Groups (SIGs) devoted to individual products or types of work.

One thing almost all user groups have in common is they are looking for people like you to come and speak at their meetings. They also like to have products they can give away as door prizes.

The best place to start is right in your own back yard. Check around and see what user groups are in your area. Ask if they would be interested in seeing a demonstration of your product. Offer a free copy or two as an incentive.

And if you do these demonstrations, *be prepared*. A poorly planned product demo is sure to convince the audience that your product isn't worth investigating. Rehearse and refine your demonstration beforehand, and you'll give your product a fighting chance.

Going out of town to speak to user groups generally means spending your own money to get there. That type of expenditure makes sense only if the potential benefit from making the appearance outweighs the number of greenbacks you're parting with to get there.

Strategic Alliances

No, I don't mean making a pact with the devil. Effective Micro-Budget Marketing is about trying every way possible to get your product seen and used (and hopefully bought). One possibility might be to bundle your software with another (larger and more established) company's products.

In seeking this type of alliance, keep in mind that your product should not compete directly with the other product. And remember that established companies with established products get solicited for this type of bundling all the time. Choose carefully, make your case as persuasively as possible, and don't be surprised if the answer is "No." Just keep trying.

Another type of Micro-Budget strategic alliance involves participation in trade shows. Of course, you don't have the kind of money to allow you to have a booth at a major show. But there are a lot of people who are in that same boat. And if enough of you pool your resources, perhaps you can make a go of it. Or as Confucius say: "Together, Many Micro Budgets Scale Mighty Marketing Mountain."

An alternative to renting a booth might be for your group to get a suite at a nearby hotel, and set up shop there. Attend the show, make some introductions, and invite editors and other important folks up to the suite to see some demonstrations. Perhaps you could offer them some food and refreshments as well. It's often possible to contact key editors before the show to set up appointments. Trade shows are where you'll find a lot of editors

(continues)

in one place, so spending a few bucks to be there with them is often a good idea.

I've seen this type of thing work very successfully. The key is putting the right group together. It takes some work and organization, but hey, that's what Micro-Budget Marketing is all about.

Product Trade

Finally, let's talk about some good old-fashioned bartering. Suppose someone offered you several thousand dollars worth of display advertising in exchange for a certain amount of your product. Should you do it?

In virtually every instance, I would say *yes*! Product trade is a good way to do some advertising, because it means that people are going to sell your product to pay for your ad. And let's face it, having other people sell your product for you is easier than selling it yourself. And every little bit of advertising helps when you're trying to build awareness of your product in the market.

Product trade should be thought out carefully before the deal is done. If your ad will appear in a catalog, for example, look at the catalog's circulation, and find out who the readers are. Are they the type of users who are likely to need your product? Examine the track record of the publications involved. Talk with some people who already advertise in the publication. Find out all you can. Once you've got the facts, it's easier to make a decision. And be careful of trade-outs if you're already buying ad space in other publications. If your existing ads are generating direct sales, then you might not want to trade out for additional advertising. Because most catalogs that operate on trade-outs offer software at a discount, you may be robbing yourself of direct sales.

Next Month: Goldilocks and The Three Wares: FreeWare, ShareWare, and VaporWare.

About the Author:

David Irwin is President of Irwin Ink, a microcomputer marketing and consulting company. David was the Technical Editor of ***Data Based Advisor*** *and President and Chief Executive Officer of Data Based Solutions, the parent company of* ***Data Based Advisor*** *and a dBASE-language software publisher. David has recently written for* ***DBMS Views****, (a marketing publication covering the dbase community) and* ***IDBUG Journal****.*

EOF

Multiple Data Entry Screen

By Joshua Zavelovich

Introduction

Almost every database programmer encounters problems with the entry of multiple data fields. dbase dialects provide seemingly universal and convenient instruments for this problem, specifically, multiple SAY...GET statements activated by a single READ. However, most people who use these instruments are quite annoyed by the idiosyncrasies they possess. For example, pressing the UP key moves you to the upper field when you are in the middle of the data table, but if you press it accidentally while editing the uppermost field, you will jump to the first command line after READ. There is no simple way to implement wraparound action in your data entry, even though it is standard in today's user interfaces. You can improve the situation somewhat by locking the SAY...GET...READ sequence within a DO WHILE...ENDDO loop. Here is an example of such an approach:

```
************************************************************
*       Program: DATASCR1.PRG
*
*       Author: Joshua Zavelovich
*  Copyright (c) 1989, Joshua Zavelovich & Medasyst, Inc.
*
************************************************************
SET STATUS OFF
SET ESCAPE OFF
SET BELL OFF
SET TALK OFF
SET SCOREBOARD OFF
SET PROCEDURE TO datascr1

SET FUNCTION 2 TO ''
SET FUNCTION 3 TO ''
SET FUNCTION 4 TO ''
SET FUNCTION 5 TO ''
SET FUNCTION 6 TO ''
SET FUNCTION 7 TO ''
SET FUNCTION 8 TO ''
SET FUNCTION 9 TO ''
SET FUNCTION 10 TO ''

RELEASE workarray
RELEASE xposition
RELEASE yposition
DIMENSION workarray(6)
DIMENSION xposition(6)
DIMENSION yposition(6)

* Arrange exit on pressing of F2 key
ON KEY = 316 DO endentry
IF ISCOLOR()
  defcolor = 'W+/B,N/W'
  enhcolor = 'GR+/R,B/W'
ELSE
  defcolor = 'W/N,N/W'
  enhcolor = 'W+/N,N/W'
ENDIF

SET COLOR TO &defcolor
```

