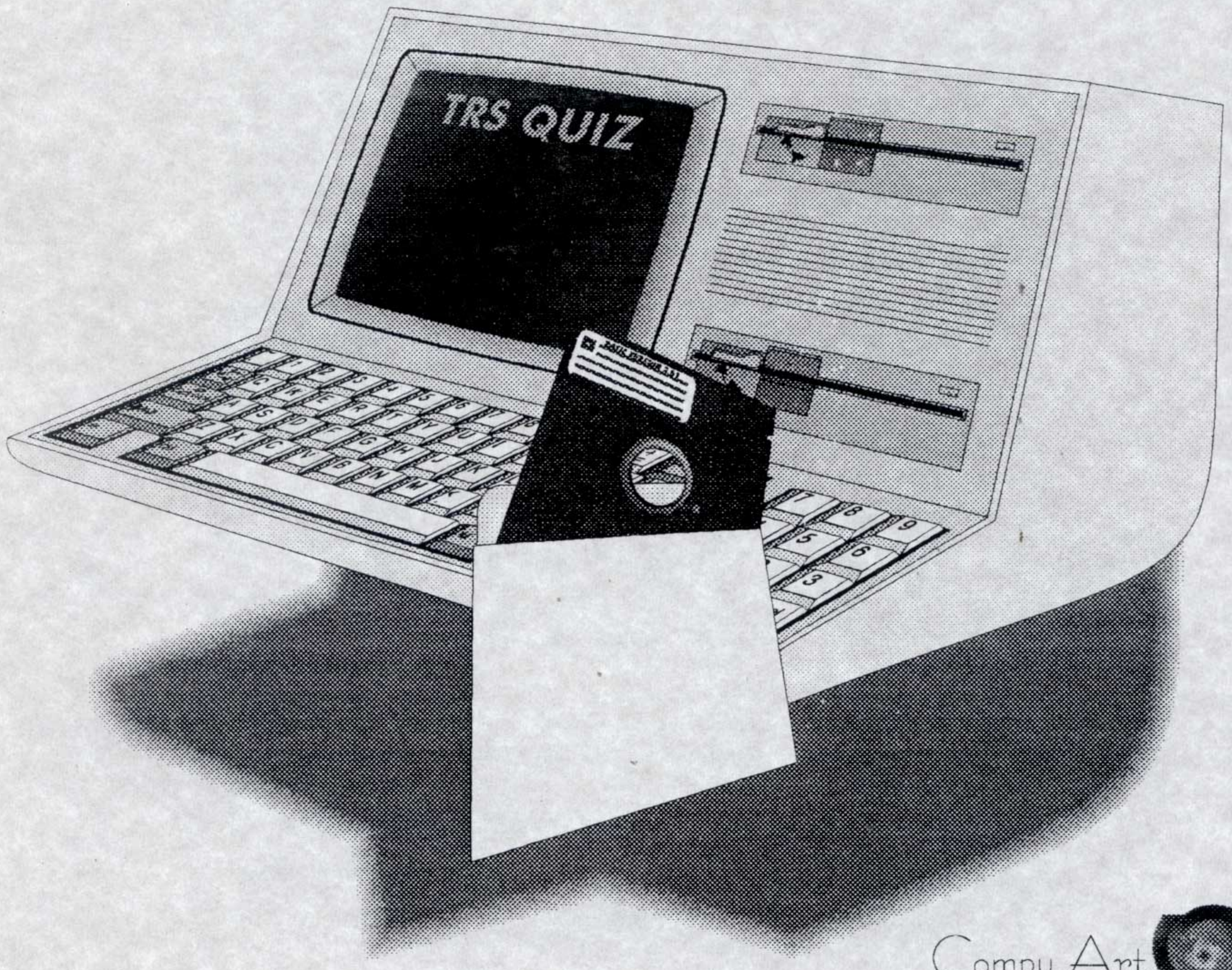
TRSTimes

Volume 8. No. 4 - Jul/Aug 1995 - $4.00



Compu Art

DR. PATCH

UTILITY FOR TRS-80 MODEL 4 AND LS-DOS 6.3.1

A *'MUST HAVE'* FOR ALL LS-DOS 6.3.1 OWNERS.

DR. PATCH MODIFIES LS-DOS 6.3.1 TO DO THINGS THAT WERE NEVER BEFORE POSSIBLE.

COMPLETELY SELF-CONTAINED - MENU-DRIVEN FOR MAXIMUM USER CONVENIENCE.

FAST & SAFE - EACH MODIFICATION IS EASILY REVERSED TO NORMAL DOS OPERATION.

DISABLE PASSWORD CHECK IN FORMAT/CMD
FORMAT DOUBLE-SIDED AS DEFAULT
FORMAT 80 TRACKS AS DEFAULT
DISABLE VERIFY AFTER FORMAT
CHANGE 'DIR' TO 'D'
CHANGE 'CAT' TO 'C'
DIR/CAT WITH (I) PARAMETER AS DEFAULT
DIR/CAT WITH (S,I) PARAMETERS AS DEFAULT
CHANGE 'REMOVE' TO 'DEL'
CHANGE 'RENAME' TO 'REN'
CHANGE 'MEMORY' TO 'MEM'
CHANGE 'DEVICE' TO 'DEV'
DISABLE THE BOOT 'DATE' PROMPT
DISABLE THE BOOT 'TIME' PROMPT
DISABLE FILE PASSWORD PROTECTION
ENABLE EXTENDED ERROR MESSAGES

DISABLE PASSWORD CHECK IN BACKUP/CMD
BACKUP WITH (I) PARAMETER AS DEFAULT
BACKUP WITH VERIFY DISABLED
DISABLE BACKUP 'LIMIT' PROTECTION
DISABLE PASSWORD CHECK IN PURGE
PURGE WITH (I) PARAMETER AS DEFAULT
PURGE WITH (S,I) PARAMETERS AS DEFAULT
PURGE WITH (Q=N) PARAMETER AS DEFAULT
IMPLEMENT THE DOS 'KILL' COMMAND
CHANGE DOS PROMPT TO CUSTOM PROMPT
TURN 'AUTO BREAK DISABLE' OFF
TURN 'SYSGEN' MESSAGE OFF
BOOT WITH NON-BLINKING CURSOR
BOOT WITH CUSTOM CURSOR
BOOT WITH CLOCK ON
BOOT WITH FAST KEY-REPEAT

DR. PATCH IS THE ONLY PROGRAM OF ITS TYPE EVER WRITTEN FOR THE TRS-80 MODEL 4 AND LS-DOS 6.3.1.

DISTRIBUTED EXCLUSIVELY BY TRSTIMES MAGAZINE ON A STANDARD LS-DOS 6.3.1 DATA DISKETTE, ALONG WITH WRITTEN DOCUMENTATION.

DR. PATCH $14.95

NO SHIPPING & HANDLING TO U.S & CANADA. ELSEWHERE PLEASE ADD $4.00
(U.S CURRENCY ONLY, PLEASE)

TRSTimes magazine - dept. DP
5721 Topanga Canyon Blvd. #4
Woodland Hills, CA 91367

DON'T LET YOUR LS-DOS 6.3.1 BE WITHOUT IT!

TRSTimes magazine

Volume 8. No. 4 - Jul/Aug 1995 - $4.00

PUBLISHER-EDITOR
Lance Wolstrup
CONTRIBUTING EDITORS
Roy T. Beck
Dr. Allen Jacobs
TECHNICAL ASSISTANCE
San Gabriel Tandy Users Group
Valley TRS-80 Users Group
Valley Hackers' TRS-80 Users Group

TRSTimes is published bi-monthly by TRSTimes Publications. 5721 Topanga Canyon Blvd., Suite 4. Woodland Hills, CA 91367. U.S.A. (818) 716-7154.

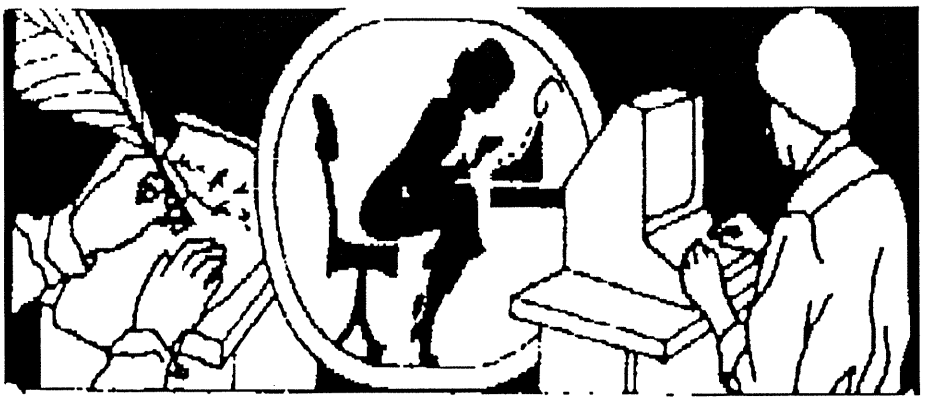
Publication months are January, March, May, July, September and November.

1995 subscription rates (6 issues):
UNITED STATES $21.00
CANADA $22.00 (U.S.)

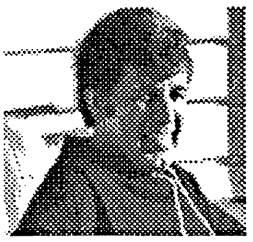
EUROPE, CENTRAL & SOUTH AMERICA:
$26.00 for surface mail or $34.00 for air mail. (U.S. currency only)

ASIA, AUSTRALIA & NEW ZEALAND:
$28.00 for surface mail or $36.00 for air mail. (U.S. currency only)

Article submissions from our readers are welcomed and encouraged. Anything pertaining to the TRS-80 will be evaluated for possible publication. Please send hardcopy and, if at all possible a disk with the material saved in ASCII format. Any disk format is acceptable, but please note on label which format is used.



LITTLE ORPHAN EIGHTY



This issue brings a variety of TRS-80 information for a variety of Tandy computers. In the tradition of the early days of TRSTimes, we feature another article of PEEKing and POKEing. This time, however, the PEEKS and POKES are not for models I, III or 4, but rather for the Model 100. Thanks to Tony Anderson for giving me a fun evening testing out all those little goodies on my wife's machine.

Also, we received several letters, all worried that the C tutorial had ceased. Not to worry, it is back in full force in this issue and we all thank Frank Slinkman for sharing his expertise with us.

The Hints & Tips section brings another Model 100 goody, a couple of Newdos/80 tidbits, a TRSDOS 1.3. patch by the master himself, Andy Levinson, found in the TRSTimes vault, as well as other fun stuff. Thanks goes out to George Mueden, George Saladino, Bob Seaborn, Chris Edwards, Andy Levinson, Jim Kajpust, Eric Robert Greene, Sam Wayne, and Frank Durda IV.

Daniel Myers is back with the solutions to a couple of the Scott Adams adventures, Voodoo Castle and Golden Voyage. I haven't tried these two yet, but little by little, I am working my way through all of them. Thanks Dan.

Recollecting what the good old days were *really* like, John Phillipp writes about his first experiences with hard drives. Besides writing articles for Softside, 80 Micro, and the other TRS-80 publications, Dr. John is the former, long-standing editor of the Interface, the newsletter of the San Gabriel Valley Tandy Users Group (SAGATUG), and we thank him for his kind permission to reprint this article.

Thanks to Kelly Bates for sharing his hardware and software experiences with us and, of course, to my friend, Roy Beck, for another informative article.

Gary Shanafelt brings us up to speed on the Internet. It is good to see that the TRS-80 community can, indeed, join in and access the 'Information Super Highway'. I am impressed with the '*fixing*' of Omniterm. Thanks Gary.

Speaking of the Internet — it appears to be the hottest thing at the moment. However, I do think that the term 'Information Super Highway' is somewhat exaggerated. As one Net old-timer explained "Rather than a super highway, it is more like unending, twisted, narrow back-alleys. The information is there, but it is not always easy to get there. YOU must know where to look." I wholeheartedly agree.

I have spent the last few months trying to educate myself on the 'net' and I can tell you that the learning curve is steep — but it is worth it.

