Znews

Vol. 1 No. 3, May 1989

Playing CatchUp, or *"Please Pass the Heinz 57"*

Boy, did I ever ignite a firestorm in our first issue (at least for a few people). Seems that y'all got the first **Znews** *after* my extended introductory offer expired. I won't belabor the reasons for that occurrence except to say that if you have received *this* month's newsletter in May, we have rectified the problem. Or started to.

But holy guacamole, Batman, did some of you take great exception to the snafu or what? I guess a lot of people assumed that we were out to gouge you in every way possible. Let me state again, for the record, that I guarantee your satisfaction. A quick postcard or call will fix just about anything. Please do not assume the worst of us.

Here's my plan to remedy the present situation: there will no longer be a time limit on the introductory offer if you were a subscriber to the old *Z* newsletter. If you *already* resubscribed with us at the higher price, we will refund the difference at your request.

Yes, that cost us a considerable amount of money. But we're a sole proprietorship with no need to answer to shareholders - we just answer to ourselves, and ultimately, you, our subscribers. We do what we want, when we want. And we want to convince you of our good intentions, fairness, goodwill, and general niceness.

'Nuff said, I hope.

Speaking of good intentions, if you have not yet subscribed to **Znews** and you intend to, now would be a good time. I have included an order form on page 14 to help expedite the process.

The reason that I mention subscriptions at all is that most folks signed on with Zedcor for one year (four issues). Since the Z-gang delivered one newsletter in 1988, (no, I don't count the announcement of our assumption of the newsletter in January), that means we owe the one year folks three issues. This is the third issue and completes our obligations to most of you.

If you have not received three issues of **Znews** (March, April, and May), please let us know and we will remedy the situation.

It is also apparent that a surprisingly large number of you folks were not aware that we publish *two* ZBasic newsletters: **Znews** covers the MS-DOS, Apple II, and Z-80 worlds, while **McZ** details all things Macintosh. The Mac version is sufficiently different from the others (due to the infamous toolboxes and the Mac interface), that we felt everybody would get better coverage if we split things up.

It is a little premature to say too much at this point, but we have some plans I'd like to tell you about. We here at the Ariel igloo maintain an active presence on several online databases, including GEnie, AppleLink PE, and a private bulletin board being setup in Florida even as I write. We hope to be the clearinghouse for ZBasic information worldwide. We've also initiated some other projects, but I'm not at liberty to discuss them at this time. My point: we're working to make your subscription to **Znews** a valuable commodity, indeed.

I'm also interested in finding out about commercial packages written in ZBasic. I've run across several - I think you'd be surprised at the terrific things ZBasic users are able to do - in part because of the capabilities and usefulness of the language.

Quality Computers, for example, is publishing Professor Gary Morrion's **RepairWorks**, a program dedicated to repairing and restoring damaged AppleWorks files. Subscriber David Marans also wrote some greyhound racing analysis software with MS-DOS ZBasic that was even reported as profitable by the sports section of his local newspaper. His program inspired this month's foray into expert systems elsewhere in this issue.

Tell me more, folks. I don't intend to review *any* of the programs here. Rather, my intention is to compile a list of commercially available software packages written in any version of ZBasic.

MS-DOS Alert

I don't know if Apple II programmer's submit articles as a way of life or what, but in order to provide an appropriate balance, I need to request that some of you MS-DOS and Z-80 fans get crackin'. We pay between $50-$75 when we accept an article, so in addition to the prestige (?) you can also get a little spare change for your hardware fund.

We're Moving...

Yes, Ariel Publishing is moving "Outside", as we say in Alaska. Our new address will be:

Ariel Publishing
P.O. Box 398
Pateros, WA 98846

phone: (509) 923-2025

This address and phone number will be good after June 10th, 1989. We will be unavailable from May 29th - June 9th. I hope that doesn't inconvenience anyone, but it takes time to move.

We've got a lot of personal reasons for moving, but one major factor is that it will make our publishing business much, much easier to operate. You cannot imagine the headaches and hassles we've endured out here on the Bering Strait.

In spite of our move south, I'm sticking to my roots; no big towns for this country boy. Pateros is a tiny community of about 300 people that sits at the mouth of the beautiful Methow River in Central Washington. The Methow Valley is famous for apple's (the fruit, that is, and mostly organically grown) and trout. The craggy central Cascade Range provides a gorgeous backdrop to any picture.