(continues)

```
CLEAR

I = 1
DO WHILE I <= 6
  workarray(I) = '    '
  IF I < 4
    xposition(I) = 20
    yposition(I) = 12+2*(I-1)
  ELSE
    xposition(I) = 40
    yposition(I) = 12 + 2 * (I - 4)
  ENDIF
  I = I + 1
ENDDO

over = .F.
@ 10,7 TO 17,48 DOUBLE
SET COLOR TO &enhcolor
@ 10,16 SAY  'Enter data and press F2'
SET COLOR TO &defcolor
I = 1
DO WHILE I <= 6
  @ yposition(I),xposition(I) - 8 SAY STR(I,1,0) + ' field' I = I + 1
ENDDO
DO WHILE .NOT. over
  I = 1
  DO WHILE I <= 6
    @ yposition(I),xposition(I) GET workarray(I)
    I = I + 1
  ENDDO
  READ
  LASTKEY = READKEY()
  IF .NOT. over
    DO CASE
    CASE LASTKEY >= 6 .AND. LASTKEY <= 7 .OR. ;
      LASTKEY >= 262 .AND. LASTKEY <= 263
      DO jumpkey
      CASE LASTKEY = 12 .OR. LASTKEY = 268
        DO jumpkey
      OTHERWISE
    ENDCASE
  ENDIF
ENDDO
@ 20,0 SAY 'Entry completed'
WAIT
SET PROCEDURE TO
CANCEL
************************************************************
PROCEDURE endentry
*Turn on flag to end editing
over = .T.
* Staff keyboard buffer with PGDN key to jump to the line
* after READ
KEYBOARD CHR(18)
RETURN
************************************************************
PROCEDURE jumpkey
SAVE SCREEN TO oldscreen
?? SYS(2002)
?? CHR(7)
@ 13,15 TO 20,30
@ 14,16 SAY 'You may assign'
@ 15,16 SAY '   this key   '
@ 16,16 SAY '   another    '
@ 17,16 SAY '   function   '
@ 18,16 SAY 'Press any key '
@ 19,16 SAY ' to continue  '
presskey = INKEY(0)
?? SYS(2002,1)
RESTORE SCREEN FROM oldscreen
RETURN
************************************************************
```

If you run this program, you will see that you can freely move the cursor through the data fields and have semi-wraparound action. You can leave editing from any field. Here, we take advantage of the visible idiosyncrasy of the PGDN key, so we can leave data editing from any field at any time. Note, however, that the field from which you leave will not hold an unfinished entry, so move to the next one and press F2.

Still, there are some obvious disadvantages: Any key that takes you past READ will force the cursor to the first position in the top field. Press F1 for an example of this. On the other hand, you can utilize such keys by making them do something useful. In the above example, this is demonstrated by the PGUP, PGDN, and ESC keys. Since you leave editing via the "ON KEY =" action, notice that you need to finish editing the last field, or the last data entered into this field will be erased. This was extensively discussed by Pat Adams in her article on SYS() functions (foxtalk, March 1989). The use of arrays in the above program is not mandatory, but it saves a few lines of code.

This procedure can be made more elaborate to include individual PICTURE clauses and validation. Suppose now that we want to have on-line help. We can achieve this two ways: First, we can use the F1 key, detect it in the line after READ, and transfer control to the help routine. In this approach we are able to pass partial data input to the field from which on-line help was invoked to the help routine. However, we cannot create field-sensitive help this way since we lose all the information on the field we were editing when we pressed F1. In addition, the user will be forced back to the first field when he returns to the data editing screen.

Another approach is through the use of the "ON KEY =" method as was done by Jerry Whittaker in his excellent article on SCRLBOX procedures (foxtalk, March 1989). In that approach, the SYS(18) function is used to inform the help routine where the user needs assistance, and then the program returns the user to the same field from which he asked for help. Overall, the latter approach is much better; still, the penalty is that partial data input into the field is not preserved. If partial input was preserved, the help routine could have used it in matching with the contents of the indexed look-up tables, like the one discussed in the mentioned article by Jerry Whittaker.

Here, I would like to suggest an approach that will solve several problems at once. It will allow complete wraparound action, field-sensitive help with preservation of partial data input, and true XY movement on the data screen. It also includes some other helpful features. One of these features is the individual field-specific validation of data entry. The initial filtering of the user's input is accomplished via an array of PICTURE functions and/or templates and the use of macros. Beyond that, the problem of validation is more complicated.

(continues)

We can construct a UDF with "i" as an argument to use as a clause in the VALID statements. This will provide the user with "i"-sensitive and therefore field-specific validation. However, it will interfere with field-sensitive help. In other words, the user will be locked on that field if he cannot enter information that will pass the VALID filter. This will leave the user with only two choices: either abort data entry or seek help in some other way. I suggest an alternative approach that will circumvent this obstacle; that is, to validate entered data in a separate procedure or UDF.

This approach simultaneously solves yet another problem that has been discussed at length in foxtalk. I refer to the inability of FoxBASE+ to support nested GET...READs. Here, the procedure may include an input from the user and search on indexed files. I refer the reader again to Jerry Whittaker's March '89 article on SCRLBOX, where the author suggested one way to circumvent the problem of nested GETs by substitution of GET...READ with an INKEY() routine. In our case, we would be able to accept the user's input through the simple GET..READ sequence. Finally, I want to stress that the task accomplished here would have been impossible without the use of arrays, as opposed to the first example. Thanks once again to the folks at Fox Software!

```
************************************************************
*
*       Program: DATASCR2.PRG
*
*        Author: Joshua  Zavelovich
*  Copyright (c) 1989, Joshua Zavelovich & Medasyst, Inc.
*
************************************************************
SET ESCAPE OFF
SET STATUS OFF
SET BELL OFF
SET TALK OFF
SET SCOREBOARD OFF
SET PROCEDURE TO datascr2

SET FUNCTION 2 TO ''
SET FUNCTION 3 TO ''
SET FUNCTION 4 TO ''
SET FUNCTION 5 TO ''
SET FUNCTION 6 TO ''
SET FUNCTION 7 TO ''
SET FUNCTION 8 TO ''
SET FUNCTION 9 TO ''
SET FUNCTION 10 TO ''

RELEASE workarray
RELEASE datanames
RELEASE picarray
RELEASE xposition
RELEASE yposition

DIMENSION workarray(8)
DIMENSION datanames(8)
DIMENSION picarray(8)
DIMENSION xposition(8)
DIMENSION yposition(8)

CLEAR
IF ISCOLOR()
  defcolor = 'W+/B,N/W'
  intcolor = 'GR+/R'
  enhcolor = 'W+/R'
ELSE
  defcolor = 'W/N'
  intcolor = 'W+/N'
  enhcolor = 'N/W' ENDIF

STORE 'Last name      ' TO datanames(1)
STORE 'First name     ' TO datanames(2)
STORE 'Date of birth  ' TO datanames(3)
STORE 'Soc security #' TO datanames(4)
STORE 'Street     ' TO datanames(5)
STORE 'City       ' TO datanames(6)
STORE 'State      ' TO datanames(7)
STORE 'Zip code   ' TO datanames(8)

* Initialize data array
STORE SPACE(10)      TO workarray(1)
STORE SPACE(8)       TO workarray(2)
STORE '  /  /  '     TO workarray(3)
STORE '   -  -    ' TO workarray(4)
STORE SPACE(10)      TO workarray(5)
STORE SPACE(10)      TO workarray(6)
STORE SPACE(2)       TO workarray(7)
STORE SPACE(5)       TO workarray(8)

* Create array of PICTURE clauses
STORE '@A!'             TO picarray(1)
STORE '@A!'             TO picarray(2)
STORE '99/99/99'        TO picarray(3)
STORE '999-99-9999'     TO picarray(4)
STORE ''                TO picarray(5)
STORE '@X'              TO picarray(6)
STORE '@! AA'           TO picarray(7)
STORE '99999'           TO picarray(8)

I = 1
DO WHILE I <= 8
  xposition(I) = IIF( I <= 4 ,2 ,30)
  yposition(I) = IIF( I <= 4, 3 + 2 * (I - 1) ,3 + 2 * (I - 5) )
  I = I + 1
ENDDO

@ 1,0 TO 11,52 DOUBLE
SET COLOR TO &enhcolor
@ 1,14 SAY  'Enter data and press F2'
@ 11,10 SAY  'Esc - abort data entry, F1 - help'
SET COLOR TO &defcolor

I = 1
DO WHILE I <= 8
  @ yposition(I),xposition(I) SAY datanames(I)
  @ yposition(I),xposition(I) + IIF(I <= 4,15,10) ;
    GET workarray(I)
  I = I + 1
ENDDO

CLEAR GETS
* Make F2 an exit key
ON KEY = 316 DO dataexit
* Set up exit flag
over = .F.
I = 1
DO WHILE .T.
  ipresent = I
  SET COLOR TO &intcolor
  @ yposition(I),xposition(I) SAY datanames(I)
  SET COLOR TO &defcolor
  @ yposition(I),xposition(I) + IIF(I <= 4,15,10) ;
    GET workarray(I) PICTURE picarray(I)
  READ
  IF READKEY() = 12 .OR. READKEY() = 268        && Esc
    EXIT
  ENDIF
```