The whole thing started when my wife, Sylvia, attended an Internet seminar for writers. She came home with an armful of brochures and documentation — and a months free access to the Internet. She told me how much fun it had been, and that night we dialed the access number and 'surfed the net' for the very first time. Being free, the service provider had given us a simple shell account; that is, we could access the Internet via text mode only. And we soon realized, that in order to do anything meaningful, you had to know a little UNIX. My wife smiled sweetly at me and said "You learn it, Dear", and she got up and left the computer. Well, not being one to ignore a challenge, I fought with it for a couple of days — then I broke down and bought a UNIX book. I learned just enough to get my feet wet, sending and receiving e-mail, access newsgroups, etc. I was spending most of my free evening time on the net, exploring the nooks and crannies. That stopped immediately when I got my phonebill. True, the Internet access was free, but the company that provided the free acess was in West Los Angeles, and I live in the San Fernando Valley — a toll call. My bill was $170. A*in't nothing free!*

I cancelled the service and found a company that offered a local dial-up number. I now have a 'slip/ppp' account; that is, no more UNIX text — instead, a complete graphical interface with all the bells and whistles. As much as I enjoy working at the system level of any machine, I must admit that I am having a great time exploring the World Wide Web, FTPing and TELNETing all over the world.

For the TRS-80 crowd, though you cannot handle the graphics, the Internet is still a place that you should explore. Use Omniterm, as Gary recommends, and experience computing of the nineties. It is probably the last bastion of complete freedom. I call it the Wild West — anything goes, good or bad. It won't last long. The Post Office is already trying to muscle in on e-mail, and politicians who don't know the difference between a byte and a bite are already trying to pass legislation to regulate the information. But that is a story for another day and another magazine.

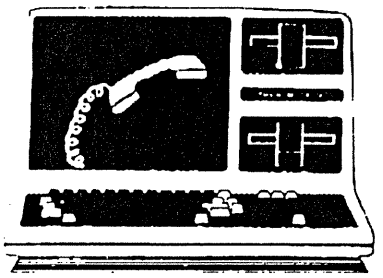
Before I close, a very, very special THANK YOU to the artist who drew the cover for this issue. His name is Ben Byer — and he is 14 years old.

Eat your heart out guys!

And now.....Welcome to TRSTimes 8.4

RETURN TO THE INTERNET

by Gary W. Shanafelt



Back in the July/August 1994 issue, I wrote a rather long-winded article about hooking up my Model 4 TRS-80 computer to the Internet through our school VAX. A lot has changed since then; the Internet has become so well known that you can simply refer to it as the "Net," and everyone will know what you mean (though how the term "surfing" got transferred from the beach to the Information Highway is still beyond me). The changes make it more complicated to get access from your TRS-80, though it's still possible.

The main change is that you need much better (or fuller) VT100 emulation capability than you did a year ago. Then, Mel Patrick's FastTerm was fine (and for simple applications, it's still fine). But it supports only the minimal level of VT100 emulation. Most internet services expect at least ANSI command support, which the VT100 includes, so that a sequence like "[0m" puts the cursor at a certain place on your screen rather than being displayed as "[0m". FastTerm gives you this level of compatibility. These ANSI commands are also supported by Richard VanHouten's programs VT100/CMD and ANSITRM4/CMD.

Unfortunately, Net services — and particularly WWW (World Wide Web) connections — expect not just basic VT100 formatting, but support of the FULL command set. This allows pull-down menus, scrolling of screens back and forth, etc. If you call the student telephone directory of Texas Christian University with FastTerm, for example, all the names end up on the same line because the VT100 scrolling command isn't supported.

So what do you do? I've been experimenting with some old commercial terminal programs for the Model 4 which support VT100 emulation, and gotten them to work up to a point. But since the companies which made them no longer exist, finding a copy is a major hassle (I ended up posting messages on CompuServe for a used copy). The one which seems to work the best is Omniterm Plus by Lindbergh Systems (not the original Omniterm), with Telexpress' Teleterm a runner-up.

Of course, the Model 4 hasn't got the hardware to accomodate all the options of the VT100 standard, like displaying boldface type or underlining on the screen. And while it can display reverse video, it is at the cost of switching off all high-bit characters (ASCII 128-255). The various emulators handle this in different ways. Teleterm displays only reverse video commands and ignores anything in boldface. It employs a reverse video cursor — a nice touch in keeping with what you would see on a VT100 screen. Omniterm uses reverse video for both reverse video and boldfacing. Neither can display colors or high-resolution graphics images. If you want these, it's time for new hardware.

The main advantage of Omniterm Plus' VT100 emulation mode is that it is the only VT100 emulation I have come across on the Model 4 which compensates for an obscure but potentially disastrous quirk in the VT100 command set. As every longtime TRS-80 user knows, our computers automatically execute a linefeed (LF) as well as a carriage return (CR) whenever they encounter the carriage return command. The VT100 (like MSDOS) doesn't: to get the same effect as a CR on a TRS-80, it sends both codes together — CRLF. If a CRLF is received by a TRS-80 terminal program, the result is double spacing where there should be single spacing. A good VT100 emulator thus allows you to suppress automatically any LF coming after a CR.

However, occasionally a mainframe program sends a sequence of CRLFCR, with only the first CR being followed by a LF. Why? Perhaps out of sheer perversity: I really don't know. One suggestion is that in the old world of Teletypes, putting several CRs in a row was a way of padding data at a time of slow transmission speeds, like adding null values between data bytes. On a non-TRS-80 terminal, multiple CRs would have no effect; they would just move the cursor to the left of the same line several times without affecting the linespacing. But several CRs on our computers result in the cursor actually moving down the screen — double or triple spacing where single spacing was intended — even if your emulator is suppressing all the LFs it gets. This is disastrous on programs (and I've encountered several) with pull-down menus and complex cursor positioning for selecting command options; any multiple CR sequence makes the whole screen quickly unreadable.

There should be ways around this, like going into the translation tables of a program like Teleterm and switching the LF and CR values or translating 0DH (the CR code) to something else. But nothing I've managed so far seems to have any effect, or has worse effects. Omniterm, however, is apparently coded to check for multiple CRs and to automatically delete them. In any case, I've had no problems with its screen formatting, to the extent that I can even call up our VAX text editor, edit full-screen ASCII text from my Model 4, moving the cursor up and down the screen at will, and save my changes back to the VAX.

But Omniterm is not really the TRS-80 Internet program for the 90s. In fact, some of its features are antiquated even by TRS-80 standards. It doesn't come close to FastTerm's sophisticated scripting capabilities or use of the full 128k memory, if you have it, as a text buffer. The buffer must saved to disk, and there's an aggravating tendency for the save-to-disk feature to switch on by itself if certain code sequences come in from your telephone line. Worse, it (like Teleterm) supports only checksum XMODEM file transfers, not the more accurate CRC option (FastTerm gives you both). CompyServe and other services now seem to support only the CRC version, so this obviously limits your ability to transfer files to and from your computer. With Teleterm, you can use the DOS command option to run Mel Patrick's stand-alone CRC XMODEM utility, but since that requires *CL to be set to COM/DVR which Omniterm doesn't use, if you try the same thing from within Omniterm you'll hang your whole system.

Some of Omniterm's quirks can, however, be overcome. One of the strangest is the non-use of the characters [,], and _. For some reason, the program will not allow you to type these characters in VT100 emulation mode (though they type fine in TTY mode, and though the translation tables show no change of the relevant codes). To make things worse, our computerized university library system now incorporates some commands not available on the original VT100, like asking you to hit the NEXT key to scroll listings. The code for the NEXT key, present on VT320 and VT420 keyboards, is <ESC>[6~. While Omniterm allows you to translate individual bytes, it has no macro facility for sending four bytes (to emulate that NEXT key) with a single keystroke. What to do?

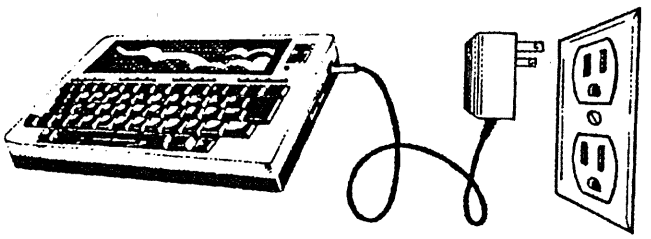
This now gets rather technical, but even if you don't have Omniterm Plus the solution might come in handy. First, in Omniterm's transmit (outgoing) translation table, I took the high bit codes FC, FD, and FE (hex) which I would normally never send from the keyboard to a remote machine, and changed them to the codes for [,], and _. Then, I returned to the DOS and created a KSM (keystroke multiply) file, following the instructions in the DOS manual. I assigned the NEXT code sequence to the N key, with the code for [being input as FC. While I was at it, I input the code for several other VT320/420 keys, as well as each of Omniterm's non-accessible characters (I assigned the underscore _, to the U key). Then, I put my new key definitions on-line with the KSM/FLT filter (though it took me a while to figure out the manual's rather obscure directives for how to do this). Finally, I created a JCL file to automate future installation of the filter and, for good measure, SYSGENed the whole thing to the disk.

Now, when I run Omniterm and need to hit the NEXT key, I hit CLEAR N and the KSM code is sent out. FC is encountered in the transmit translation table and changed to [, the host system thinks the NEXT key has been pressed, and the screen scrolls just as if I were using a VT320 system. Or if I want one of the forbidden keys by itself, for example, to type a name like G_Shanafelt, I just hit CLEAR U for the underline character (which is echoed back to the screen from the host). Not bad: I've gotten around one quirk of Omniterm and also figured out how to customize a VT100 emulation to handle VT320/420 keyboards.

The next step is to figure out how to get Omniterm to handle CRC XMODEM file transfers. The ideal, of course, would be to get Mel Patrick to rewrite FastTerm to support the complete VT100 command set, but that's not likely anytime soon. Mel's hard drive burned out some time ago, so he's now completely out of the TRS-80 world. But as the Internet gets more demanding, a new Model 4 terminal program for the 90s is clearly something people with major programming ambitions should be thinking about.

POKING MODEL 100

by Tony Anderson



TELCOM FUNCTIONS:

POKE 63064,0	sets HALF DUPLEX mode.
POKE 63064,255	sets FULL DUPLEX mode.
POKE 63065,255	sets ECHO to printer. (Turns ECHO on)
POKE 63065,0	sets not ECHO to printer. (Turns ECHO off)
POKE 63066,1	send a LF (line feed) after a CR (carriage return) Some devices need them.
POKE 63066,0	reset to No LF after CR

To be able to clear the screen when in TELCOM, type the following two pokes in BASIC, before you use TELCOM:

POKE 64268,49 and POKE 64269,66

Then when you want to clear the screen, just press the F6 key.

To reset to normal operation, in BASIC, type:

POKE 64268,247 and POKE 64269,127

It is often useful to poke TELCOM status into memory from a basic program, for example in a file transfer program. "Status" is held in RAM in locations 63067 to 63071, in a five character field. The leading character is the baud rate, and if this is all you want to change, you can use the direct command: POKE 63067,"M" (or whatever you want to poke there).

If, for example, you want to reset TELCOM to modem status before leaving a file transfer program, you could add this short routine to your program:

```
1000 REM  Pokes "M7I1E" into "Status"
1010 FOR A = 63067 TO 63071
1020 READ B : POKE A,B
1030 NEXT
1040 DATA 77,55,73,49,69
1050 REM  M 7 I 1 E
```

The values in the DATA statement are the ASCII value of the character you want to poke into memory. Change the values in the DATA statement to set any "Status" required.

There are two changes that will enhance TELCOM's basic functions by providing two additional function key actions, using F6 and F7. While in TELCOM or TERM mode, pressing F6 will respond with the current number of bytes free in RAM; and F7 will give you a list of file names in RAM, similar to the F1-FILES command in BASIC. Here is a short routine which will make the necessary pokes:

```
1 FOR I = -1268 TO -1265
2 READ X : POKE I,X
3 NEXT
4 DATA 172,126,58,31
```

While you will probably never need this, the dialing pulse rate is stored at RAM address 63019, and can be set, or changed with a POKE.