Enough trivia. Let's get on with the show, as they say.

Znews

Subscription prices in US dollars *(Canadian and Mexican subscribers add $5 per year, all other non-North American subscribers add $15 per year)*:

1 year... $35 2 years $...65

A quarterly diskette is available for an additional $20 per year *(non-North American subscribers add $10 per year)*

WARRANTY and LIMITATION of LIABILITY

I warrant that the information in **Znews** is correct and somewhat useful to somebody somewhere. Any subscriber may ask for a full refund of their last subscription payment at any time. MY LIABILITY FOR ERRORS AND OMISSIONS IS LIMITED TO THIS PUBLICATION'S PURCHASE PRICE. In no case shall I or my contributors be liable for any incidental or consequential damages, nor for ANY damages in excess of the fees paid by a subscriber.

Please direct all correspondence to:

Ariel Publishing
P.O. Box 266
Unalakleet, AK 99684

Znews is a product of the United States of America

The ProDOS MLI: An Introduction

by Gary Morrison

Have you ever wondered how to create a subdirectory, a new AppleWorks word processing file, lock or unlock a file, or how to use ZBASIC's nifty ON LINE command within your ZBASIC program? There is a rather simple way to greatly expand disk and system access capabilities of the Apple II ProDOS ZBASIC. In fact, it's as simple as using a template.

One method of expanding the capabilities of ZBASIC is through the Machine Language Interface or MLI. Now, do not run away scared because of the term machine language. What I know about machine or assembly language will fit between the period at the end of this sentence and the first letter of the next sentence. Apple first provided assembly language programmers with this nifty tool, and then ZBASIC provided BASIC programmers with an easy access to this interface. In fact, if you know how to use the POKE and PEEK commands you are ready to use the MLI. Let's take a closer look at the Machine Language Interface.

The MLI

Why would anyone want to use the MLI interface when many of the essential MLI calls are easily done from AppleSoft BASIC when BASIC.SYSTEM is operating? For example, in Applesoft we can create an AppleWorks file with the line,

```
PRINT CHR$(4)"CREATE MYWORD, TAWP"
```

(editor: this creates a file of the same filetype as AWP files, but the internal structure is not necessarily the same.)

An assembly language programmer, however, might want to write a program that does not require BASIC.SYSTEM (e.g., APPLWRKS.SYSTEM). Thus, Apple provided the Machine Language Interface so assembly language programmers would not have to write large amounts of code just to OPEN and READ a file. Fortunately for us, the MLI is not limited to just assembly lanugage programmers.

In the ProDOS Appendix of your ZBASIC manual, you will find a one page description of the MLI interface with some important memory locations. ZBASIC provides us with an 18 byte "template" starting at memory location $1F00, the machine language code we need to call the MLI routine, and an error handler if we want ZBASIC to display the error message. All we need to provide is the information for the template. If you are going to do some serious programming with ProDOS, I would encourage you to purchase a copy of Apple's ProDOS Technical Reference Manual. Make sure you get the one for ProDOS 8 as the ProDOS 16 and GS/OS manuals will not be helpful with ZBASIC. Also, I will be using hex numbers (with a $) to maintain consistency with published information on the MLI calls.

MLI Calls

There are 25 separate MLI calls which Apple divides into four groups. First, there are the housekeeping calls such as CREATE and DESTROY (delete). Second are filing calls like OPEN, FLUSH, CLOSE, and READ. The third group consists of system calls for getting the time and managing interrupts.

Fourth are direct disk access commands like READ_BLOCK and WRITE_BLOCK. We can also divide the group into those calls that use the ZBASIC MLI template and those that require us to provide a larger buffer. We will start with the calls that use the ZBASIC MLI template.

ZBASIC's MLI Template

There are three parts to the MLI template. First is the number of items to be passed back and forth through the template. This number is referred to as the parameter count (param_count). Second is information you provide for MLI interface to use in the call. This information could be a number, a pathname, a filename, or memory location. Third is information returned from the MLI call which is referred to as a result. In our ZBASIC template and in all MLI calls, the first item is the parameter count. Parameter counts for each of the calls are supplied in the ProDOS Technical Reference Manual. In the ProDOS calls, the information supplied by you comes next. The results come after your input. Let's use the MLI to write a program that provides information about a file and allows us to change that information.