(continues)

```
  IF READKEY() = 36 .OR. READKEY() = 292        && F1
    DO datahelp
    LOOP
  ENDIF
  IF .NOT. truetest(I)                          && Validation
    LOOP
  ENDIF
  IF over
    EXIT
  ENDIF
  exitkey = READKEY()
  DO CASE
    CASE exitkey = 14 .OR. exitkey = 270
      EXIT
    CASE exitkey = 0 .OR. exitkey = 256    && Left or Backspace
      IF I > 4
        I = I - 4
      ENDIF
    CASE exitkey = 1 .OR. exitkey = 257    && Right or Backspace
      IF I <= 4
        I = I + 4
      ENDIF
    CASE exitkey = 6 .OR. exitkey = 262           && PgUp
      I = 1
    CASE exitkey = 7 .OR. exitkey = 263           && PgDn
      I = 8
    CASE exitkey = 4 .OR. exitkey = 260           && Up
      IF I > 1
        I = I - 1
      ELSE
        I = 8
      ENDIF
    CASE exitkey = 5 .OR. exitkey = 261           && Down
      IF I < 8
        I = I + 1
      ELSE
        I = 1
      ENDIF
    OTHERWISE
      IF I < 8                                 && Any other key
        I = I + 1
      ELSE
        I = 1
      ENDIF
  ENDCASE
  @ yposition(ipresent),xposition(ipresent);
    SAY datanames(ipresent)
ENDDO
IF over
  @ 12,8  SAY 'You have entered the following data'
  I = 1
  DO WHILE I <= 8
    @ yposition(I) + 10,xposition(I) SAY datanames(I)+':'
    @ yposition(I) + 10,xposition(I) + IIF(I <= 4,15,10);
      SAY workarray(I)
    I = I + 1
  ENDDO
ELSE
  ?? CHR(7)
  @ 20,0 SAY 'You have aborted data entry routine'
ENDIF

WAIT
SET PROCEDURE TO
CANCEL

************************************************************
PROCEDURE  datahelp
* On-line help
PRIVATE oldscreen
SAVE SCREEN TO oldscreen
?? SYS(2002)
?? CHR(7)
@ 10,15 CLEAR TO 21,33
@ 10,15 TO 21,33
@ 11,16 SAY 'You have invoked'
@ 12,16 SAY 'help routine    '
@ 13,16 SAY 'by pressing F1  '
@ 14,16 SAY '  function key  '
@ 15,16 SAY 'You asked for   '
@ 16,16 SAY 'help entering   '
SET COLOR TO &intcolor
@ 17,16 SAY datanames(I)
SET COLOR TO &defcolor
@ 18,16 SAY 'Your input was  '
SET COLOR TO &intcolor
@ 19,16 SAY workarray(I)
SET COLOR TO &defcolor
@ 20,16 SAY 'Press any key...'
presskey = INKEY(0)
?? SYS(2002,1)
RESTORE SCREEN FROM oldscreen RETURN
************************************************************
PROCEDURE dataexit
* Turn on exit flag
over = .T.
* Force exit from the GET...READ
KEYBOARD CHR(13)
RETURN
************************************************************
PROCEDURE truetest

PARAMETERS fieldnum
DO CASE
CASE fieldnum = 3
  IF DTOC(CTOD(workarray(I))) = '  /  /  '
    ??CHR(7)
    ?? SYS(2002)
    SAVE SCREEN TO oldscreen
    @ 12,8 CLEAR TO 15,25
    @ 12,8 TO 15,25 DOUBLE
    @ 13,9 SAY 'Input error'
    @ 14,9 SAY 'Press any key...'
    presskey = INKEY(0)
    STORE .F. TO over          && prevent accidental exit
    STORE .F. TO ok
    RESTORE SCREEN FROM oldscreen
    ?? SYS(2002,1)
  ELSE
    STORE .T. TO ok
  ENDIF
OTHERWISE
  STORE .T. TO ok
ENDCASE
RETURN ok
************************************************************
```

With little additional effort, this routine can be transformed into a generic data entry procedure.

About the Author:

Joshua Zavelovich, PhD, has written a number of programs for device-to-PC interfacing and data management. He has also developed comprehensive office management systems for physicians. He can be reached at Medasyst,Inc. P.O. Box 59610, Chicago, IL 60659.

EOF

Faster Search Program III:

Eliminate the Macros! And the Arrays too!

By Edward C. Marzo

Herman Rohr's October "Faster Search Program" generated some excellent ideas for improvement (as published in the December 1989 foxtalk). Additional ideas will simplify the process still further... and make the coding easier to follow. They will eliminate the need for macros and will correct several logical errors in Rohr's program.