POKE 63019,1 = 10 pulses/second
POKE 63019,20 = 20 pulses/second

BREAK KEY - CONTROL-C - FUNCTION KEYS

The interrupt routine can be turned on and off at RAM location 63056. The command to disable one of these functions will disable them all. There is no way to "selectively" disable one or the other.

POKE 63056,128	disable the Break Key, Control-C and Function Keys.
POKE 63056,0	re-enable the same functions.

SECRET STORAGE SPACES

The following warrants a "Use at your own Risk" disclaimer. There are 36 bytes of RAM which can effectively, be rendered invisible to the operating system, and will be protected from everything short of a cold start; which can be used to store an identification name or number permanently in RAM. You might also find other uses for this 36 bytes. If you have installed any alternate ROM in your M100, then these 36 bytes are not available to you, as they are the bytes that hold the M/L routines for accessing the optional ROM.

If you have used Rick Perry's ALARM or ALARMS programs, first POKE 62975,201 to completely disable any residual code still in memory from these programs. Then, do these two:

POKE 62981,201 and POKE 63018,255.

Once these two pokes are done, addresses 62982 to 63017 are free, and can be used for anything you want. The above pokes remain effective until a cold-start occurs. What's happening here is that the original code from 62981 to 63011 checks for the existence of an optional ROM everytime you power up. If ROM is installed, then the value 255 is stored in 63018, and the name of the ROM (for the Menu) will be placed in 64164-64171. Code from 63012 to 63017 is used when choosing the option ROM from the Menu.

The POKE 62981,201 puts a M/L RETurn there, which effectively makes the M100 think that the optional ROM is installed. You need the POKE 63018,255 to prevent a cold-start on power-up.

Whatever you put in those 36 bytes will stay there, regardless of whatever BASIC or normal machine language programs you are running. Obviously, you use POKE commands to get whatever you want into those locations.

Interested in more free RAM to poke stuff into? Space used in TELCOM to store the previous screen (64704 to 65023) seems safe, although it is also used for lots of different things, including some Menu Directory stuff. The lowest addresses might be useful. Also, the optional ROM LUCID makes use of some of this area, if it is installed. The MDM & COM receive buffer (65350 to 65413) can be used as long as MDM/COM is not being accessed for input or output. That's 384 bytes to poke away in, at least for temporary use during a running program. If a running program does a CLEAR or LOADM, then all variables are erased. Before using either of these commands, a program can poke values/characters it wants to save into these unused RAM areas, then PEEK them back later.

BASIC PROGRAM POKES

The following pokes would generally be used from a BASIC program, but could also be used in direct mode, if needed.

As you know, SCREEN,0 will turn off the label line. However, in some programs, it's also desirable to prevent the user from pressing the LABEL key to turn the label line back on. POKE 64173,0 will disable the Label Key, so that pressing it will have no effect. If the label line is on, it will stay on. If it is off, it will stay off. However, the effect is temporary; returning to the menu will automatically re-enable the LABEL key.

POKE 64173,0	disable the Label key.
POKE 64173,1	enable the Label key.
POKE 63048,175	turn on Reverse Video display, until cancelled.
POKE 63048,0	turn off reverse video, and return to normal display.
POKE 65348,175	turns Sound Off.
POKE 65348,0	turns Sound On.

To place a string of characters in the keyboard buffer, just as if they had been typed in from the keyboard, follow the following proceedure: (Assume that the string you want to put in the buffer is A$) Use the following line, either in direct mode, or written as a line of code in your program.

```
FOR I = 1 TO LEN(A$):
POKE 65449+2*I, ASC(MID$(A$,1)):
POKE 65450+2*I,0:
NEXT:
POKE 65450,I
```

This puts the characters from A$ into the odd addresses from 65451+, zeroing out the even addresses which are reserved for FN key entries; then puts the number of characters into 65450. This is very useful before a program does a SAVEM, LOAD, or MERGE, which would cause the program to stop execution. Using the above code, with A$ = "RUN"+CHR$(13) will make the program restart automatically. But be careful, the length of A$ must

be <= 32 characters.

Here's a quickie poke that will initialize the RND seed based on the time of day:

```
J = 63795:
FOR I = 64634 TO 64637:
POKE I,16*PEEK(J) + PEEK(J+1):
J = J+2:
NEXT
```

And, this one will re-seed the random number generator with one of 125 possible values:

POKE 64634,PEEK(63791)

The following poke will send the next PRINT statement in a BASIC program to the printer port, instead of to the LCD screen. It works only on the next PRINT statement in a program, so has limited usefullness, but might be useful in a trace or debugging program. Just add it into any BASIC program as a BASIC statement.

POKE 63096,1

RAM address 64228 is the place to intercept the print routine, just before it prints a character. PCSG and others use that location for an intercept to add line feeds. But, if one wanted to, they could intercept the character and re-direct it to another location, to the serial port, for example.

If you wish to disable printer output completely, in order to prevent a program lock-up if the printer is not on and ready, the following two pokes will bypass the printer port:

POKE 64228,136 : POKE 64229,20

To return to normal, use:

POKE 64228,243 : POKE 64229,127

The "bypass" poke, will apparently also prevent the PRINT key in the function key row from being used.

Alternately, if you wish to test the printer port to determine if the printer is powered up, and ready to accept data, you can use the following statement in a BASIC program:

```
IF (INP(187)AND6) <> 2 THEN BEEP :
PRINT "Printer Not Ready" :
STOP
```

Or you can devise alternate tests: a 0 means the printer is not ready; a 2 means it is; and a 6 means it is not connected.

```
IF (INP(187)AND6) = 0 THEN (Printer not ready)
IF (INP(187)AND6) = 2 THEN (Printer ready)
IF (INP(187)AND6) = 6 THEN (Printer not
connected)
```

Of course you realize, any address which can be "poked" can also be "peeked" to determine what is currently happening at that location, or what will happen depending on the value set at that address.

C PROGRAMMING TUTORIAL

Part 7

by J.F.R. "Frank" Slinkman

In the last article, I issued a call for readers to send in short- or medium-length BASIC programs to be converted to C. I thought (and still think) the side-by-side comparison of BASIC and C program code would be a good way to help readers make the transition from BASIC to C.

However, I only received one suggestion, and that was a request for a graphics format conversion program which would be extremely difficult for even an experienced professional programmer to write, and way beyond the scope of these articles.

However, I do appreciate the suggestion, and am still looking for submissions.

Also, I apologize for fact that Part 7 was not ready for inclusion in the last issue of TRSTimes. It won't happen again.

If you'll recall, the last article was concerned with disk directory files with a program, "prog11/c," that produces a report somewhat similar to the one generated by the TRSDOS/LS-DOS 6 DIR command.

This article will be devoted to a routine which performs similarly to the BACKUP command, but with some important differences.

The program, "bydate/c," will not copy password-protected or invisible files, but it does copy non-protected, visible files in date and time order.

This program makes extensive use of many of the less commonly used standard header files which pertain to disk drives and disk files, and makes extensive use of allocated memory.

It demonstrates the use of "block files" in C. These are roughly analogous to the "random access" files in BASIC, but with more speed and power.

Before we look at the code, we need to think about what the program needs to do:

1 Analyze the command line to check for errors or omissions and parameter validity, and report any errors to the user;

2 Make sure the user-specified source and destination drives are both valid and available, and report any errors to the user;

3 Read the directory of the source drive, skipping all password-protected and invisible files;

4 Sort the list of files to be copied in the desired order;

5 Copy the files from the source drive to the destination drive in the correct order, one by one, except when a file of the same name exists on the destination drive; and

6 Keep the user informed, via messages displayed on the monitor screen, of the progress.

O.K. Here is the code for the program bydate/c:

```
/* Title:       bydate.c
 * Author:      J.F.R. "Frank" Slinkman
 * Date:        01-Jul-94
 * Compiler:    Pro-MC
 * Function:    Copy files in date/time order from
 *              one disk drive to another
 * Copyright:   Reserved
 */

#include   <attrib.h>
#include   <dirent.h>
#include   <fcntl.h>
#include   <malloc.h>
#include   <stat.h>
#include   <stdio.h>
#include   <unistd.h>
#include   <ustat.h>
#include   <utime.h>

#option INLIB
#option REDIRECT  OFF

#define FILEDAT struct fileinfo
#define uint    unsigned int
#define ulong   unsigned long

void    get_names(), copy_files();

int     sorc, dest, count = 0;

struct attrib at_buf;

FILEDAT {  time_t  date;
           ulong   size;
           char    name[15];  }  *array;

/*=*=*=*=*=*=*=*/

main( argc, argv )
int argc; char **argv;
{
   printf( "\x1c\x1f" ); cursor( 23,2 );
```

```
    puts( "BYDATE - File Organization Utility" );
    cursor( 12,3 );
    printf( "(c)1994 - J.F.R. \"Frank\" Slinkman - " );
    puts( "All Rights Reserved!\n" );

    if ( argc != 3 )
    {  puts( "usage\tbydate :s :d\n\nWhere:\
    \":s\" is the source drive number\n\
    \":d\" is the destination drive number" );
       exit(EOF);  }

    sorc=argv[1][0]==':'?argv[1][1]-'0':argv[1][0]-'0';
    dest=argv[2][0]==':'?argv[2][1]-'0':argv[2][0]-'0';

    if
(sorc<0||sorc>7||dest<0||dest>7||sorc==dest)
    {  puts( "Illegal drive number" );
       exit(EOF);  }

    {
       struct ustat ubuf;

       if (ustat(sorc, &ubuf) || ustat(dest, &ubuf))
       {  perror( "ustat()" ); exit(EOF); }
    }

    if ( !( array = calloc( 256, sizeof(FILEDAT) ) ) )
    {  puts( "Memory error" ); exit(EOF);  }

    get_names();

    realloc( array, count * sizeof(FILEDAT) );

    copy_files();
    free( array );
}
/*=*=*=*=*=*=*=*/

void get_names()
{
    static char     drive[2] = { "x" };
    FILEDAT         *ara_ptr;
    DIR             *dirp;
    struct dirent   *entry;
    struct stat     sbuf;
    int             kill, i;

    ara_ptr = array;
    *drive = '0' + sorc;

    dirp = opendir( drive );
    if ( !dirp || dirp == EOF )
    {  puts( "opendir() error" ); exit(EOF);  }

    while ((entry = readdir(dirp)) && entry != EOF)
    {  stat( entry->d_name, &sbuf );
       strcpy( ara_ptr->name, entry->d_name );
       ara_ptr->date = sbuf.st_mtime;
       ara_ptr->size = sbuf.st_size;
       ++ara_ptr;
       ++count;
    }
    closedir( dirp );

    for ( kill = 0, i = 0; i < count; i++ )
       if ( gattrib( array[i].name, &at_buf )
         || at_buf.a_vis )
       {  array[i].date = 0xffffffffL;
          ++kill; }

    qsort( array, count, sizeof(FILEDAT), compare );
    count -= kill;
}
/*=*=*=*=*=*=*=*/

void copy_files()
{
    extern char *alloc();

    struct utimbuf  tbuf;
    FILE    *s_file, *d_file;
    char    *copy_buf, dfilname[15];
    ulong   fil_siz;
    uint    buf_siz;
    int     i;

    if ((buf_siz=((freemem()-1024)&0xff00)) > 0x7f00)
       buf_siz = 0x7f00;
    if ( !( copy_buf = alloc( buf_siz ) ) )
    {  puts( "Memory error" );  exit(EOF); }

    for ( i = 0; i < count; i++ )
    {  strcpy( dfilname, array[i].name );
       *(dfilname+strlen(dfilname)-1) = '0' + dest;

       if ( !( access( dfilname, F_OK ) ) )
          continue;

       tbuf.modtime = array[i].date;
       fil_siz = array[i].size;

       cursor( 0, 11 );
       putchar( '\x1f' );
       cursor( ( 76 - strlen( dfilname ) ) >> 1, 11 );
       printf( "Writing %s", dfilname );

       s_file = open( array[i].name, O_RDONLY );
       d_file = creat( dfilname, 0777 );

       lseek( d_file, fil_siz - 1L, 0 );
       write( d_file, "\0", 1 );
       lseek( d_file, 0L, 0 );
       while( fil_siz >= buf_siz )
       {  read( s_file, copy_buf, buf_siz );
```

```
        write( d_file, copy_buf, buf_siz );
        fil_siz -= buf_siz; }

      if ( fil_siz )
      {  read( s_file, copy_buf, (int)fil_siz );
         write( d_file, copy_buf, (int)fil_siz );
      }
      close( s_file );
      close( d_file );

      utime( dfilname, &tbuf );

      gattrib( dfilname, &at_buf );
      at_buf.a_mod = AUNMODIFIED;
      sattrib( dfilname, &at_buf, ASETFLAGS );
   }
   free( copy_buf );
}
/*=*=*=*=*=*=*=*/

int compare( a, b )
FILEDAT   *a, *b;
{
   if ( (ulong)a->date > (ulong)b->date )
      return 1;
   else if ( (ulong)a->date < (ulong)b->date )
      return -1;
   else
      return strcmp( a->name, b->name );
}
```

First, there is a long list of standard header files to be included. Among these are:

Attrib/h, which supports two non-standard functions, gattrib() and sattrib(), which let us examine and alter system-unique file attributes, such as password protection levels, invisibility, the "mod" and "created" flags, etc.