(editor: WARNING! Changing file attributes can render any given file useless. Changing the filetype of an AppleWorks file, for example, will make it "invisible" to AppleWorks. Test SET_FILE_INFO on files you don't care about!)

File Info Program

We will be using two MLI calls, SET_FILE_INFO and GET_FILE_INFO, for our first program. The param_count for SET_FILE_INFO is 7 and GET_FILE_INFO is $A (10 in base 10).

If we use the GET_FILE_INFO first, we can then use the results from it as the input for the SET_FILE_INFO as both of these calls use a similar template. Table 1 shows the template for these two calls.

As we can see from Table 1, the template for the two calls is just about the same except for the 7th ,8th, and 9th bytes of SET_FILE_INFO. ProDOS is telling us that we cannot change the storage_type or blocks_used with the SET_FILE_INFO command. Nor can we change the creation date and time with this command as they are controlled by ProDOS.

Table 1
SET_FILE_INFO and GET_FILE_INFO Templates

$1F00+ SET_FILE_INFO	$1F00+ GET_FILE_INFO
0 param_count= 7	0 param_count=10
1 pointer to pathname	1 pointer to pathname
(two byte pointer)	(two byte pointer)
3 acceess	3 access
4 file_type	4 file_type
5 aux_type	5 aux_type
(two byte pointer)	(two byte pointer)
7 null field	7 storage_type
8 null field	8 blocks used
9 null field	(two byte pointer)
10 mod_date	10 mod_date
(two byte pointer)	(two byte pointer)
12 mod_time	12 mod_time
(two byte pointer)	(two byte pointer)
	14 create_date
	(two byte pointer)
	16 create_time
	(two byte pointer)

When we use the GET_FILE_INFO call, we must provide three bytes of information. The first is the parameter count, $A, and a two byte pointer to the pathname. Ooops, there is a new term--pointer. Pointers are memory addresses that have special significance. For example a pointer to the hi-res screen would contain the value $2000 which is the start of the hi-res screen. MLI calls often require a pointer to tell it where a variable such as a pathname is stored or where the beginning of a buffer is located. We will talk about pointers in more detail in a moment. The remainder of the "template" will be filled with information that is the result of the MLI call. The SET_FILE_INFO call requires us to provide all of the information for the call except for the blank null fields.

In our program (see the listing), we will store the filename in the variable Path$. We can use the command VARPTR(Path$) to determine the two byte pointer (where the contents of Path$ are stored in memory) for the pathname variable. Now, we are ready to begin our MLI call. First, let's set up a LONG FuNction for the GET_FILE_INFO call that POKES the necessary information into the template. The first byte of the template is for the param_count. The param_count for GET_FILE_INFO is $A and the line, POKE $1F00,$A, stores it in the first byte of the template. The second and third bytes of the template for this call contain a two byte pointer (memory location) to the filename we are using. Since this is a two byte pointer we use the line, POKE WORD $1F01, VARPTR (Path$), to tell the MLI where the filename is stored. We only need to provide three bytes of input for this call. The two MACHLG statements are used to make the MLI call and to have ZBASIC display the error message. The first statement, MACHLG $A9, $C4, $20, $865; is machine code that tells ZBASIC which MLI call to do. The second number ($C4 in this example) is the MLI command number. You will need to change that line each time you program a different call. The MLI command numbers can be found in the ProDOS Technical Reference Manual. You can leave off the second MACHLG statement if you have your own error handling routine as it intercepts and prints any error messages. After we call the function, we can PEEK into memory and obtain information about the file and display it to the screen. Below is a LONG FN for the Get_File_Info which has the command number $C4.

```
LONG FN Get_File_Info (Path$)
  POKE $1F00,$A:              REM param_count is $A or 10 base 10
 POKE WORD $1F01,VARPTR (Path$): REM two byte pointer to our filename
 MACHLG $A9, $C4, $20, $865:    REM $C4 is MLI number for call
 MACHLG $90, 3, $20, $87F:      REM Error handler optional
END FN
```