Macros increase execution time, and it is more difficult to follow the logic if macros are used. On both scores, eliminating them is advantageous. My tests reveal that in this program, commands without macros will be three to six times faster than with macros!

The idea of using arrays to simplify processing has merit. But arrays are not needed. Perhaps their principal value would be to make an easily modifiable skeleton program.

For the "DO WHILE" loop, all that is needed is the key value in a memory variable and the starting position — in a string composed of all the fields concatenated — of the key field. For example:

```
If TDESC is the key, starting position is 1
If TMFG is the key, starting position is 26
If TMDL is the key, starting position is 46 etc.
```

For the IF statement, again no macro is needed. Simply compare each field either to itself or to the memory variable, depending on whether or not the memory variable has a value. The IIF function makes this easy.

In Rohr's program, a page heading is printed before it is determined that one is needed. And an extra DO WHILE &SOS is used to accomplish this. A simple technique for page headings is to initialize the line counter to a high value (99) and then to insert the page header commands after the command that tests for this counter against the limit. In his case, initialize S to 99. If S exceeds 56, print the header. This should be done *before* the detail line is written; otherwise, a heading may be printed with no detail line to follow!

```
*SEARCH.PRG
*
* E.C. Marzo   Jan. 9, 1990
*
* Revised version of SEARCH.PRG published in Oct. 89
* and modified in Dec. 1989 FoxTalk
*
*  Revised to simplify and to eliminate macros
*
*    Makes it easier to adapt the same approach
*    to similar problems. And it avoids the use of
*    time-consuming macros. This also makes
*    the logic of the program easier to follow.
*
*    Uses a UDF (VAD) to check each field and to record
*    the Key field and the value to which the index order
*    should be set.
*
*  DBF and index files are the same as those in original
*   program
*
* For FoxBASE+ or FoxPRO
*
SET PROCEDURE TO SEARCH
DO MAIN
PROCEDURE MAIN
  *
  SET TALK OFF
  SET CONFIRM ON
  SET BELL OFF
  SET EXACT OFF
  SET STATUS OFF
  *
  USE BIOMSTR INDEX DESCMSTR, BIOMFG, BIOMDL, DEPTMSTR
  *
  DO WHILE .T.
    CLEAR
    STORE SPACE(25) TO TDESC
    STORE SPACE(20) TO TPM, TMFG, TMDL
    STORE SPACE(4)  TO TCTR
    *
    * MKEY will ultimately contain the value in the
    * first non-blank field.
    *
    * KEYSTART will contain the position, in a string made
    * up of the four dbf fields concatenated, where the
    * key starts. Thus, if TDESC is blank and TMFG has a
    * value, KEYSTART will be 26
    *
    DO WHILE .T.
      STORE '' TO MKEY
      STORE  1 TO KEYSTART
      STORE 99 TO MORDER
      *
      @  4,4 SAY 'Enter Search Criteria: '
      @  8,4 SAY 'Description.....' GET TDESC ;
                 VALID VAD(TDESC, 1)
      @ 10,4 SAY 'Manufacturer....' GET TMFG ;
                 VALID VAD(TMFG, 2)
      @ 12,4 SAY 'Model Number....' GET TMDL ;
                 VALID VAD(TMDL, 3)
      @ 14,4 SAY 'Cost Center.....' GET TCTR PICT '9999' ;
                 VALID VAD(TCTR, 4)
      READ
      *
      * In VAD proc, ESC is stuffed into keyboard if user
      *  went back to a field after MKEY was determined
      *
      IF LASTKEY() = 27
        LOOP
      ELSE
        EXIT
      ENDIF
    ENDDO for DO WHILE .T.
    *
    IF LEN(TRIM(TDESC + TMFG + TMDL + TCTR)) = 0
      RETURN
    ENDIF
  *
  SET ORDER TO MORDER
  SEEK MKEY
```

(continues)

```
  *
  SET DEVICE TO PRINT
  *
  STORE 99 TO S
  STORE  1 TO Y
  *
  * Instead of using a macro for the DO WHILE, the value
  * in MKEY is compared to that part of the string
  * DESCR+MFG+MODEL+CTR which is appropriate for the key.
  *   Thus, KEYSTART indicates where in the string
  *         to start the comparison. And the length
  *         of the key field indicates how many characters
  *         to compare
  *
  * Note also that a .NOT. EOF() is included to insure
  * proper processing should EOF be encountered.
  *
  DO WHILE .NOT. EOF() .AND. ;
    MKEY = SUBS(DESCR + MFG + MODEL + CTR, KEYSTART,
LEN(MKEY))
    *
    * Instead of using a macro for the IF, all four fields
    * are compared.
    * Each field is compared either to itself or to the
    * value entered by the user. The decision is made
    * simply by using the IIF function to ask if a value
    * was entered
    *
    IF DESCR = IIF(LEN(TRIM(TDESC)) = 0, DESCR, TDESC) .AND.;
       MFG = IIF(LEN(TRIM(TMFG))  = 0, MFG, TMFG) .AND. ;
      MODEL = IIF(LEN(TRIM(TMDL))  = 0, MODEL, TMDL) .AND. ;
       CTR = IIF(LEN(TRIM(TCTR))  = 0, CTR, TCTR)
      *
      * The test for page end is made before the line
      * is printed. If this is not done, a page may be
      * printed with no detail lines
      *
      IF S >= 57
        *** PAGE HEADING COMMANDS
        @ 1,1 SAY 'PAGE HEAD '
        *   etc
        STORE 2 TO S
        Y = Y + 1
      ENDIF
      @ S,1 SAY 'DETAIL DATA ' + DESCR + ' ' + MFG + ' ' ;
      + MODEL + ' ' + CTR
      *   etc
      S = S + 1
    ENDIF
    SKIP
  ENDDO
ENDDO
*
*
************************************************************
*       PROC VAD
************************************************************
PROCEDURE VAD
  *
  * First parameter is the field
  * Second is the corresponding index order
  *
  PARAMETER PPFIELD,PPORDER
  *
  * Clear any error message which may exist
  @ 16,4
  *
  * Any value entered?
  *
  IF LEN(TRIM(PPFIELD)) = 0
    *  No.
    * If the key(MKEY) is not yet determined, the starting
    * position of the key should be increased by the length
    * of this field
    *
    IF LEN(TRIM(MKEY)) = 0
      STORE KEYSTART + LEN(PPFIELD) TO KEYSTART
    ENDIF
    RETURN .T.
  ENDIF
*
SET ORDER TO PPORDER
SEEK PPFIELD
IF .NOT. FOUND()
  ?? CHR(7)
  @ 16,4 SAY 'No match on ' + PPFIELD
  RETURN .F.
ENDIF
*
* This is a test to see if user has gone back to a
* previous field after the key has already been
* determined. If so, the key must be reestablished.
* This is done by forcing an ESC
*
IF LEN(TRIM(MKEY)) > 0 .AND. PPORDER < MORDER
  KEYBOARD CHR(27) + CHR(13)
  RETURN .T.
ENDIF
*
* If the key field has not yet been determined,
* set it up and record the order
IF LEN(TRIM(MKEY)) = 0
  STORE PPFIELD TO MKEY
  STORE PPORDER TO MORDER
ENDIF
RETURN .T.
```