Malloc/h, which contains forward declarations for the standard memory allocation functions but, importantly, not for the non-standard alloc() function.

Ustat/h, which defines both the ustat() function and a data structure to store information about a disk drive, such as its physical characteristics, capacity, free space, etc.

Next are a couple of non-standard "options" which tell the compiler to search the IN/REL library for certain non-standard functions and to eliminate the code which supports standard I/O redirection, to make final program size smaller.

These are followed by three text-substitution #defines, and the forward declarations for the functions get_names() and copy_files();

Next we globally define three ints, "sorc" to hold the source drive number, "dest" to hold the destination drive number, and "count" to hold the number of files to be copied.

Then we globally define two structs: "at_buf" as a struct of type "attrib," as defined in the "attrib/h" header file, and "array," as a pointer to structs of type "fileinfo" (via the #defined FILEDAT name) just as was done in the program "prog11.c."

Now, inside the main() function, the first thing we do is display the program's "billboard."

Next we check "argc" to get the number of command line arguments. If the number is anything other than three, there's an error; so the user is given the "usage" message, describing the correct syntax, and the program terminated.

The next line of code allows the user a little flexibility in entering drive numbers. He is permitted to enter just the number, or the number preceded by a colon (e.g. ":4") as is usual for TRS/LS-DOS.

If the first character is a colon, then the value of the second character is picked up. If not a colon, then the value of the first character is picked up. In both cases, the value of the ASCII character "0" (zero) is subtracted from the value to get the actual drive number.

This line could also have been written:

```
if ( argv[1][0] == ':' )
   sorc = argv[1][1] - '0';
else
   sorc = argv[1][0] - '0';
```

The next line does the same thing for the destination drive number.

Next the drive numbers are checked for validity. If either are less than zero or greater than 7, then the user has entered an invalid value. Also, the two values are equal, an error is assumed since it makes no sense to copy files to and from the same drive.

If anything is found to be wrong with the drive numbers specified, the user is given an "Illegal drive number" message, and the program is terminated.

Next, the ustat() function is used to check both the specified drives. If ustat() returns anything other than a NULL, there is an error -- probably that the

drive is not in the system or has no floppy disk installed. In such an error occurs, the perror() function is used to inform the user of the error, and the program is terminated.

Notice that "ubuf," the struct of type "ustat," exists only in the small block of code defined by opening and closing braces. We did exactly the same thing in the program "prog11.c."

O.K. Assuming we've gotten this far, both drive numbers were entered correctly and both specified drives are available and ready; so we now use the standard calloc() function to allocated a zeroed block of memory large enough to hold 256 structs of type "fileinfo," just as we did in "prog11.c."

Now the get_names() function is called to read the directory of the source disk. Note this function is similar to the get_names() function in "prog11.c," and the compare() function in "bydate.c" is identical to the one in "prog11.c." You will probably save time and effort by starting with prog11.c and editing it than by typing in "bydate.c" from scratch.

In get_names(), after the directory file is opened, a "while" loop is used to read all directory entries. First, the standard stat() function loads the information about each file into the struct "sbuf." Then information from "sbuf" is selectively copied to members of the array of structs of type "fileinfo" pointed to by "array" and "ara_ptr."

For each file, the file name, most recent mod date and time, and the file size in bytes are stored in the "array" array element.

After all directory entries have been read and the directory file closed, a "for" loop is used to check each file to see if it's either password-protected or invisible.

Such files are counted by the "kill" variable, and are flagged for deletion from the list by having their dates changed to 0xffffffffL, which will cause them to sort to the top of the array, and thus out of the array when the global variable "count" is reduced by "kill" after the sort.

Back in main(), the "array" array of structs is altered via the realloc() function, completing the deletion of non-qualifying files and freeing up the RAM space not needed to store data for qualifying files.

Next we call the copy_files() function, in which we used block I/O for the first time.

In copy_files(), the first thing we do is use the non-standard freemem() function to learn how much free RAM is available for a disk file buffer. If more than 0x7f00 (127 x 256 = 32,512) bytes are available, we limit the buffer size to 0x7f00 for two reasons:

First, the read() and write() commands are limited by the positive side of the range of a short signed int. In other words, the largest number of bytes which can be read or written at one time is 32,767. Second, we want the number of bytes read and written to be evenly divisible by 256 as often as possible, because doing that affords the fastest disk I/O speeds.

Once the buffer size has been determined, it is allocated via the non-standard alloc() function. Notice that alloc() was declared as "extern" in the beginning of the copy_files() function. We did this because no forward declaration for this function is included in either the "stdio/h" or "malloc/h" header files.

On our systems, because both ints and pointers are only 16 bits, we could have gotten away with not declaring alloc() in advance. In fact, we did exactly that in "prog11.c."

But on most computers these days, pointers are 32 or 64 bits while short ints are still 16 bits. For this reason, portability considerations require you to be aware that A POINTER IS NOT AN INT!

Next we set up a "for" loop to go through the list of files in the "array" array of structs, one by one.

In this loop, we first copy the source file name to the "dfilename" string using the standard strcpy() function. We know that the last character in this string will be the drive number, and that the drive number copied is the number of the source drive.

So we use the standard strlen() function to get the length of the string, subtract one from it to point to the character containing the drive number, and change that character to the ASCII representation of the destination drive number.

For example, suppose we copied the string, "zork/bas:0" meaning that "dfilename" would point to the "z." The strlen() function will return 10 as the length of this string. Thus dfilename[9] (the 10th character) must contain the source drive number, which is zero in this case.

Changing that character therefore changes the drive spec in the string.

Also, remember that

*(dfilename + strlen(dfilename) - 1)

is just another way of writing

dfilename[strlen(dfilename) - 1]

except that the first way often generates more efficient code.

Now that we've built the filespec with the correct destination drive number, we use the standard access() function to see if it already exists on the destination drive. If it does, the "continue" statement skips all the following statements in the "for" loop, thus skipping this file.

If the file is to be copied, the next thing we do is store the Unix time in the "tbuf" struct which will be later used by the utime() function to make the copied file's date and time the same as that of the source file.

After that, we get the file size in bytes from the current "array" array element and store it in the variable "fil_siz."

Next, we set the cursor to the first column of row eleven and issue a "clear to end of frame" command by using putchar() to send the value 0x1f. These two statements do the same thing as the BASIC command:

PRINT @ (11,0), CHR$(31);

The next statement positions the cursor to the column on row 11 which will center the the "Writing filename" message. Here, ">> 1" is used to shift the column number one bit right, which is the same as dividing by two, but much faster. Seventy-six is used instead of 80 to adjust for the length of the word "Writing" plus a following space.

Then the message is displayed, properly centered, for the user.

Next, both the source and destination files are opened using the open() and creat() functions. These functions are only for block I/O operations, and are different from fopen() and the other functions used for byte stream I/O.

At this point, you need to refer to the docs (the Pro-MC manual or other) to learn about both the open() and creat() functions.

But, in a nutshell, open() only works on existing files, and creat() must be used to create a new file.

Next we use a "trick" to speed up the file writes. Since we know how large the destination file will be, we use the standard lseek() function to position the file control block to the last byte, and write a single byte of zero to the file. This forces the disk directory to be updated. This, in turn, eliminates the need for the directory to be updated each time a new disk storage granule is needed as would normally be the case, and which takes quite a bit of time, especially on floppy disk drives.

After that is done, lseek() is used again to position the file control block back to the start of the file before any "real" data is written.

Notice lseek() is used for block files -- not fseek(), which is only used for stream files.

Now it's time to actually copy the file. A "while" loop is used to break large files up into smaller chunks consistent with the size of the buffer.

If the file is smaller than the buffer, the "while" loop will be skipped. If equal to or larger than the buffer, then one buffer's worth of data will be read from the source file and then written to the destination file. Then "fil_siz" is reduced by "buf_siz." Then the "while" loop condition is evaluated again, and the operation repeated as often as needed until "fil_siz" becomes less than "buf_siz."

Then, when "fil_siz" is less then "buf_siz" the entire remainder of the source file is read into the buffer, and then written to the destination file.

The files are then closed using the standard close() function [N.B.: close(), not fclose()]. Then the standard utime() function is used to change the destination file's date and time to match that of the source file, and the non-standard gattrib() and sattrib() functions are used to reset the system's "mod" flag.

To fully understand this, you need to study page 14 of the Release 1.6 upgrade portion of your Pro-MC manual.

Before returning to main(), the RAM allocated to the disk buffer is released via the standard free() function.

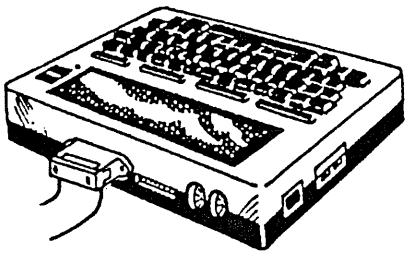
Now, back in main(), the last thing the program does is to use the standard free() function to release the RAM allocated to the "array" array of structs.

And, finally, compare() function, which is identical to compare() in "prog11.c," simply tells the qsort() function how to rank the items being sorted, namely in date order, and alphabetically within equal dates in the event two files have identical mod dates and times.

HINTS & TIPS

DELETION HELPER FOR THE MODEL 100

by George Mueden



When the RAM in your Model 100 get crowded and you must make room by deleting files, it helps to have a program that lets you select files to be killed by moving the cursor, or by having them presented to you with a YES or NO choice to be answered with a single keystroke. However, such programs take a good deal of space themselves. Here is one that tells you how much numeric and string space you have and saves you from spelling errors by listing the files. As a Basic program, it occupies only 123 BYTES of ram.

```
10 CLS:PRINT:FILES
15 PRINT FRE(0);FRE("");"Exit with 'F8'"
20 INPUT"Delete which .DO file";D$
25 IF ASC(D$)=77 THEN MENU
30 D$=D$+".DO":KILL D$:GOTO 10
```

NEWDOS/80 BUILD GENERATOR

by George Saladino

Have you ever tried to follow the instructions in the NEWDOS manual to make a Build or JCL file? Lots of luck! Well, here is a short program that generates a build file from Basic. To run your file, just type "DO", space and the name of your build file while in DOS. If your build file is named "START", you can have it run automatically when the disk is booted up by entering "DO START" after the auto command. More on build files in a moment, but first the program listing.

```
5 'NEWBUILD/BAS
10 CLS
20 LINE INPUT"Name of BUILD FILE ";N$
30 N$=N$+"/JCL"
40 OPEN"O",1,N$
50 PRINT"Type in BUILD commands. To end, type
* on a blank line"
60 LINE INPUT A$
70 IF A$="*" THEN 100
80 PRINT#1,A$
90 GOTO 60
100 CLOSE
110 CMD"S"
```

Line 20 asks for the name you want to name your file.
Line 30 adds the "JCL" extension to the filename so you do not have to include it.
Line 40 opens the file so you can write on disk.
Line 50 explains how to use the program. It also tells that typing an asterisk (*) at the beginning of a blank line jumps will end the program.
Line 60 accepts your input.
Line 70 tests for the "*". If found the program jumps directly to line 100.
Line 80 writes the input to the disk file.
Line 90 puts the program in a loop.
Line 100 close the disk file.
Line 110 sends you back to the operating system.