If we first use the Get_File_Info call, we can then use the SET_FILE_INFO to change information about the file. Using this call, we can change the access (lock/unlock), the filetype, auxiliary type, the modification date, or the modification time. Since we used the GET_FILE_INFO call first, we have all the slots in the template filled. Let's keep our first try simple by locking the file. If the file is unlocked, we should find the value 195 or greater if we PEEK ($1F03). To lock the file, we simple POKE a zero into that location of template (POKE $1F03,0). Then, when we make the call, our file will be locked. We can unlock it by doing a POKE $1F03,195 and do another MLI call. (Of course, we should AND the original Access% with 32 to determine if the backup bit is enabled, and add that to the value we are poking.) Similarly, we can change the file type by poking a new value into $1F04. Table 1 identifies the file attributes you can change using the Set_File_Info call. Do not, however, try to change the filename with this call.

Changing the filename is done with the RENAME MLI call. We will need to make two changes to when we program the SET_FILE_INFO call. First, the param_count must be changed to 7. Second, the command number in the first MACHLG statement must be changed to $C3. The following program demonstrates the use of these two MLI calls.

Next time, we will take a closer look at how ProDOS stores the date and some new MLI calls. Just remember, changing the filetype of a text file to a system file does not make the file a system file. Be careful with your changes as you can cause some problems.

Sample MLI calls in ZBASIC

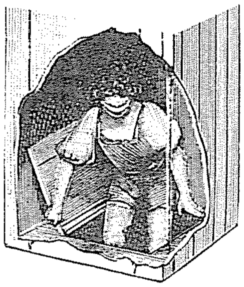
```
MODE=2
DIM 65 Path$:        REM max length of pathname plus 1 byte for length
INVERSE$=CHR$(15):   REM turn on inverse
NMAL$=CHR$(14):      REM turn off inverse
:
:
LONG FN Get_File_Info (Path$)
  POKE $1F00,$A:                  REM param_count is $A or 10 base 10
  POKE WORD $1F01,VARPTR (Path$): REM two byte pointer to our filename
  MACHLG $A9, $C4, $20, $865:     REM $C4 is number for MLI call
  MACHLG $90, 3, $20, $87F:       REM Error handler optional
END FN
:
:
LONG FN Set_File_Info (Path$,Access%)
  POKE $1F00,7:                   REM param_count is 7
  POKE WORD $1F01,VARPTR(Path$):  REM two byte pointer to our filename
  POKE $1F03,Access%:             REM value to change access
  MACHLG $A9, $C3, $20, $865:     REM $C3 is number for MLI call
  MACHLG $90, 3, $20, $87F:       REM Error handler optional
END FN
:
:
LONG FN Display
  :
  REM Now let's see what the call tells us about the file
  :
  Access%=PEEK($1F03)
  X=Access% AND 128
  LONG IF X=128
    PRINT"File can be deleted (Destroy bit is enabled)"
  XELSE
      PRINT INVERSE$" File cannot be deleted (Destroy bit is disabled)
"NMAL$
  END IF
  X=Access% AND 64
  LONG IF X=64
    PRINT"File can be renamed (Rename bit is enabled)"
  XELSE
    PRINT INVERSE$" File cannot be renamed (Rename bit is disabled) "NMAL$
  END IF
  X=Access% AND 32
```

```
   LONG IF X=32
     PRINT"File needs to backed up (Backup bit is enabled)"
   XELSE
      PRINT INVERSE$" File has not been modified (Backup bit is disabled
"NMAL$
   END IF
   X=Access% AND 2
   LONG IF X=2
     PRINT"File can be modified (Write bit is enabled)"
   XELSE
     PRINT INVERSE$" File cannot be modified (Write bit is disabled) "NMAL$
   END IF
   X=Access% AND 195
   LONG IF X=195
     PRINT"File is unlocked        access= "X
   XELSE
     LONG IF X=0 OR X=32
       PRINT"File is locked        access= "X
     XELSE
       PRINT"File is restricted     access= "X
     END IF
   END IF

   PRINT"File type is "HEX$(PEEK($1F04))
   PRINT"Auxiliary type is "HEX$(PEEK WORD ($1F05))
   X=PEEK($1F07)
   IF X=$0F THEN PRINT"File is the volume directory"
   IF X=$0D THEN PRINT"File is directory"
   IF X=1 THEN PRINT "File is a seedling"
   IF X=2 THEN PRINT "File is sapling"
   IF X=3 THEN PRINT "File is a tree"
   LONG IF X=$0F
     PRINT"Blocks used on volume are "PEEK WORD ($1F08)
   XELSE
     PRINT"Blocks used by file are "PEEK WORD ($1F08)
   END IF
   X=PEEK ($1F0B)
   Year=((X AND 254)>>1)+1900
   Month=X AND 1
   IF Month=1 THEN Month=8 :ELSE Month=0
   X=PEEK($1F0A)
   Month=((X AND 224)>>5)+Month
   Day= X AND 31
   PRINT "Mod_date is "Month"/"Day"/"Year
   X=PEEK($1F0C)
   Minutes=X AND 31
   X=PEEK($1F0D)
   Hour= X AND 63
```

```
  PRINT "Mod_time is "Hour":"Minutes
  X=PEEK ($1F0F)
  Year=((X AND 254)>>1)+1900
  Month=X AND 1
  IF Month=1 THEN Month=8:
  X=PEEK($1F0E)
  Month=((X AND 224)>>5)+Month
  X=X AND 31
  PRINT "Create_date is "Month"/"Day"/"Year
  X=PEEK($1F10)
  Minutes=X AND 31
  X=PEEK($1F11)
  Hour=X AND 63
  PRINT "Create_time is "Hour":"Minutes
  PRINT
  PRINT:
END FN
:
:
INPUT "Enter the full filename ";Path$
FN Get_File_Info (Path$)
FN Display
:
:
PRINT "Locked file= 0"
PRINT "Unlocked file=195"
INPUT"Enter value of access bit to change file to ";A$
A=VAL(A$)
:
LONG IF (Access% AND 32)=32:  REM was backup bit enabled?
  Access%=A+32:               REM yes, keep it enabled
XELSE
  Access%=A
END IF
PRINT"Changing file info"
FN Set_File_Info (Path$, Access%)
PRINT"New file info is:"
PRINT
FN Get_File_Info (Path$)
FN Display
END
```