About the Author:

Edward C. Marzo has been involved in the design, installation and management of computer systems for many years. He specializes in the apparel industry and has systems written in FoxBASE+ and FoxPro installed in many states and the Dominican Republic. He can be reached at Box 331, Agawam, MA 01001.

EOF

TELL US WHAT YOU WANT TO KNOW

Pinnacle Publishing welcomes your comments on foxtalk. We want to make foxtalk truly responsive to your needs. So, the more we hear from you, the more we can do for you. Give us a call at **1-800-231-1293**, or write to Glenn A. Hart, c/o Pinnacle Publishing, Inc. PO Box 8099 Federal Way, WA 98003.

PICKLIST

By Len Levy

Introduction

FoxPro, as you no doubt have heard by now, contains an overwhelming presentation of built-in window, menu, and browsing functions. They enable even the novice programmer to create dazzling routines in a fraction of the code that it would take FoxBASE to duplicate.

In FoxPro, memo fields are displayed dynamically, automatically adjusted to fit within the borders of the currently active window. This procedure, to a lesser extent, duplicates this function in FoxBASE 2.1, justifying the text. However, the window size is fixed full screen.

The online point-and-shoot picklist has become a standard feature in data entry procedures. There have been several articles offered here in foxtalk as well as in other newsletters and magazines providing look-up and/or validation routines for FoxBASE+ 2.1. However, these procedures, many of which are quite elegant, have certain limitations that do not fit my needs in one particular scenario, i.e., a point-and-shoot data entry routine that will read a database file with *any number* of records and display the code field plus a description field of *up to* 254 *characters*, displayed on one to four lines, formatted without awkward word breaks, and to the right of the highlighted code field.

"PICKLIST" is a UDF that requests two parameters: the FILE AREA and the MESSAGE, which appears at the top of the picklist window. A database file must be opened before the call. The structure of the file is:

```
1  CODE           Character    1 - 9 characters
2  DESCRIPTION    Character    1 - 254 characters
```

If your data tables' field names differ from "CODE" and "DESC," you can change the references to these fields in the "PICKLIST" code.

Assume that your main data file contains numerous codes related to items defined in many other data files. A simple READ MENU TO routine is fine for a short descriptor. However, if you would prefer the entire description field to appear together with the associated code, and both exceed 48 characters, you are out of luck! And if there were, say, 1,000 codes to display, you will see only the first 128!

"PICKLIST" stores the CODE and DESCRIPTION fields of every record in the "f" array. If the description field is greater than 66 characters, the "d" array is created. The "d" array contains up to four formatted description lines. After the array has been loaded, two memory files are created, D.MEM and F.MEM. If additional lookups are requested by the calling program, these memory files are recalled rather than recreating the arrays. The result is a substantial saving in processing time.

To try out "PICKLIST" from the dot prompt, first create your data file with the fields as described above. Activate the file with "USE <datafile>." Then issue the call:

```
<memvar>=picklist("A","This is the Title of my Data Window")
```

<memvar> contains the code selected. Listing memory shows that <memvar> is a public variable that could be stored in an array set up by the calling program to hold any number of selected records that could be printed, edited, sorted, displayed, etc., in succession without having to make individual queries. The variables "m_cnt" and "m_lines" have been created as flags that notify the calling program that it is not necessary to recreate the arrays.

Your disk drive now contains "F.MEM," which is comprised of the code and up to 66 characters of the description. If your data file's DESC field contained more than 66 characters, you would also find "D.MEM" on your disk, which contains four elements for each record.... one element for each formatted descriptor line in each record.

Make sure you release "m_cnt" and "m_lines" if you want to select another data file. If you don't, the old arrays will be read.

A glaring weakness in this UDF is the limitation of scrolling keys. When using MENU TO and @...PROMPT..., you are limited to Ctrl-PgUp and Ctrl-PgDn for view previous and next screens using READKEY(). However, I'm sure that some creative soul out there will find an alternate means of scrolling and jumping to the beginning or end of the file. Let me know!

"PICKLIST" makes a call to "TITLE2," which simply displays a full screen window with a double line border and your message centered on the top line between "" and "", CHR(16) and CHR(17). Be sure to include this procedure in your main procedure file or have it on the disk in your current directory.

```
************************************************************
*                          PICKLIST                        *
************************************************************
*** PICKLIST.PRG
*** (c) 1990, Len Levy, Data Management Systems
***
***  This UDF requires a your lookup DBF table to have at
***  least TWO fields....
***  CODE  (Character - up to 9 characters in length)
***  DESC  (Description field - up to 254 characters)
***
***  FoxTalk subscribers who receive the accompanying
```