A build file is the same as a "BAT" file in MS-DOS. Most operating systems allow some sort of file to allow multiple operations from the DOS.

I have a lotto program in Basic that picks six numbers at random and displays them. One version is a continuous version for demonstration purposes. It asks, before operating, the highest number you desire. If I want to have it pick between 1 and 48, I enter 48.

With a build file I can set up the program to operate upon boot up. My build file will look like this:

BASIC RUN"LOTTO/CON" 48 PRINT CHR$(13)

The computer fist loads in Basic, then my program. 48 is entered for the question asked for the highest number. The "PRINT CHR$(13)" puts in a line feed after the 48 and starts the program running.

This is just a simple example of a build file. It can be most any length and can do quite a few operations.

MORE NEWDOS/80

ZAP THE ?

by Bob Seaborn

To remove the "?" prompt at the end of a directory page, clears the screen before the next page is displayed, and allows any key to be pressed when the next page is requested instead of only <ENTER>.

<BREAK> is still used to abort the DIR command.

SYS8/SYS,03,86 (I)
from:
3E3F CDA5 50CD 4900 3DCA 2D40 FE0C 20F5

to:
CD49 003D CA2D 40CD C901 C3A5 5000 0000

The next zap tidies up the directory display when in expanded (,A) mode.

SYS8/SYS,04C8 (I)

from:
2E2E 2E2E

to:
2020 2020

SYS8/SYS,04,F3 (I)

from:
2E 2E2E 2E

to:
20 2020 20

15 laws of Computer Operation

Humor from the TRSTimes Vault

1 Any computer program which runs well, is obsolete.
2 A good program is accompanied by bad documentation.
3 The value of a computer program is inversely related to the weight of its output.
4 Program complexities grow to exceed the capabilities of the programmer responsible for it.
5 Any time a system appears to be working well, something has been overlooked.
6 What you don't do, is more important than what you do.
7 In any computer program, constants should be treated as variables.
8 Investments in system-reliability products always exceed the probable cost of errors they are designed to avoid.
9 The problem is not that computer salesmen are not knowledgeable, it's most of what they know isn't true.
10 If a system requires 'n' number of spare parts, there will be 'n-1' parts in stock.
11 Major software revisions are always requested after system installation is completed.
12 Installation and operating instructions are discarded with the shipping containers.
13 Any component part requiring the most frequent service or adjustment will be the least accessible.
14 Undetectable errors are infinite while detectable errors, by definition, are finite.
15 Nothing is impossible for the person who doesn't have to do the work.

MY LITTLE ZEVEN

by Chris Edwards

Any one remember Tandy Line Printer 7 (26-1167)? Very noisy, 30 cps, slow-slow single direction print and strange characters, lower case p's and g's not in line with the normal characters. Its operating manual is 19 pages, not thick books in the case of modern printers. Control codes are graphic mode, line deed, carriage return, double-width or normal characters. No form-feed, underline or font change codes! My sister referred the noise to a cat being slowly strangled. The noise do not bother me - I have been deaf since birth! I am quite fond of my printer.

Zeven is still printing, has eaten up its sixth ribbon and I have three more black carrots to feed it! Recommending paper up to two copies plus carbon tractor feed only. How about sticky labels? Ho-ho! Nigh impossible! I tell you what I still do. Get a few pages of tractor feed paper, sellotape, and a roll of gummed labels. Use sellotape to tack top and bottom stretch of gummed labels to the paper. Print as for sticky labels. Use you tongue to stick the labels onto envelopes. Currently printing from Tandy Models 1, 100, 200, 4P, and 1400LT. The Tandy TRS-80 catalogue 1981/82 (RSC-6) describes the Zeven as: "Low Cost, Full-performance Printer with Dot Addressable Graphics 5.32" x 16" x 8.25" weight 8.5 pounds". Full performance - my foot! Ho-ho!

YET ANOTHER TRSDOS 1.3 PATCH

by Andy Levinson

Cleaning out my closet, here is another patch for fans of TRSDOS 1.3. Unlike most DOS's, TRSDOS 1.3 "cleans out" an entire directory entry when it kills a file. That makes recovery from an accidental kill difficult if not impossible for most people. Even the "unkill" feature of Super Utility Plus cannot save the file.

With the following patch in place, only the "file activity" bit will be reset when TRSDOS 1.3 kills a file. That means that SU+ can "restore" a killed file on TRSDOS 1.3 just as easily as it does with any other DOS.

I am not going to explain here how to use SU+ or how to recover a killed file without it. Just install the following patch. When that magic moments arrives and you accidentally kill a file under TRSDOS 1.3, SU+ can now bring the file back from the dead. One important point: do NOT write to the disk until the file is restored. You could well overwrite the file you want to restore.

PATCH *3 (ADD=4FAE,FIND=3600,CHG=CBA6)

PATCH*3 (ADD=4FB7,FIND=EDB0,CHG=0000)

MEMORIES FROM 80 MICRO

by Jim Kajpust
via the Internet

I unpacked my 4P the other night and started to re-read some old 80 Micros--

from 80 Micro - May 1983

Total Access:
5 1/4" 40 track single sided drive $199
128K LNW-80 computer with RS-232/Parallel port and free 12" green monitor $1995
TRSDOS 2.3 disk and manual $20

American Small Business Computers:
Epson MX-80 dot matrix printer $419
Radio Shack 5 Meg hard drive $2395
TRS-80 Model III 48K with 2 40 track drives $1499.95

Other stuff from other places:
Hayes Smart Modem 1200 $565
Green screens $15.95
"true" lower case decender chip $18
Exatron Stringy Floppy $99.50
Omikron CP/M $199
"Custer's Revenge" video game
Mercedes Silver and The GAMER'S CAFE

Any of you guys remember this stuff - and the prices we paid??

MORE TRS-80 MEMORIES

by Eric Robert Greene
via the Internet

Yeah... there are some of us out here to remember that stuff AND (especially!) the prices we paid for it. I bought my first TRS-80 computer (a Mod I) in 1979 - 16K, Level II BASIC and I think it was right at $1000. Added the expansion interface and a floppy (and thought I had died and gone to heaven) for another exorbitant fee that I don't recall.

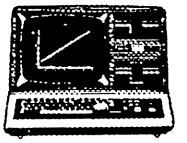
Got a copy of Dennis Bathory Kitsz's book on the "Custom TRS-80" and decended into the innards with soldering iron in hand to get lower case, improve memory addressing with the E/I and other fun projects. Added disk doublers and upgraded the system to 8" drives.

Played with TRS-DOS, DOSPLUS, LDOS and finally settled on NEWDOS-80 as my operating system of choice. And the first game - FLYING SAUCERS! Remember that hokey cannon shooting @@@ characters at the top of the screen?

I don't know if computing well ever be as much fun as it was in the "good old days!" How many people today know the satisfaction of adding a memory chip and seeing lower case suddenly appear on the screen? How many know the agonies of trying to load a 16k program from tape!

Started telecommunicating with an old Bell 103A3 modem in 1980 (this sucker had 11 circuit boards and RELAYS in it!) and put online one of the early bulletin boards in South Florida. Wrote the software for the fairly popular TRS-80 based BBS called "The Greene Machine". First time I downloaded Bill Blue's BBS list it showed about 200 boards in the US.

Computin' was a lot of fun in those days.... frustrating, but still a lot of fun!



THE MODEL I EMULATOR
by Sam Wayne

For those who have the Model 1 Emulator, you may discover, like I have that for some DOS's, you can only have a 35 track bootable disk. To try to use Vformat 40 or 80 it won;t work, or making a backup within the Emulator isn't doing it.

Now it possible to make that 40 or 80 track bootable. First make a copy of the Sytem Virtual drive (dsk) to another directory (MS_DOS COPY command), then do the following, Pressing ENTER at the end of each line. At the MS-Dos Prompt, (spaces included): DEBUG filename.dsk

```
F AB23 L 5 FC
F AB83 L 5 FC
E ABCC 5
W
Q
```

This will make it a 40 track System Disk. If you want an 80 track system disk, substitute "2D" for the three occurances of the number "5" above.

I wish to thank Jeff Vavasour and Brad Sampson for letting me know.

WHY CAN'T TANDY COMPETE
by Frank Durda IV

Editor's note: The following is borrowed from the Internet. Different people asked the questions — Mr. Durda provided the answers. When asked how he knew these things, Frank replied: "How do I know? Well, I was, uhh, sorta there!!

And all of us having read Roy Beck's articles in previous issues of TRSTimes should be aware that Mr. Durda was indeed there. He is the author of the Model 4P ROM and you will find his initials burned in there.

Why can't Tandy no longer compete?

While Best Buy or Circuit City get a 10 to 20% profit margin on computers they sell, on good days, Radio Shack demanded that the STORE profit margin never be less than 47%, and frequently the margin was as high as 300%. (Tandy Corp, manufacturing, transport and warehousing also get anywhere from 6% to 9% upcharges on things they make, transport or store.) Now, build a computer with the same or better features than the competition, but make it so cheaply that by the time that huge profit margin is added, it will still be price competitive. Take the Model 4D. Cost to build $447, which includes R&D royalties, software royalties, all manuals, manufacturing costs & profit. Sold for $1199 for when it first appeared in 1985 (1986 catalog) and finally lowered to $999 when they were closing them out. So even at closeout, the store margin was 123%. That will buy that store manager a lot of cruise vacations. With these other retailers everywhere and willing to run on much thinner margins, that scheme just doesn't work anymore. It wasn't that Tandy couldn't make money at computers. If it won't make tons of money, they aren't that interested.

Today, Tandy through Radio Shack and Computer City seems to be languishing. what happened?

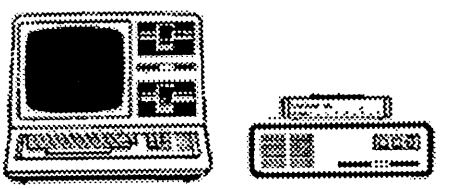
When you play "copy the people who are succeeding" all the time, you will get behind the curve and stay there. When Tandy decided that PCs were the way to go, they only implemented features that were already omnipresent in the industry. They did cut corners at every possible point to keep that profit margin nice and thick.

People claim of the shoddy tandy engineering, why can't tandy do better?

Actually given the constraints and the prime-directive of cost savings for higher store margins, the engineering was some of the best around. FYI, none of that store profit made it back to R&D. It stays in the stores.

AND, what is tandy doing about its bad rep. What would convince most tandy onwners to once again buy tandy?

Dunno, maybe they'll put the orange shag carpet back in the stores along with strobe lights and do another push on CB radios. Seriously, since Tandy doesn't make computers anymore, you find IBM laptops and AST computers (made in the ex-Tandy factory). Since these machines are sold at other retailers, Radio Shack changes the bundle, putting a different monitor, different amount of RAM or drive, and different software bundles or CD-ROM speed, so it is impossible to compare apples-to-apples with a similar model over at SAMs or Circuit City. This allows them to keep a big profit margin, although for computers it can't be anywhere near where it used to be. That is the only trick they have left to work with now.



ATTENTION TRSDOS 1.3 USERS!

ANNOUNCING "SYSTEM 1.5.", THE MOST COMPREHENSIVE 1.3. UPGRADE EVER OFFERED!

MORE SPEED!! MORE POWER!! NEW LOW PRICE!!