Xpert: An expert in a jar?



One of the hottest topics in software engineering trade journals these days is artificial intelligence. And some of the most expensive software available commercially within this genre is of a type called "expert systems".

If you can forgive an oversimplification, an expert system is a program that helps an "expert" in some domain of knowledge or specific endeavor (maybe a geologist, teacher, or even a baker) transform a subset of their knowledge into a set of rules that the computer can use for decision making. Some folks are petrified over the legal consequences of this type of software; some of the hottest (and most expensive) packages help doctors make diagnoses. It is quite conceivable that a programmer or software house could be sued for contributing to medical malpractice! No wonder the packages are so expensive...

The evolution of this class of software shouldn't be surprising, though. Program bugs and crashed hard disks aside, computers *never* forget. Never. It is quite impossible, I should think, for a modern physician, lawyer, or even a software developer to have immediate recall for every single fact pertinent to their profession. Expert systems can be a real and powerful aid. The accumlated knowledge of an *entire* profession could be redacted to a CD-ROM, for example, for use by its members. A friend of mine is a lawyer - he could have his mail delivered to the law library at the University of Washington. Imagine if his office was equipped with a program that worked something like this:

Good morning, Mr. Smith. What is the nature of your current case?

It's a divorce with messy complications.

What is messy about it?

The parties cannot agree about custody of the dog.

I see. Can you tell me more about your client?

She is 30 years old, the mother of two children, aged 10 and 12. She is currently employed as a dental hygenist.

Okay. One moment...

Based on two precedents, it appears that your best course of action is to argue that the children would suffer unnecessary trauma if separated from the dog. Additionally, other successful arguments have held that a dog was necessary for the safety and security of an older child allowed to remain at home alone after school hours.

Click the "Precendents" box to view the associated precedents and their details...

I hope it is obvious that these systems do not ever really replace humans. They are a bit like your pocket calculator in that they just do what the human could already do - but faster and easier. It would be extremely dangerous for non-professionals to think they could practice medicine, for example, because they got their hands on a diagnosis related expert system. I'll be the first to predict that some fraud with a "diploma-mill" paper on the wall will try just that, however.

Knowledge is power, and power can be turned to do both good and evil.

As you might be able to surmise from our "sample run" above, an expert system has a lot in common with your run-of-the-mill database. Both must access tons of data quickly and efficiently. Much of the art in programming these beasts involves getting at the information itself in a speedy enough fashion. We'll be examining database design in future issues, to be sure.