(continues)

```
***  program diskette can use the included help file...
***  'PICKEM.DBF' to illustrate the routine.
***
***  Before calling the routine, be sure to include
***  'TITLE2.PRG' either on your disk or in the same
***  main procedure file.
***
***  Your calling procedure, as an example, will open
***  files prior to calling 'PICKLIST':
***
***  SELE 1
***  USE mainfile
***  SELE 2
***  USE lookup INDEX lookup
***  SELE 1
***  SET RELATION TO <field> INTO B
***
***  Make the call as follows:
***  <memvar> = PICKLIST("B","LOOKUP DATA TABLE")
***                      |   |
***                     Data| Area for lookup table
***                         |
***                        Title To appear in Window
***
***  Asterisks removed from next line if called procedure
***  PROCEDURE PICKLIST
PARAMETERS file_area,msg
PRIVATE choice,old_area,old_col,zz,recs,cntr,f,d,l,s1,cntr_2
PRIVATE cntr_3,before,r,c,str1,str2,fields,old_scr,saf
SAVE SCREEN TO old_scr              && Save old screen
STORE sys(2001,"COLOR") TO old_col  && Save old colors
STORE SYS(2001,"SAFE") TO saf       && Save SAFETY status
old_area=CHR(64+select())           && Save old work area
SELE &file_area
STORE fcount() TO fields            && # of fields
STORE reccount() TO recs            && # of records
***
***  If this picklist is called two or more times in
***  sucession, it is senseless to recreate the arrays...
***  rather save them to temporary 'mem' files
***
IF TYPE([m_cnt])="U"                && Is this the 1st pass?
***
*** Initialize array-code & description
***
   PUBLIC m_cnt                     && If so, create flag
   DIMENSION f(recs,2)
   STORE "" TO f
   STORE 1 TO cntr
   GO TOP
   DO WHILE cntr<=recs              && and create 'f' array
      STORE code TO f(cntr,1)
      STORE TRIM(desc) TO f(cntr,2)
      SKIP
      IF EOF()
         EXIT
      ENDIF
      STORE cntr+1 TO cntr
   ENDDO
   SET SAFE OFF
   SAVE ALL LIKE f TO f             && Save data array
                                    && to 'F.MEM' for another
   ***                              && pass
   *** if the DESC field is greater than 66 characters,
   *** allocate the description to the 'D' array to contain
   *** up to 4 description lines to be displayed without
   *** inappropriate word-breaks.
   ***
   PUBLIC m_lines
   IF LEN(desc) > 66
      PUBLIC m_lines                && More than 1 description
      STORE .T. TO m_lines          && line will be needed
   ENDIF
   IF m_lines
      RELEASE d
      DIMENSION d(recs,4)           && Initialize array for
                                    && description Lines
      STORE "" TO d
      STORE 1 TO cntr
      DO WHILE cntr<=recs
         STORE trim(f(cntr,2)) TO str1,str2
         STORE 1 TO cntr_2
         l=LEN(STR2)
         IF l > 66                  && test DESC for length
            str_len = 0
            DO WHILE .T.
               IF LEN(str2)<=66
                  d(cntr,cntr_2)=str2
                  EXIT
               ENDIF
               STORE left(str2,66) TO s1
***
*** cntr_3 counter for finding last space from the right
*** disect string from right to left
***
               STORE 66 TO cntr_3
               DO WHILE cntr_3 >=1
                  IF SUBSTR(s1,cntr_3,1)=" "    && a space!!
                     EXIT
                  ELSE
                     STORE cntr_3 - 1 TO cntr_3
                  ENDIF
               ENDDO
               d(cntr,cntr_2)=left(s1,cntr_3-1)
               l2 = LEN(d(cntr,cntr_2))+1
               str_len=str_len+12
***
*** grab leftover characters an build new test string
***
               STORE RIGHT(str1,LEN(STR1)-str_len) TO str2
               STORE cntr_2+1 TO cntr_2
            ENDDO
         ELSE
            d(cntr,cntr_2)=trim(f(cntr,2))
            store f(cntr,2) to d(recs,1)
         ENDIF
         STORE cntr+1 TO cntr
      ENDDO
      SAVE ALL LIKE d TO d          && Save DESCRIPTION
                                    && array in 'D.MEM'
   ENDIF
***
*** Remember to erase these temporary 'MEM' files from
*** the calling program
   ***
ELSE                                && 2nd lookup
   RESTORE FROM f ADDITIVE          && Load arrays
   IF m_lines
      RESTORE FROM d ADDITIVE
   ENDIF
ENDIF
***
*** Prepare to display
***
STORE 0 TO old_cntr
STORE 1 TO cntr,old_top
STORE 3 TO col,row
STORE 12 TO col_2
SET COLOR TO b/w
***
*** Call 'TITLE2.PRG' which MUST be in procedure file or
*** current directory
***
DO TITLE2 WITH msg
SET COLOR TO r/w
@ 1,3 SAY "CODE"
@ 1,12 SAY "DESCRIPTION"
SET COLOR TO b/w
@ 2,3 say "═══"                    && ALT + 205
@ 2,12 say "═══════════"
SET COLOR TO n/w,w+/r
```

(continues)

```
?? SYS(2002)                          && Cursor off
STORE .t. TO more
DO WHILE MORE
   STORE 0 TO rec_cntr
   DO WHILE cntr<=recs
      IF rec_cntr=0
         STORE cntr TO top
         STORE cntr TO old_top
      ENDIF
      IF top+rec_cntr#recs
         STORE rec_cntr+1 TO rec_cntr
      ENDIF
      @ row,col PROMPT f(cntr,1)        && Code
      IF .not. m_lines
         @ row,col_2 SAY f(cntr,2)
      ELSE
         @ row,col_2 SAY d(cntr,1)      && Description line 1
         IF LEN(trim(d(cntr,2)))>0
            STORE row+1 TO row
            @ row,col_2 SAY d(cntr,2)   && Description line 2
         ENDIF
         IF LEN(TRIM(d(cntr,3)))>0
            STORE row+1 TO row
            @ row,col_2 SAY d(cntr,3)   && Description line 3
         ENDIF
         IF LEN(trim(d(cntr,4)))>0
            STORE row+1 TO row
            @ row,col_2 SAY d(cntr,4)   && Description line 4
         ENDIF
      ENDIF
      IF cntr=recs                      && Last record
         EXIT
      ENDIF
      IF cntr<recs
         STORE row+2 to row             && Blank Line
         IF FILE('D.MEM').and. row >= 21 && Long description?
            EXIT
         ELSE
            IF row >= 23                && Display Code to
               EXIT                     && line 22
            ENDIF
         ENDIF
         STORE cntr+1 TO cntr
      ENDIF
   ENDDO
   IF cntr<>recs
      STORE rec_cntr TO old_cntr
   ENDIF
   IF top+rec_cntr=recs
      STORE old_cntr TO rec_cntr
   ENDIF
   IF old_top>1
      STORE old_top-rec_cntr TO old_top
   ENDIF
   IF cntr<recs
      @ 24,1 SAY chr(16)+"[^End]=Next Page [^Home]=Previous"
      ?? " Page   [Esc]=Return   [ENTER] to Choose"+chr(17)
   ELSE
      @ 24,1 TO 24,78 DOUBLE
      @ 24,19 say chr(16)+" Hit [ENTER] to Choose, [Esc] to "
      ?? "Return "+chr(17)
   ENDIF
   MENU TO choice
   choice=top+choice-1
   rk=READKEY()
   ?? SYS(2002,1)
   DO CASE
      CASE rk=12                        && Escape Key
         SELECT &old_area
         RESTORE SCREEN FROM old_scr
         SET SAFE &saf                  && Restore SAFETY status
         SET COLOR TO &old_col          && Restore color
         RETURN "  "
      CASE rk=33                        && ^Home
         @ 3,1 CLEAR TO 23,78
         STORE 3 TO row
         STORE top-rec_cntr TO cntr
         IF cntr<1
            STORE 1 TO cntr
         ENDIF
      CASE rk=14                        && Last page of display
         IF cntr = recs
            @ 3,1 CLEAR TO 23,78
            STORE 3 TO row
            STORE top TO cntr
            STORE old_cntr TO rec_cntr
            LOOP
         ENDIF
         IF cntr<recs
            @ 3,1 CLEAR TO 23,78        && ^End
            STORE 3 TO row
            STORE cntr+1 TO cntr
         ENDIF
      CASE rk=15                        && Selected with [ENTER]
         SELECT &old_area
         RESTORE SCREEN FROM old_scr
         SET SAFE &saf
         SET COLOR TO &old_col
         RETURN f(choice,1)
      OTHERWISE
         LOOP
   ENDCASE
ENDDO more

-----------------------------------------------------

*******************************************************
*                      TITLE2                         *
*******************************************************
***Len Levy,  (c) 1990 Data Management Systems
***Draw Frame and Center Message on the Top Line
***
***PROCEDURE TITLE2
   PARAMETERS message
   PRIVATE old_col,l
   old_col = SYS(2001,"COLOR")
   l = (80-LEN(message))/2
   set color to b/w
   CLEAR
   @ 0,0 TO 24,79 double
   @ 0,l-2 SAY chr(16)+" "+REPLICATE(" ",LEN(message))+" ";
      +chr(17)
   SET COLOR TO r/w
   @ 0,l SAY message
   SET COLOR TO &OLD_COL
RETURN
```