While maintaining 100% compatibility to TRSDOS 1.3., this upgrade advances DOS into the 90's!
SYSTEM 1.5. supports 16k-32k bank data storage and 4MGHZ clock speed (4/4P/4D).
DOUBLE SIDED DRIVES ARE NOW 100% UTILIZED! (all models).

config=y/n	creates config boot up
time=y/n	time boot up prompt on/off
blink=y/n	set cursor boot up default
line='xx'	set *pr lines boot up
alive=y/n	graphic monitor on/off
tron=y/n	add an improved tron
type=b/h/y/n	high/bank type ahead on/off
slow	2 mghz speed (model 3)
cpy (parm,parm)	copy/list/cat ldos type disks
sysres=y/n	disable/enable sysres
spool=h/b.size	spool is high or bank memory
spool=n	temporarily disable spooler
spool=reset	reset (nil) spool buffer
spool=close	closes spool disk file
filter *pr.iglf	ignores 'extra' line feeds
filter *pr.filter	adds 256 byte printer filter
filter *pr.find	translate printer byte to chng
filter *pr.lines	define number of lines per page
filter *pr.tmarg	adds top margin to printouts
filter *pr.page	number pages, set page number
filter *pr.tof	moves paper to top of form
filter *ki.echo	echo keys to the printer
attrib :d password	change master password
filedate=y.n	date boot up prompt on/off
cursor='xx'	define boot up cursor character
caps=y/n	set key caps boot up default
wp=d.y/n	write protect any or all drives
trace=y/n	turn sp monitor on/off
memory=y/n	basic free memory display monitor
fast	4 mghz speed (model 4)
basic2	enter rom basic (non-disk)
sysres=h/b/'xx'	move/sys overlay(s) to hi/bank mem
macro	define any key to macro
spool=d.size='xx'	link mem spooling to disk file
spool=y	reactivate disabled spooler
spool=open	opens, reactivates disk spooling
filter *pr.adlf=y/n	add line feed before printing0dh
filter *pr.hard=y/n	send 0ch to printer (fastest tof)
filter *pr.orig	translate printer byte to chng
filter *pr.reset	reset printer filter table
filter *pr.width	define printer line width
filter *pr.bmarg	adds bottom margin to printout
filter *pr.route	sets printer routing on/off
filter *pr.newpg	set dcb line count to 1
filter *pr.macro	turn macro keys on/off
device	displays current config

All parms above are installed using the new LIBRARY command SYSTEM (parm,parm). Other new LIB options include DBSIDE (enables double sided drive by treating the "other side" as a new independent drive, drives 0-7 supported) and SWAP (swap drive code table #s). Dump (CONFIG) all current high and/or bank memory data/routines and other current config to a disk data file. If your type ahead is active, you can (optional) store text in the type buffer, which is saved. During a boot, the config file is loaded back into high/bank memory and interrupts are recognized. After executing any active auto command, any stored type ahead data will be output. FANTASTIC! Convert your QWERTY keyboard to a DVORAK! Route printer output to the screen or your RS-232. Macro any key,even F1, F2 or F3. Load *01-*15 overlay(s) into high/bank memory for a memory only DOS! Enter data faster with the 256 byte type ahead option. Run 4MGHZ error free as clock, disk I/O routines are properly corrected! Spool printing to high/bank memory. Link spooling to disk (spooling updates DCB upon entering storage). Install up to 4 different debugging monitors. Print MS-DOS text files, ignoring those unwanted line feeds. Copy, Lprint, List or CATalog DOSPLUS, LS-DOS, LDOS or TRSDOS 6.x.x. files and disks. Add top/bottom margins and/or page numbers to your hard copy. Rename/Redate disks. Use special printer codes eg: LPRINT CHR$(3); toggles printer output to the ROUTE device. Special keyboard codes add even more versatility. This upgrade improves date file stamping MM/DD/YY instead of just MM/YY. Adds optional verify on/off formatting, enables users to examine *01-*15, DIR, and BOOT sectors using DEBUG, and corrects all known TRSDOS 1.3. DOS errors. Upgrade includes LIBDVR, a /CMD driver that enables LIBRARY commands, such as DIR, COPY, DEBUG, FREE, PURGE, or even small /CMD programs to be used within a running Basic program, without variable or data loss.

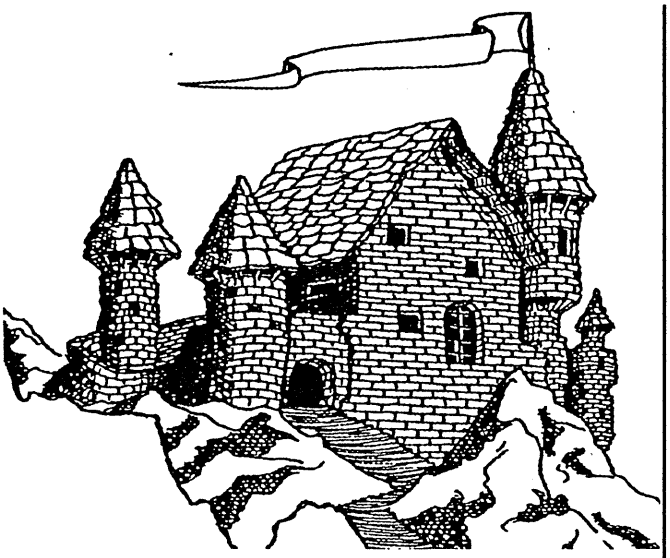
SYSTEM 1.5. is now distributed exclusively by TRSTimes magazine.
ORDER YOUR COPY TODAY!

NEW LOW PRICE $15.95

TRSTimes - SYSTEM 1.5.
5721 Topanga Canyon Blvd., Suite 4
Woodland Hills, CA. 91367

BEAT THE GAME

By Daniel Myers



VOODOO CASTLE

The Scott Adams Adventures

Ah, VooDoo Castle! Nothing like a little spooky stuff now and then! This time around, you're trying to remove a nasty curse from Count Christo (who is a good Count, unlike Count Dracula, whom you polished off in a previous adventure).

The game starts with you standing in the chapel, next to the Count's wearing a sapphire ring. Take the ring, then go West to the ballroom. Enter the fireplace and get the idol. Leave the fireplace by going South, then clean off the idol, which is your source of light.

Now go East to the chapel, then South, West, South, and East into the armory. Plenty of rusting old junk here! However, some of it is still useful, so pick up the shield and the sword. From there, return to the stairwell, then go East into the Kettle Room and then North to the room with the animal heads.

Pause (paws?) here for a moment, and move the heads (that's what you need the sword for). This will reveal a safe. You don't have the combination yet, but you will soon enough. For now, just drop the sword and the head, then continue onward by going Eastward until you come to the lab.

Ignore the Ju-Ju bag for now and get the chemicals. Don't worry if anything explodes, the shield will protect you. Once you have the chemicals, go West and drop the shield. Make your way back to the torture chamber, moving the kettle (when you come to it) along the way. When you get to the torture chamber, mix and then drink the chemicals.

ZAP! You are now small enough to fit through the tiny doorway. Go through the door, and pick up the saw. Look at the graves, and get the four-leaf clover. Return East, then go back to the chapel. From there, go East to the huge stone door with the sapphire set in it.

Sapphire? Hmm, does that remind you of anything? Yep, sure does! Wave the ring, and the door disappears to be replaced by a tunnel with a slippery chute heading down. Drop the ring, then slide down the chute into room below. Pick up the plaque, then enter the hole in the wall.

You now emerge into the Medium's room. She runs out when you enter, but she will return if you "Summon Medium", and will make helpful, if cryptic, remarks. Now look at the crystal ball, and you will be magically returned to the tunnel. Get the knife, and go back to the chapel. Drop the knife there, then go on to the stairwell.

Once at the stairwell, pick up the glass, then go East to the kettle room. Since you moved the kettle before, you will notice there is now a dark hole in the floor. Drop the idol, then enter the hole. Read the plaque, which has the combination of the safe written on it (the writing is tiny, so you needed the glass to see it. The plaque itself glows in the dark.).

Once you know the combination, drop the plaque and the glass. Now get the rabbit's foot. Even though you can't see it, it is there, and you can pick it up. Climb out of the hole and get the idol again, then head along North to the room with the safe.

Enter the combination by saying Turn (first number) and then Turn (second number). Open the safe and get the antique hammer that's inside. Now you're ready to tackle the fireplace again, so return to the ballroom fireplace. On the way, stop off at the chapel, and put the rabbit's foot on Count Christo.

Saw through the grating, to reveal a button. Push the button, and a giant fan sucks you further

up the chimney, to where a chimney sweep is stuck. Give him a good push, and he's free! In return for helping him, he gives you a piece of paper with a magic word written on it. Remember that word, because it's important.

Climb back down to the fireplace, and then once again, go to the stairwell. This time, go upstairs to the parlor, where the Ju-Ju statue is. Say the magic word from the paper, then go back down and head for the laboratory. Remember to pick up the shield before you go in!

In the lab, get the Ju-Ju bag, and make tracks for the tunnel with the chute. You can drop the shield along the way, since you won't be needing it any longer. Slide down the chute, then wave the bag. Voila! The little crack in the wall is now large enough for you to pass through.

Go crack, and get the torn page. Now, open the bag, and get the book. Read both the book and the page, and make careful note of the ritual needed for freeing Count Cristo from the curse. Drop the book and the page, and get the stick. Go South back to the chute room, then enter the hole. Look once again into the crystal ball.

So, here you are in the tunnel once more. Hang in there, you're almost done! Go West, then North to the room with the window. Go window, and get the little VooDoo doll on the ledge. Go back South into the room, then return to the chapel. You now have everything you need to revoke the curse.

Following the instructions from the magic book, complete the ritual, freeing the Count. TA-DA! You have successfully finished the adventure! .

GOLDEN VOYAGE
The Scott Adams Adventures

Feeling in the mood for a little sea voyage? I hope so, because in this adventure you'll be spending a lot of time sailing in the ocean! Not to mention doing a lot of going back and forth! So, stretch your sea legs, and we'll be off on a Golden Voyage.

Here you are in a Persian city, standing in front of a merchant's stall. Don't bother trying to get the sandals, you don't have any money...yet! You have to pay a visit to the king, first. So, head along West, skip that merchant as well, and continue on North. Ah, there's the palace! Go inside, and the king will tell you what he wants done. He will also give you a bag of gold, so you can purchase everything you need.

Take the gold, and go East, South, East. You're back at the merchant again. Buy the sandals, and take them. Remember, in this game, buying and taking are two different things. You have to take each item you purchase. Ok, now go West and buy everyhing on sale here, and go West one more time. There's the ship! Ain't it a beauty? Of course, you have to buy that, as well!

Now, go on board. You can drop the gold and the compass here; you don't need to carry them around any more. Raise anchor, and sail West. Hmmmm, doesn't seem to be anything around, so climb the mast, and from there look through the telescope. Land Ho! You're just off a small island. Climb down again, drop the anchor, and wear the sandals.

Note: Always make sure you drop the anchor before leaving the boat, or the boat will drift away!!).

Once on the island, hike North. Here you see a hut and a mountain. You can ignore the hut. Go to the mountain, where you will find a sword. Get that, go down the mountain and South and West. Pick up the shovel, then return to the ship. Raise anchor, and sail East, which brings you back to the city, and then sail East twice more and North once. Climb up the mast again, and peek through the telescope. There's a sandy beach!

Come back down, and drop the sword and telescope, as well as the anchor. Go to the beach. Ignore the man. Now go into the jungle, and North from there. You are in front of a cave. However, you can't go in there yet, since you don't have a light source. In the meantime, dig twice on the spot where you're standing. The first time, you'll find a torch, the second time a stone.

OK, back to the ship. Drop the stone and the shovel, raise anchor (at least you're developing your arm muscles in this adventure!), and sail East twice, which takes you back to the city, and South one time. Get the telescope. Once again, climb the mast and look through the telescope. Now you're at a rocky strand. Climb down again, and now you can drop the telescope, it's served its purpose in the game.

Get the sword and drop anchor, then go to the strand. Interesting little statue, isn't it? Hmmmm, maybe you should take another look at it? Ooops! The statue just came to life, and it doesn't like you very much! Hotfoot it to the stairs, with the goddess in pursuit. Don't worry if she swings her sword, you will be protected by your own sword. Now, push the

statue down the stairs. Crash! So much for the stone goddess.

Drop your sword, and get the flint and steel, then walk up the rest of the way. At the top, go West. Here you are at an altar. This is a good place to pray, so do that, and a secret passage appears! Light your torch, go into the passage, then West. Here is another little stone. Get that and go back to the altar. Unlight your torch (very important!), then go to the stairs and walk down.

At the bottom is a pile of rubble, all that remains of the statue. Look in there, and by golly, you just found another small stone! Take that, and go back to the ship.

All right, time to be moving on again. But first, drop the two small stones, which will mysteriously unite to form a second tablet. Leave it for now, raise anchor, and sail North then East twice. Ah yes, that sandy beach! Drop anchor, get the shovel, and go to the beach.