The other primary hurdle involves establishing the "rules" by which a program interprets the data. Our mythical legal expert system above checked for relevant cases with a positive outcome and suggested strategies based on those used in those cases - an oversimplification, to be sure, but an example of the process nonetheless.

As is common with computers, yes/no, on/off sorts of relationships are far and away the easiest to manipulate in an expert system. One such system I've seen is available for high school science classes. Students are queried by the computer about the properties of a rock given to them by their teacher. Based on the student's observations, the software spits out a tentative identification. I say tentative because students are not always the best observers (an understatement). This particular program guarantees itself understandable student responses - the students may only supply words and descriptions offered in menus onscreen.

I'm certain that the software just keeps an array of all the "factors" or characteristics it uses for each rock. It then checks the "list" of possibles and reports back the best match.

Sorta like what we're gonna do in **Xpert**, this month's software adventure.

Although as literal minded as the rock ID program I mentioned, Xpert is a tiny bit different. Xpert is a "shell" into which you can place your own little domain of knowledge. It therefore requires two separate programs. The first module helps you set up and quantify your "knowledge". The second takes your data (analagous to the student's observations I mentioned), compares it to the knowledge database, and then returns with an identification.

In it's present form, Xpert is nothing more than a deformed database. It is brutal (you cannot even edit your entries once entered), it is slow (it saves its data in a standard sequential text file), and it is dumb (the comparisons it makes are crude match or no-match decisions). So why bother with it? Because it could be the core a truly wonderful expert system. And because you might learn a little from it.

If you provided better "rules", Xpert itself might be able to learn, too. With just a little more coding, the software might ask for some differentiating characteristic for items it cannot positively identify . For example, check the output of an old BASIC game called "Animals" in which the software guesses the animal the user is thinking of.

```
Does your animal have horns?

No.

Does your animal have a tail?

Yes.

Is your animal a dog?

No.

What is the name of your animal?

Cat

How is your animal different than a
dog?

It has pointed ears.
```

"Animals" then proceeds to write the new information to disk. The next time someone is thinking of a cat (assuming they don't cheat like I did), it would eventually add one more question before giving up. It would ask if the animal had pointed ears. If the user responded with a "yes", the program would guess they were thinking of a cat. If they were not, it would try to get more information into its knowledge base, just as in our example.

Xpert could also greatly increase its speed of operation on any computer with Btree schemes (for large knowledge-bases), or smart searches with small ones. A smart search would involve teaching the program to examine all of the "factors" in the database. A factor is one of the characteristics the system will use to make comparisons. If any factor was the same for all elements, it would be ignored in a search, thereby speeding up the process (comparisons take a lot of time).

I've already told you how to make Xpert better, perhaps now I ought to tell you how it *does* work.

The program first asks you to identify all of your "elements". In the context of this program, an element is one of the target items, analagous to the types of rocks stored in the geology knowledge-base we discussed earlier.

Next, Xpert will want to know all of the "factors" to be used in the identification or matching process. For example, let's say that you were creating a system that could identify certain people based on certain characteristics. Some of your factors might include height, weight, hair color, eye color, identifying marks, etc.

I *did* plug in a little intelligence - if there is not a perfect match for any of your elements, the program will report the closest match and display all of the factors for both it and the user's data. In this way, the program can at least deal with some measure of probabilities.

Speaking of probability, with only slight modifications to the code, Xpert could be turned into a decision analysis program. The elements would become the different options available. The factors would be the factors affecting your decision. You would go through and give a weight or rank to each element for each factor. The software would then add 'em up and suggest your best option.

That is, as they say, an excercise for the reader.

Xpert's code is nothing revolutionary. It was interesting to me how little code it took to get a workable system up and running. Xpert is minimal, of course, but I think it is a decent kernal for exploration.

I limited the potential number of factors and elements to 15 for no decent reason. In truth, laziness prevailed - such limits kept the screen from scrolling (cough, cough).