About the Author:

Len Levy is a FoxBASE programmer, consultant, teacher, and musician. He started as a teacher in the Yonkers, NY Public Schools System in 1961, initially teaching music and later teaching computer courses in the middle schools. Since 1986 he has worked continually as Programmer and Network Administrator for Yonkers' Special Education Department. He began computing in 1977 and, since 1983, has been addicted to dbase. His private consulting business, DATA MANAGEMENT SYSTEMS, is located at 709 Ardsley Road, Scarsdale, NY 10583, (914) 693-6933.

EOF

Binary Storage Revisited

By Jerry Ela

Introduction

I enjoyed Jim Hewitt's article [*in the December 1989 issue of foxtalk*] on saving disk space with "binary" storage. It's nice to know that some people still realize that disk space isn't unlimited yet.

However, Jim's method isn't as efficient as it could be. Jim was able to pack 24 "bits" into eight bytes. However, it should be possible to represent 24 bits as three bytes, and in FoxPro it is.

There's been a lot of talk about how FoxPro memo fields can be used to store binary data. Because FoxPro permits any byte in a character string, character variables and fields can also be used to store binary data. Note: FoxBASE+ doesn't allow for character 0 in a string, so this won't work in FoxBASE+.

The following two FoxPro UDFs, PakBin and UnPakBin, convert between "binary" character strings in the form "001001010" and packed binary strings (true binary data stored in character strings). These will work with strings of any length (within the limitations of FoxPro).

PakBin pads the right side of the string with "0s" to a multiple of 8, which will remain after the string is unpacked, so be careful if you're concatenating "binary" strings or doing anything else that expects them to be shorter. So long as you are checking the "bits" based on their offset from the beginning of the string, this will have no effect.

```
* pakbin.prg

   * calls: none
   * globals: none
   * returns: char; a packed binary string
   * does: packs a "binary" number string like "001010"
     * into a char string to save disk storage space --
     *uses 1 byte per 8 "bits"
   * written by: Jerry Ela
   * Copyright 1990 by Jerry Ela
   * version: 1.002
   * last modified Wednesday 1/10/1990 @ 10:06

   parameters BinStr_              && string in form "010100"
   private PakBinStr_, PakChrVal_, i_, j_
   PakBinStr_ = ""                 && the packed binary string
   * pad w/ "0"s to multiple of 8
   BinStr_ = padr(BinStr_, 8 * ceiling(len(BinStr_) / 8), "0")
   for i_ = 1 to len(BinStr_) / 8
      PakChrVal_ = 0          && value of current char to be packed
      for j_ = 1 to 8         && convert 8 "bits" to numeric value
         if substr(BinStr_, j_ + 8 * (i_ - 1), 1) = "1"
            PakChrVal_ = PakChrVal_ + 2 ^ (8 - j_)
         endif
      endfor
      PakBinStr_ = PakBinStr_ + chr(PakChrVal_)
      && convert to char & accumulate
   endfor
return   PakBinStr_

---

* unpakbin.prg

   * calls: none
   * globals: none
   * returns: char; an unpacked binary string like "001010"
   * does: unpacks a packed binary string into a char
   * "binary" string like "001010"
   * written by: Jerry Ela
   * Copyright 1990 by Jerry Ela
   * version: 1.003
   * last modified Wednesday 1/10/1990 @ 12:39

   parameters PakBinStr_        && packed binary string
   private BinStr_, PakChrVal_, i_, j_, BitVal_
   BinStr_ = ""                 && "binary" string like "001010"
   for i_ = 1 to len(PakBinStr_)
      PakChrVal_ = asc(substr(PakBinStr_, i_, 1))
      for j_ = 7 to 0 step -1
         BitVal_ = 2 ^ j_       && the value of the current bit
         if PakChrVal_ >= BitVal_      && test if bit is set
            BinStr_ = BinStr_ + "1"
            PakChrVal_ = PakChrVal_ - BitVal_ && drop tested bit
         else
            BinStr_ = BinStr_ + "0"
         endif
      endfor
   endfor
return   BinStr_
```

About the Author:

Jerry Ela is the head of the Computer Services Section at the Division of Water, New York State Department of Environmental Conservation, where he supports about 50 FoxBASE+ users.

EOF

The foxtalk Workshop

Reader Valentine Spiegel writes the following about FoxPro:

I have followed your praises of FoxPro for some months now and feel it is time for a dissenter to be heard. FoxPro is just great for a sophisticated user but has little in it to recommend it for programmers over FoxBASE+.