Make your way to the jungle again, go North to the cave entrance, and light your torch. Now you can go into the cave, where you see a strange fountain. If you've looked at the tablet you bought in the city, you'll have seen it has a picture of a cave on it. Well, this is the place! Put the tablet in the fountain, then go West out of the cave.

Unlight the torch, and go South. Dig here, and you will find a rope. Take that, and return to the ship. Once on board, raise anchor, and sail West 3 times. Now your're back at the small island again (I told you there was a lot of going back and forth here!). Drop anchor. Get the second tablet, and go to the island, then North to where you can see the mountain.

If you look at the second tablet, it has a picture of the mountain, and also a word. Say that word now, and Rumble! a crevice appears! Light your torch again, and go into the crevice. Well, looky there..another fountain! Drop your tablet into that one, but don't run off yet! After the ground stops shaking, look in the fountain. Aha! A mysterious globe! Get that, and go out the crevice, unlight the torch (handy, isn't it?), and return to your ship.

Don't relax just yet, there's still much to be done! Raise anchor, and sail East, then South. Here you are at the strand again. Drop anchor, go to the strand, then South. Dig here, and you will find a small key. Drop the shovel, and pick up the key. Hang in there, the end is in sight!

Now you're ready for the final part of the adventure. Go back to the stairs, walk up, and go to the altar. This time, go North, and you'll see a fancy chest. Unlock the chest, drop the key, and look inside. Wow! A gold mask! Take that, and go back to the altar.

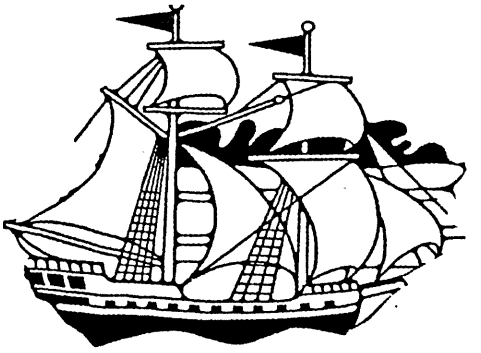
Here you can drop your sandals. Now, look at the altar, and you'll see a gold chalice! (No wonder this is called "Golden Voyage"!). Get the chalice, light your torch, and go into the secret passage. Oho! The stone block is no more, and there is a hallway here now (so that's what all the fuss was about back at the second fountain!).

Enter the hallway, and go East. You'll see a pit and a stalagmite. I wouldn't advise jumping down in there; better tie the rope to the stalagmite, then drop the rope into the pit. Now you can climb down safely. When you reach bottom, wear the mask. Everything looks a little strange, doesn't it? Don't worry, the use of the mask will soon become apparent.

Go South. Uh oh! There's a nasty-looking Cyclops here, and he doesn't like you any more than the statue did! Better do something quick! How about dropping the globe? Boom! There's an explosion of light! Good thing you had the mask on, or you wuldn't be able to see a thing!

Ok, slip past the dazzled Cyclops into the cave. Go West, and here you see a beautiful fountain. At last! This is the one you want! Fill your chalice, remove the mask, and go back East, North, North. Climb back up the rope, and return to the altar room. Unlight your torch.

Pick up your sandals (gotta watch out for those scorpions!), put them on, and go back down the stairs to your ship. Raise anchor yet again, and sail North. You've reurned to the Persian city. Drop anchor, and leave your ship for the city. Go East, then North, and you're back at the palace. Go in, and give the chalice to the king. He drinks the contents, and is magically transformed into a young man! Congratulations! You've accomplished your mission!



THE GOOD OL' DAYS

HARD DISK HARD TIMES

by Dr. John T. Phillipp

Let me say right at the outset - the hard disk drive wasn't really my idea. Well, it was, but it wasn't. I didn't need a hard disk - I didn't even WANT one. But my brother-in-law needed one, so I got involved.

My brother-in-law uses the same computer in his practice that I use in mine - a Model 4 (name of Maxine) with 2 double-sided 40- and 2 double-sided 80 track drives. However, with 5,000 names in his names-file, and about 8,000 ledger entries in his ledger- file, the 80track drive was getting pretty full. I could have split the files, putting the names-file on one of the 80 track drives, and the ledger-file on the other, but it seemed easier to bite the bullet and get a hard drive, since I figured we would need one sooner or later anyway. I should have stood in bed the day THAT idea struck.

Actually, I had been thinking about a hard disk for the other office... I even went so far as to order NEWDOS80 version 2.5 - the version that works with hard disks. Very interesting - I had called Apparat, and found out that version 2.5 only cost $65, but they wouldn't sell it to anyone that didn't have a registered copy of version 2.0!! No bootleg operating systems allowed!!

Anyway, with NEWDOS80 version 2.5 in hand, all I needed was a hard disk to use it on.

I kept putting it off, and putting it off, but I finallv ordered a hard disk from Software Support, Inc. after I saw their ads in 80 Micro magazine. Their prlces seemed to be about the best - not the $400 an IBM hard disk costs those days - but a reasonable TRS-80 $700. And they seemed to be around, month after month, so I figured they would be around to support me if I had any problems.

"Hello, Software Support. I want to order a hard disk for a TRS-80 Model 4."

"Fine. What operating system?"

"NEWDOS80 2.5"

"No problem. We have one here that's running under NEWDOS. You'll just be able to plug it in and run it."

Ha!

I ordered a 10 megabyte drive. After all, a double-sided 80 track drive holds about .75 megabytes, so a 10 meg hard disk is the equivalent of 13 floppies. It sounded like enough storage. About a week later, the drive arrived - the drive, cable, a disk labeled "INSTAPATCH Model III NEWDOS80 2.5 Omti 20L Controller", and 10 pages of documentation. It seems that NEWDOS doesn't recognize the type of controller they use, so it has to be patched.

I had to admit that the documentation seemed pretty complete (yes, I read the documentation BEFORE trying to install the hard drive. You only make an ass of yourself when you THINK you know what you're doing. I cheerfully confessed ignorance. The only hard drive I had any experience with was the one in my IBM clone, and George Fisher came over and set that one up for me). In fact, one page had directions on installing the drive with four partitions without formatting or anything. All you had to do was put the INSTAPATCH disk in drive 1, a copy of the NEWDOS 2.5 in drive 0, plug in the hard drive, turn it on, type "DO CONFIG", and press <ENTER). As they said in the docs "When the task has been completed, your MEGADISK will be fully operational." Simple.

Simple all right. It simply didn't work.

Well, I hit <ENTER) and all kinds of things happened. The disk drives whirred, the hard disk red light came on, neat messages flashed on the screen (like erasing BASIC from drive 0, copying BASIC from drive 1 to drive 0, then erasing BASIC on drive 0 again). After about 2 minutes came the last message: "ERROR". And everything froze up. I tried the whole procedure again, and got the same result.

So much for the easy way.

All was not lost, though. The documentation listed a more complicated way to prepare the disk for use. The alternate way allowed dividing the hard drive into any number of volumes, rather than 4, like the easy way. So I tried it.

First I had to answer a lot of questions about the drive: number of surfaces, number of cylinders, step rate and so on. To my relief, all those parameters were listed on the second page of the documentation. Then I had to run a batch file based on the number of megabytes, number of cylinders, and number of volumes you wanted. Then, as the docs

said, 'When prompted, please press <ENTER> to format each volume. This program will format all drives, copy information onto the hard drive, turn the diskette in drive 0 into a boot-to-hard-drive diskette, and once booted, will turn control over to the hard drive."

OK. I typed DO B306106 and stood back.

Everything seemed to work fine. Disks whirred, messages flashed. There was one that said "Formatting hard drive, this will take 2 minutes", and it formatted the hard drive in 2 minutes, and then said "DONE". Finally it got to the NEWDOS80 FORMAT routine and said "Press <ENTER> when destination disk mounted in drive 2".

OK, I pressed <ENTER>. The red hard disk light came on and the asterisks began to flash in the upper right corner of the screen.

And they flashed. And they flashed. And they flashed.

Sometimes they flashed very fast, almost a blur. Sometimes they flashed very slowly. Sometimes they stopped, and I stared at one frozen asterisk for 15 or 20 seconds. Then they'd start again. After an hour of this I began to get concerned. I know that hard disks are BIG, but an hour to format one-sixth of one?

So I called the tech support number given in the documentation (Monday to Friday 9:00 to 12:00 and 1:00 to 5:00 EST) for a little tech support.

The gentleman on the phone was very polite. It was his opinion that as long as the asterisks were flashing, that meant that everything was proceeding normally. I told him that I thought it was taking an awful long time. He said that hard disks are big, and they take a long time to format.

"By the way" he said, "how long has it been formatting?"

"About 2 hours", I said.

"It's broken", he said. "Copy this return authorization number and send it back to us."

And that's what I did.

Well, I kind of forgot about it then... it was being used in my brother-in-law's office, remember, so I didn't really NEED it. Anyway, after about three weeks or so the drive arrived back in the mail. I set it up, and tried to run the "CONFIG" DO file. No luck.

In fact, the drive was dead. There are two lights on the hard drive... a red one that comes on when the drive is being accessed, and a green one that comes on when the power is applied. This time, even the green one wasn't lit. I thought that was funny, because I could hear the fan working inside the drive. I unplugged it and checked the outlet. I wiggled the power cable. Nothing. No green light.

On the phone again.

"I just got my drive back from having it repaired, and it's dead"

"What's the problem?"

"The green light doesn't go on."

"Oh, you must be missing the jumper!"

"Huh?"

It turns out that the power supply will work with 110 volts or 220 volts, jumper selected. If there is no jumper in place, then a relay is supposed to close and configure the system for 110 volts automaticall1y. But, they explained, in some of the drives the relay didn't work, and the jumper had to be physically in place. Sadly, they had neglected to put jumpers on many of the drives, so a lot of them were being returned because they wouldn't work.

I mentioned that the drive had just been returned to me after having been sent back for repairs, and that I didn't think very much of their quality control. I wondered how they could test the drive and repair it if it wouldn't work without the jumper, and no one had noticed the jumper wasn't there!

"By the way", I asked, "what repairs were made on the drive, anyway."

"A report of the repairs done was packed with the drive."

I looked through the packing case again.

"No report in there", I said.

"Hmmm, maybe they checked it out, and didn't find anything wrong."

This did NOT sound encouraging.

"Can you put in the jumper yourself?", he asked.

"Why not," I said. "It's better than sending the drive back for repairs again.

He explained that there was a three prong connector on the circuit board. The left prong was labeled 110, the right one was labeled 220, and the middle one wasn't labeled at all. All I had to do was connect a jumper between the left prong and the center one. So I hung up, took the case off the drive, located the connector, and put on the jumper.

Success. The green light came on.

I tried the "CONFIG" DO file again. The screen gave messages, the drive whirred, the red light

went on and off, and then the message came up... DISK ERROR. Just like it had before the drive was "repaired".

I tried the "long form" configuration, but that didn't work either. Back on the phone.

"It doesn't work."
"It worked when it left here."
"It doesn't work"

Well we discussed the matter for five or ten minutes, and then inspiration struck the technician.

"Have you" he asked, "purged ALL the files off your NEWDOS80 2.5 disk?"

"No" I replied. "Should I?"

"The configuration DO files won't work if you don't"

I should explain that the NEWDOS80 version 2.5 system disk comes with all the usual NEWDOS80 utilities on the disk - SUPERZAP, DIRCHECK, LMOFFSET, EDTASM, and the rest. I looked through the megadisk documentation again (when all else fails, read the documentation). Sure enough, it instructed me to purge all the visible files off the copy of the NEWDOS80 2.5 system disk I was using.

I purged.
I ran the "long form" configuration.
It worked.

Well, sort of. We got further than we ever had before. I was partitioning the hard disk into 6 volumes. We got the message "Press <ENTER) when destination disk on drive 5". We pressed <ENTER>. It said "Formatting...", and really did. After a while it said "Verifying..." and did that, too. After a while it said "Done... Press <ENTER> when destination disk on drive 6".

I pressed <ENTER> with a feeling of exhilaration. It worked. I was actually getting the hard disk formatted. I was really going to have volumes with a thousand free grans on each. I was going to become a HARD DISK USER and never have to worry about disk space again!! I mentally took back all the nasty things I had said about Software Support, Inc.