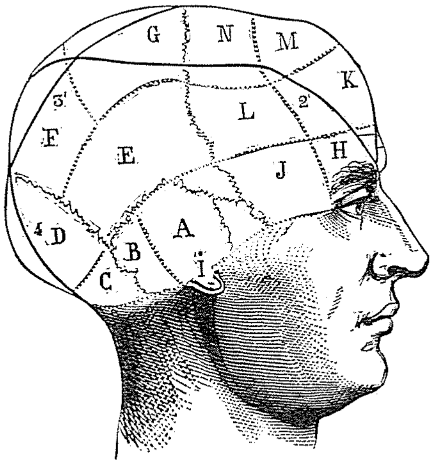
As I suggested earlier, expert systems seem to me to be glorified databases. The effectiveness of a given implementation is greatly constrained by the data management properties of the software. In future issues of **Znews** I'll be looking at relational databases and how you can write yer own.

Until then, then.

- Ross W. Lambert, Editor

```
REM --------------------------------
REM AI Experiment 1 - XPert
REM
REM by Ross W. Lambert, Editor
REM Znews
REM
REM - spaces req'd between keywords
REM - expressions optimized to integer
REM - locate statement in X,Y order
REM -------------------------------
  :
  :
 DIM 30 ELEMENT$(15),FACTOR$(15),
 IDENTITY$(15,15)
 DIM 81 L$, 2 INV$, NORM$
  :
 L$ = STRING$(80,"-")
 INV$ = CHR$(15)
 NORM$ = CHR$(14)
  :
 MODE 2
  :
 PRINT INV$;" XPert - a Mini-Expert
 System ";NORM$
 PRINT@(0,3)"Enter all of the elements
 to be identified (RETURN alone
 to end):"
 PRINT L$
  :
 "doElements"
 ID
```

```
DO
  ELEMENT = ELEMENT + 1
  PRINT@(0,ELEMENT+5)"Element number";ELEMENT;": ";
  INPUT "";ELEMENT$(ELEMENT)
UNTIL LEN(ELEMENT$(ELEMENT)) = 0 OR ELEMENT = 15 : REM 15 max or empty
string to end
:
"cont1"
TOTALELEMENTS = ELEMENT - 1 : REM nab total # of elements and reset
counter
ELEMENT = 1
LOCATE 0,3 : CLS PAGE
PRINT "Enter all of the factors to consider (RETURN alone to end):"
PRINT L$
:
"doFactors"  : REM cycle through and get each factor to use in making
an ID
DO
  FACTOR = FACTOR + 1
  PRINT@(0,FACTOR+5)"Factor number";FACTOR;":";
  INPUT "";FACTOR$(FACTOR)
UNTIL LEN (FACTOR$(FACTOR)) = 0 OR ELEMENT = 15
:
"cont2"
TOTALFACTORS = FACTOR - 1 : REM nab total # of factors and reset
counter
FACTOR = 0
:
FOR ELEMENT = 1 TO TOTALELEMENTS : REM get attributes of each element
  LOCATE 0,3 : CLS PAGE
  PRINT "For each element, enter the value for each factor:"
  PRINT L$
  PRINT INV$; "Element name: ";ELEMENT$(ELEMENT);NORM$
  FOR FACTOR = 1 TO TOTALFACTORS
    PRINT@(0,FACTOR+8);FACTOR$(FACTOR);":";:INPUT "
";IDENTITY$(ELEMENT,FACTOR)
  NEXT : REM factor loop
NEXT : REM element loop
:
LOCATE 0,3:CLS PAGE
PRINT"Saving matrix to disk..."
:
OPEN "O",#1,"XPERT.INFO"
PRINT #1,TOTALELEMENTS
PRINT #1,TOTALFACTORS
FOR X = 1 TO TOTALELEMENTS
  PRINT #1,ELEMENT$(X)
NEXT
:
FOR X = 1 TO TOTALFACTORS
  PRINT #1,FACTOR$(X)
NEXT
:
```



```
FOR X = 1 TO TOTALELEMENTS
  FOR Y = 1 TO TOTALFACTORS
    PRINT #1,IDENTITY$(X,Y)
  NEXT
NEXT
:
CLOSE
:
MODE 2
PRINT "Done!"
END
```

```
REM -----------------------------
REM AI Experiment 1
REM XPert.USER module
REM
REM by Ross W. Lambert, Editor
REM Znews
REM
REM -space req'd between keywords
REM -expressions optimized to int
REM -locate statemnt in X,Y order
REM -----------------------------
:
DIM 30 ELEMENT$(15),FACTOR$(15), IDENTITY$(15,15), INFO$(15)
DIM 81 L$, 2 INV$, NORM$
:
L$ = STRING$(80,"-")
INV$ = CHR$(15)