I am angry and frustrated with the new FoxPro interface. There is no easy way of just setting talk, echo, step, and status on and then debugging one step at a time on screen without a clutter of windows occupying and obscuring the entire screen. The only essential use I have found for the mouse is to reach the "QUIT" command when the keyboard hangs up. The new commands are great when you need them. Many of them replace commands and functions we have written for ourselves, but I would have been happier if they had just been incorporated in a newer version of FoxBASE+. My most time–consuming programs involve multi-indexing of large files, and these have been improved by about five percent in speed without expanded memory and about 10 percent with it. There is no factor of 3 or 4 in speed as I had been led to believe.

My favorite editor, "ME Text Editor," by Magma Systems can no longer be called as an external editor. It loads but doesn't seem to be able to make use of buffers or pages as it did before. It operates much like "Brief" in that one writes macros for it to import text, open windows to other files, pretty print, and compile programs from within the editor.

My advice to any FoxBASE+ user would be to postpone purchase of FoxPro and stay with FoxBASE+ until an external compiler that compiles to a small executable file not requiring a huge runtime module and overlays, and an uncluttered interface for debugging and editing programs are available.

Unless I am yet to discover valuable new features, this program will probably sit in an unused disk directory for the foreseeable future.

Valentine Spiegel

Needless to say, I disagree rather vehemently with almost everything reader Spiegel says. In my opinion, FoxPro offers a tremendous number of powerful advantages to programmers, among them the new commands and functions, many of which would be difficult or impossible to write in FoxBASE+. If Mr. Spiegel is using an EGA or VGA video board, he evidently didn't discover the display modes which allow the trace and debug windows to be positioned below the normal 25x80 display area. In any case, I find FoxPro's debugging facilities are infinitely better than the rather primitive methods in FoxBASE+ that Mr. Spiegel evidently prefers (so much so that I even use FoxPro to debug applications written in FoxBASE+!).

I don't know Mr. Spiegel's hardware environment, and neither have I yet run comprehensive indexing benchmarks. Still, with the specific commercial applications I create, FoxPro is much faster than a mere five or 10 percent.

Debating editors is almost always fruitless. Few subjects generate such vehemence and self-justification. I've used a variety of external editors with FoxBASE+ (my current favorite is Golden Bow's VQ, but I've used Brief for several years, and wrote the original articles in **PC Magazine** *discussing the dBRIEF macro package). It took me a while to become really facile with FoxPro's editor, but I like it very much now. With a few enhancements already planned for release 1.1, I'll be even happier with it. FoxPro allows use of external editors for program files, but not for memo editing, a situation that aggravtes a few users. I've never heard of the ME Text Editor, so I have no idea why it evidently doesn't coexist happily with FoxPro.*

Fox intends to continue marketing FoxBASE+, so those who agree with Mr. Spiegel can go that route. Here in **foxtalk**, *we'll be continuing to cover both (as well as FoxBASE+/Mac). Personally, though, while I continue to have great affection for FoxBASE+, I'm deeply in love with FoxPro – what I can achieve with it, how easy it is to program and debug, how fast it is, and much more.*

EOF

Source Code Disk Subscription Program Saves Time and Money

Attention **foxtalk** subscribers: **foxtalk** is filled with very valuable and useful source code. If you would like to receive a source code disk each month with your publication, please seriously consider the following offer.

Pinnacle Publishing regularly offers a source code disk subscription program. For just $5.00 per month*, subscribers to this service automatically receive a disk each month containing all of the source code found in that month's issue of **foxtalk**. This program is designed to save subscribers hours of laborious rekeying of published source code.

Call us at **1 800 231-1293** or **206/941-2300**; or write to us at **Pinnacle Publishing, Inc., PO Box 8099, Federal Way, WA 98003.**

* $60.00/year

Writer's Guidelines:

The editorial staff of foxtalk encourages anyone who is interested in writing for the journal to submit articles to us. We realize that our readership will be better served by a journal that presents a wide variety of viewpoints, expertise, and topics rather than just those which come from our own FoxBASE+/FoxPro experience.

If you would like to write for the journal, here are some rules we must ask you to follow:

1. All material, prose, and source code must be original and must be owned by you.

2. You must inform us if other publications are considering your article.

3. Pinnacle Publishing reserves all rights to all articles published in foxtalk. *Be aware that your articles, and, particularly your code, may be used by readers in the development and implementation of individual applications.*

4. Articles may be submitted on a diskette (5 1/4," 360K or 1.2M) or via modem. The preferred format is Microsoft Word. Also acceptable is straight ASCII text.

5. *All text must be clean and unformatted,* free of indentations, tabs, control codes, and manual "eye-balling" alignment of text using the space bar. If you need to emphasize key words, or extensively format a table or diagram, please enclose a separate printout of your article that shows the appropriate emphasis. You should include a proposed title and a brief biographical sketch which may appear at the end of the article.

6. Please make certain that no line of text exceeds 60 characters.

7. The editorial staff will decide what to publish and when based on what we think is of most interest to our readers, and on the quality of material submitted. Our target audience is the serious, professional FoxBASE+/FoxPro user, so please write with that individual in mind.

All material submitted for publication will receive fair and judicious treatment; however, we cannot guarantee publication to anyone. Honoraria are paid after publication.

Editorial Cont.

enhancement to system sales. I've come to realize, though, that most dealers are afraid to sell these items to most customers, since they must tacitly admit that computers do crash and data are constantly in jeopardy. Thankfully, this situation is changing as the hardware comes down in price, and both dealers and users become more aware of the issues.

In any event, Pat discusses the hardware issues, backup policies, network administration policies, and password systems, with her usual first-rate code examples as well.

• Marketing maven David Irwin returns with another meaty installment of marketing advice. This time around, he provides some extremely useful techniques to market your applications with little or no marketing budget. Sounds impossible, but it really can be done.

• All of us must contend with data entry screens, a never-ending process in every application. There are almost as many styles and techniques as there are programmers and programming languages. In some respects, FoxBASE+, FoxBASE+/Mac, and, especially, FoxPro elevate the entry screen to new levels of sophistication, but there are some gotchas too. Another first time foxtalk contributor, Joshua Zavelovich, offers some interesting methods that have wide applicability to a variety of applications.

• Herman Rohr's fast seeking concept also continues to challenge creative programmers. Yet another more efficient implementation of this approach comes from Ed Marzo, a new foxtalk contributor.

• Picklists continue to be a most popular topic. Len Levy, whose earlier foxtalk contributions garnered much favorable comment, shares yet another wrinkle on this timeless subject.

• New contributor Jerry Ela improves on the binary packing routines submitted by Jim Hewitt and published in December's foxtalk. His FoxPro functions can save significant disk space in certain applications.

Glenn A. Hart