Meanwhile, the screen said "Verifying..." and then stopped. "Error while verifying destination sector 1395. <S>kip <R>etry <Q>uit". Oh no!! I mentally said all the nasty things I had taken back all over again. Why do these things always happen to me!!??!!

I pressed <R>, but the error persisted. Eventually I pressed <S). The asterisks flashed a few times and then "Error while verfifying destination sector 1420..." (The facts I am presenting are true... the sector numbers have been changed to protect the innocent. Dummm dum dummm dummmm.) And it went on like that for the remaining 5 volumes. The volumes formatted, but there were errors verifying many of the sectors, many, many of the sectors. Retrying didn't help, and I had to skip them. I hoped the bad sectors would be LOCKED OUT, but I doubted it, since NEWDOS80 doesn't LOCK OUT sectors on floppys - they feel that disks are so cheap, that if some sectors are bad, you should discard the disk. So I figured they'd do the same for a hard disk. Actually, at that point, discarding the hard disk didn't seem to be such an unreasonable idea!!

After getting all the volumes formatted, even though they seemed to have all those bad sectors, the configuration program finished. I was left with a floppy disk in drive 0 that had a copy of NEWDOS80 2.5 on it, and that copy had been modified so that I could boot with that disk, and it would transfer control to the hard disk. After booting, drive 0 would be the hard disk, and I wouldn' t need a svstem disk in drive 0. Great idea. Sadlv, it didn't work.

Because of all the bad sectors on the hard disk, it was impossible to copy any of the system or program files from the floppies to the hard disk. The copy process kept aborting in the middle with "Sector Not Found" and other error messages. Finally I gave up, called Software Support, got another return authorization number, and sent the drive back for repairs... again. I included a letter telling them that this was the second time the drive had been returned, that I was sure that there was something wrong with the platters in the drive, and that I would appreciate it if they would send me a new drive, rather than the same one back again.

I waited another two or three weeks, then finally the drive came back. There was a note attached to the drive. "We are sorry for all the inconvenience you have had. We have replaced your 10 megabyte drive with a 15 megabyte unit at no charge.

Now that's software support.

Anyway, I can finish this story quickly (it has gone rather on and on... and on, and on and on...). The 15 megabyte drive formatted flawlessly, divided into 6 volumes. The system disk in drive 0 was converted into a boot disk, and after booting, the volumes were numbered 0 to 6. All files transferred perfectly.

Interestingly, COPIES of the boot disk work differently. The copies, when booted, DON'T make the

hard disk drive 0. Drive 0 remains the drive 0 floppy, and drive 1 remains the drive 1 floppy. Floppy drives 2 and 3 are ignored, and the hard disk volumes are numbered 2 to 7. Why? I haven't the faintest idea. But I decided to stop messing with things I don't understand. I'm using the (floppy) system disk in drive 0, my program files are on (hard disk) drive 2 (with all the files on that volume, I still have 900 free grans!!), and my data files are on (hard disk) drive 3. My brother-in-law will never have enough patients to fill a 15 megabyte hard drive.

The system is still working perfectly, and all concerned are very happy!!

KELLY'S CORNER

by Kelly Bates

Opened another can of worms today! ModelA/III is used only on a Model 4P, I believe, to get the 4P into Model III mode. I have found six different versions, based on the date stamp, and MULTIDOS may have yet another version.

The dates on the different versions I found are 1 Mar 80 (on MULTIDOS), 15 Mar 84 (Cat 26-0316 put out by RS), 15 Mar 82 (on most of my boot disks), 15 Sep 83 (found on LDOS 5.1.4), no date (TRSDOS 1.3), 30 Jan 84 (another Cat 26-0313 by RS), and 03/83 on SUPERDOS by Sandy Bair. All these versions of MODELA/III seem to work OK.

Found one other that will not work on my 4P. I get weird errors, such as "no disk in drive :0:" and "command too long". The last one was displayed when I tried to run DBTA/CMD, a directory menu used on NEWDOS/80 v2.0. The date stamp is 1 Dec 86 and comes from LSI. Using a disk zapper on one of the sectors of the file I found "Copr 1986 fdiv / distribution by LSI". Most likely someone zapped this copy before passing it along to me (or perhaps my copying it garbled it), as LSI usually put out good stuff. The boot ROM date on my 4P is 18 Oct 83.

Now the worm — some programs run just fine and no errors occur. However, if you've run into this, I suggest you use the 82 or 84 version — or drop me a line and I'll send you a copy of the one I use. By the way, the worm is not a problem for me as I am aware of it, but I just wanted to let you know. Somebody else using it may think the computer went bad.

A note on on MODELA/III. Holding F3 and P until MODELA/III starts will get you the prompt "change disks and press ENTER", whereas pressing F3 by itself does not give you a 'pause' to change disks.

Still working on getting that 4P backup system ready to go to Chicago so my sister can take it back with her when she comes later this year. Things are going OK. One more thing I want to ship is a complete set of the Prosoft fonts (on six 720k floppies). Another pair of floppies completed is the ALL-WRITE, ELECTRIC WEBSTER, LETTERSET MANIPULATION UTILITIES manuals. Wish I had the DOS manuals on floppies. Got the Gemini 10X printer all checked out, it now works like a champ. I may or may not check out the DMP120 since it is a 7 pin printer.

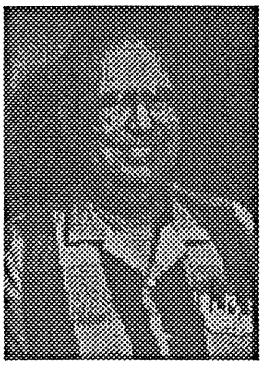
OK, now have the 4P ready to use external floppy drives. The last part was the wiring of the 4 wires on the CPU. Delay caused by lack of a small tip on my soldering iron (did a bit of trim with a file). So now I have two 4Ps that have external floppy interfaces. This last 4P has the 3 1/2" 720k internal floppies that would not let Hypercross work properly. So this morning I fired up Hypercross after hooking up two 360k external floppies. I designated drive :2 as the Alien. It copied a file to :0 flawlessly. So, if other users of Hypercross experience the problem I had, just designate a 360k drive as the Alien. At some point in time I will exchange the two 720k floppies for two that will let Hypercross operate on :0 and :1 (the normal defaults).

I had intended to send my old 4P to Chicago, but the new one has the cluster of arrow keys to the right of the caps lock key, and my older 4P has the arrows split — Up and Down on the left of the keyboard, and Left/Right near the Enter key. So I am rethinking which one to ship. Unlike the old 4P, the new one does not yet have a joystick interface, so I will probably still send the old one.

Now you can see how my logic progresses. The Hypercross problem had existed since I got the second 4P and upgraded it to have 720k floppies. I had to get the external floppy mod completed before I could pursue the problem further. Over the years I have discovered that a problem should be thought about a bit before hitting the 'panic' button. Interim solutions I call 'work around' until I get to a point where I can 'make it right'. Without two proper 720k floppies internally, I am using the 'work around' method. And Hypercross at least has the programming to let me use a 'work around' solution until later on.

Are You Violating a Copyright?

by Roy T. Beck



Introduction

Recently, a caller on my bulletin board (TRSurtrove BBS, 213-664-5056, 300-2400 baud, 8-N-1) asked a rhetorical question, to the effect where are all the old TRS programs, because after all, all the copyrights expired 6 years or so ago?

I knew he was wrong in saying all the copyrights expired 6 years ago, but while I remonstrated with him, I also realized I did not know the precise rules which **are** in effect. Therefore, I went to the library to do a little research. I found two books on point, both by a lawyer named Stephen Fishman. They are:

The Copyright Handbook, 2nd Ed., Copyright 1995, Nolo Press
Copyright Your Software, Nolo Press

Written Texts

A little background. In the years BC, (before computers), the life of a copyright, which then referred to a published book or other written material was limited by only a few simple rules. If an individual wrote a book for his own benefit, copyrighted it, and published it, then the copyright endured until 50 years after the author's death. Very simple and neat. Of course, very often the copyright was granted in the name of the publisher, but the same rule applied. However, if an employee was hired to write material for his employer, for example articles to fill an encyclopedia, then the copyright was good for 75 years after first publication, or 100 years from creation, whichever limit is reached first.

I should be explicit that the above rules applied through 1977. After that date, new rules came into effect beginning in 1978, hence my joke about BC. These new rules were in effect through 1988, when more changes were made starting in 1989 and continuing until Congress changes the rules again.

The new rules effective beginning 1978 made some retroactive changes. A work both created and published before 1978 was granted a 75 year term dating from date of publication. A work created before 1978 but not published before 1978 was given an arbitrary expiration date of December 31, 2002. I have no idea why such an arbitrary date was imposed.

I did not study the changes made effective 1989, which I think were detail changes.

Software

Prior to 1978, there was no provision in the Copyright law for computer programs, which therefore had no copyright protection. The need for such protection came into effect for programs stored on tapes, cards, disks, etc. In addition, a new form had recently come into existence, that being programs written into ROMs, which were neither strictly software (programs) nor hardware. Hardware had some protection under patent laws, which is yet another matter. Great arguments were presented by great lawyers to great judges on both sides of the issues of copyrighting software in various forms.

At the same time the copyright law covering written texts was altered in 1978, copyright protection was created for software. Where software was created by an individual for his own benefit and was copyrighted and published, the copyright had an expiration date 50 years after that of the author. If the copyright was created by two or more authors in a group effort for their mutual benefit, then the copyright ran until 50 years after the death of the last member of the group. LS-DOS was such a committee product.

Where software was created by employee(s) working for hire, then the copyright was good for 75 years after publication or 100 years after the date of creation, whichever comes first. This latter class of creators included both consultants and anonymous authors. I think MS-DOS falls into this class.

This same revision of the law made retroactive provision for software created before 1978. If a program was created after 1964 and before 1978, then its copyright life is 75 years after date of publication. Programs written before 1965 were entitled to copyright protection for 28 years plus 47 additional years

if renewal was applied for during the 28th year. This in effect allowed copyrights for software written as early as 1950.

Backup Copies

The law presently allows the buyer of a copyrighted program to make only one backup copy. This backup copy cannot be used on a second machine. Most software publishers will enter into an agreement for a user to make and use multiple copies on multiple machines under his control, but this is strictly only by an agreement, commonly known as a "site license". This is not a matter of right under the copyright law. The law only allows one copy as backup, and that cannot be legally used on a second machine.

Some Common Misconceptions

Many people think a written work must be "registered" with the US Copyright Office to be protected. This has not been true since the 1978 revisions of copyright law. Protection begins automatically the moment a work is set to paper or otherwise fixed in a tangible form. (I think this includes word processors).

Another belief is that protection is afforded only if a Copyright Notice is in it. This requirement was removed effective March 1, 1989.

Another belief is that no one can legally use or photocopy a protected work without the copyright owner's permission. This was never true. Anyone can photocopy or otherwise use protected works so long as the use comes under the "fair use" rule. This does not diminish the value of the protected word.

Can you copyright your great ideas? No, copyright only protects the expression of an idea, not the idea itself. Great ideas are the starting points for all kinds of things and developments, but you must apply the idea to arrive at something which can be patented or copyrighted.

In Conclusion

Copyrights are a powerful and, in general, enduring form of protection for the authors and vendors of software. Just because a vendor has gone out of business, you cannot steal and use his copyrighted programs with impunity. True, as a practical matter, he may not have the inclination (or wherewithal) to pursue a lawsuit against you because you violate his copyright, but that does not, of itself, excuse you from obeying the law. You may have noticed that Stan Slater of Computer News 80 has gone to great lengths to find and make agreements with the copyright owners of outstanding software which has gone out of publication. In this way he has restored to availability some good software, but has had to find the copyright owners and make legal agreements with them before he can sell their software.

In some cases, authors have declared their copyrighted software to be "public domain", meaning they now allow anyone to make any desired use of the software with little or no restrictions. This is a voluntary action by the author, and does not occur until and unless he opts to do so. Just because you find bootleg copies of copyrighted programs somewhere, that does not mean you are entitled to use them. Legally, they are still restricted by copyright, and you use them at your own risk. The only certainty is when **you** posess the **original** disk.