NORM$ = CHR$(14)
MODE 2
PRINT "Reading XPert.INFO..."
:
OPEN "I",#1,"XPERT.INFO"
INPUT #1, TOTALELEMENTS
INPUT #1, TOTALFACTORS
:
FOR X = 1 TO TOTALELEMENTS
  INPUT #1,ELEMENT$(X)
NEXT
:
FOR X = 1 TO TOTALFACTORS
  INPUT #1,FACTOR$(X)
NEXT
:
FOR X = 1 TO TOTALELEMENTS
  FOR Y = 1 TO TOTALFACTORS
    INPUT #1,IDENTITY$(X,Y)
  NEXT
NEXT
:
CLOSE
:
CLS
PRINT INV$;" XPert - USER MODULE ";NORM$
```

```
PRINT@(0,3);"Enter your data for each factor below and press RETURN."
PRINT L$
:
FOR FACTOR = 1 TO TOTALFACTORS
  PRINT@(0,5+FACTOR);FACTOR$(FACTOR);": ";:INPUT "";INFO$(FACTOR)
NEXT
:
LOCATE 0,3
CLS PAGE
PRINT "Working..."
:
FOR ELEMENT = 1 TO TOTALELEMENTS : REM loop through all elements
  FOR FACTOR = 1 TO TOTALFACTORS : REM ... and all factors for each
element
    IF IDENTITY$(ELEMENT,FACTOR) = INFO$(FACTOR) THEN MATCH = MATCH +
1
  NEXT
  LONG IF MATCH > BIGGESTMATCH
    IDENTITYELEMENT = ELEMENT
    BIGGESTMATCH = MATCH
  END IF
  MATCH = 0
NEXT
:
LOCATE 0,3
LONG IF BIGGESTMATCH > 0 AND BIGGESTMATCH < TOTALFACTORS: REM we got a
match on at least one factor
  PRINT "The closest element is: ";ELEMENT$(IDENTITYELEMENT)
  GOSUB "DisplayData"
  GOTO "Exit"
END IF
:
LONG IF BIGGESTMATCH = TOTALFACTORS : REM match on all counts
  PRINT "You exactly matched: ";ELEMENT$(IDENTITYELEMENT)
  GOSUB "DisplayData"
  GOTO "Exit"
XELSE : REM no matches at all
  PRINT "Your data matches none of the elements."
END IF
:
"Exit"
PRINT@(0,20)"Done!"
END
:
"DisplayData"
PRINT@(0,5) "Factors";TAB (30);"Your data";TAB
(60);ELEMENT$(IDENTITYELEMENT)
PRINT L$
FOR FACTOR = 1 TO TOTALFACTORS
  PRINT @(0,7+FACTOR);FACTOR$(FACTOR);TAB
(30);INFO$(FACTOR);TAB(60);IDENTITY$(IDENTITYELEMENT,FACTOR)
NEXT
RETURN
```

Don't miss a single issue! Re-up today!

Mail to: Ariel Publishing, Box 398, Pateros, WA 98846 or call (509) 923-2025 after June 10th

Name ____
Address ______________________________________
____
City ____________________________ State __________
Zip Code ________________________

Canadian and Mexican orders please add $5 per item checked. Non-North American orders add $15 per item checked.

_____1. **Znews** 1 year, $29.95
_____2. **Znews** 2 years, $52
_____3. **Quarterly disk** 1 year, $20 (see note below #4)
_____4. **Quarterly disk** 2 years, $35

Quarterly disk orders: indicate computer type (MS-DOS, Apple II, or Z-80) and preferred format (i.e. 5.25" or 3.5")

____

_____5. **ProTools,** ZBasic programmer's libraries, $39.95 (Apple II ProDOS only at present)
________ order total
________ extra shipping for non-USA (see above)
________ Total enclosed

Method of payment: _____ check _____ charge card _______ purchase order
If a credit card, check one: _____ Visa _____ MasterCard

Credit card or purchase order number:
____

Signature
____ Date ________________

Ariel Publishing
Box 266
Unalakleet, AK 99684

Mail to: