OS-9 USERS GROUP MOTD
DEDICATED TO EXCELLENCE IN OS-9 COMPUTING

Late Winter Edition 1993 | The Official Voice of the OS-9 Users Group | Volume 1, Issue Number 4

Out with the Old...

The new OS-9 Users Group is soon coming up on it's One Year Anniversary (*Has it really been that long?!*). All members should have received their ballots by now. We hope you will take the time to cast your votes for the new officers who will take up the reigns for the 2nd year. Please do mail your ballot back to us as soon as possible.

Upcoming Fests...

The "**Mid-Iowa and Country CoCo**" Fest is gearing up for a weekend event in Des Moines, Iowa (*Land of Microware!*) on March 27th and 28th. Rumors have it that 2 of the Famous (*Infamous?*) Mugsters on your CoCo will be in attendance. (Just press ALT-CNTL and Reset).

Guest speakers at the event will include James Jones from Microware, and our own Boisy Pitre, User Group President. This fest promises to be "heavy on the OS-9". Call 1 (800) 255-3050 for reservations.

Also, coming this May 1st and 2nd is the "**2nd Annual Last Chicago CoCoFest**" in Illinois. You can contact Tony Podraza (Glenside CoCo Club President) at (708) 428-3576 for more information. I hope to have more information in the next MOTD about this fest and maybe a report on the Mid-Iowa Fest as well.

Alan Sheltra, MOTD Editor

IconBASIC Review

by James Jones

Icon BASIC is a package from HAWKSoft that essentially puts a wrapper around BASIC09 (or Microware BASIC, as it's officially called on OS-9/68000, though I'll simply say "BASIC09" in the following) via pipes and does two things:

1. looks for various BASIC09 keywords in standard output and displays icons in their stead

2. lets you call up a graphical menu, click on the icon of your choice, and cause the corresponding keyword to appear in standard input of the process running BASIC09

Installing Icon BASIC is easy, and it comes with a program to let you edit your own icons.

Does Icon BASIC make life easier for a BASIC09 programmer? I must reluctantly say it does not, for various reasons.

1) Some of the icons are quite counterintuitive. "FOR" is a hand pointing upward, "NEXT", a hand with three fingers pointing upward, and "STEP" a hand with all fingers raised. (You thought that would be "STOP", not "STEP"? No; "STOP" is an octagonal traffic stop sign.) Scalar and string types are represented by values of those types.

Sure, I can edit the icons, but what should I replace them with? Graphically depicting the concept of real numbers instead of a specific real number is difficult, and perhaps the types would be best left un-iconized.

(*Continued on Page Three*)

FROM THE DESK OF...

by Boisy G. Pitre

In light of the recent developments concerning the OS-9 Users Group, we must ask ourselves "*Where are we going?*"

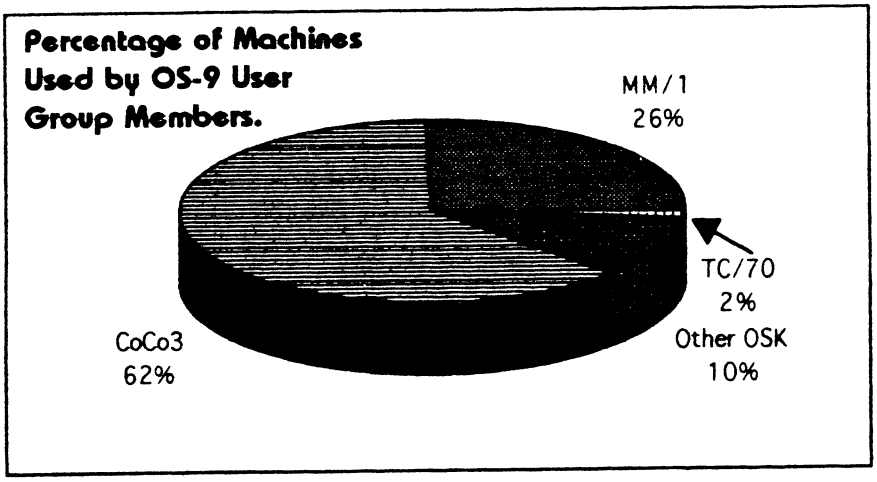
OS-9 has undergone many changes over the years. It will undoubtedly undergo more changes in the future. As a group which supports this operating system, we must adapt to these changes as well.

Unfortunately, the OS-9 Users Group is not representative of the current state of the OS-9 world. Out of curiosity (and because I've been asked several times), I perused our membership list and counted the number of OSK vs. the number of OS-9/6809 systems. Some members have both, and I counted these too. Here are the results:

CoCo 3	MM/1	TC-70	Other OSK (Gimix, UniQuad, etc.)
78	32	2	13

This comes to be 78 OS-9/6809 systems vs. 47 OSK systems. I don't believe anyone would say that 38% of OS-9 world uses OSK while the other 62% use OS-9/6809.

These numbers only reflect the current status of the OS-9 Users Group.



While we do have a commitment to our 6809 OS-9 members, we must understand the importance of keeping up with today's software.

If this organization is to become a force in the OS-9 Community, then we *MUST* follow the continuing evolution of OS-9. This means getting actively involved with Microware on enhancements and input. It also means embracing OS-9 and OS-9000, and letting go of OS-9/6809.

A new group of leaders will be coming into the picture in a few months. These people have the arduous task of leading this organization into a future of growth and productivity. It can be done, but it will require commitment to change.

-Boisy

Submit!

A Quick Note from the Editor...

This IS your publication! You are encouraged to submit any OS9-related article or program submissions for publication in a future issue of the MOTD. Guidelines for submission are as follows:

- Submissions should be in ASCII text format, unformatted text. Program listings may include TABs (prefered) if needed.

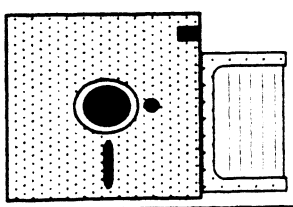
- Submissions may be sent on diskette (5.25" or 3.5", OS-9 CoCo or OSK format, also Macitosh format is accespatable if you have access. Disk(s) should be sent to:

MOTD Submissions
4650 Cahuenga Blvd., Ste #7
Toluca Lake, CA 91602

or, you may send your submission to the following E-Mail addresses.

CIS: 70673,1754
StGNet: MOTD@Zog
Internet: "Zog!MOTD@abode.ttank.com"
You may also call (818) 769-1938 and login as "MOTD" (no password) to upload your submission

- Deadline for the next issue is March 30th



Alan Sheltra

(*Continued from Page One*)

2. The filter needs to learn about string constants. If you If you insert the line:

```
PRINT "IF you can READ this THEN you're too darn CLOSE."
```

in your program, Icon BASIC will display the icons for BASIC09 keywords appearing in the string constant. (It does know not to do that when the string constant is actually printed. Making it notice string constants at other times should be pretty straightforward.)

3. Icon BASIC only concerns itself with language keywords, so you've still got typing to do. Icon BASIC is happy to let you type with the menu selected, and apparently will even feed the output through, but when I tried it, I couldn't see the results of my typing until I deselected the menu. Icon BASIC would be much more convenient if I could leave the menu around and still see what I was typing.

4. Icon BASIC would be a lot more convenient if the strings it emitted for the keyword-equivalent icons started with space. Otherwise, you're still stuck with having to hit the space bar at the beginning of every line you add to a BASIC09 procedure.

5. The icons are so big that they cut in half the number of source code lines one can display on the screen, which makes it harder to follow listings.

Icon BASIC could be much better, were it to take more of the job onto itself, i.e. doing the sort of work that so-called "syntax-directed" or "language-specific" editors do. To give some examples of what a BASIC09 GUI could do to help me as a programmer, here's a, you should excuse the buzzword, scenario:

I want to edit a new procedure. I click on an icon, and the program asks me its name, and then prompts me for any TYPEs needed (and lets me recall previously-specified TYPEs from earlier in the session, or maybe even from a file in my home directory where I've stored the results of previous sessions), any parameters, and any local variables. Parameters and local variables would be remembered and displayable in a list that I could call up and choose from just as I can choose an icon (thus eliminating the major remaining source of typing).

When I click on IF, I'm prompted for an expression and also whether I want an ELSE clause. Then icons for the complete IF schema, IF, THEN clause, and ELSE clause, appear—with clauses empty, ready for me to insert statements. The program would give similar assistance for other control flow constructs—more, perhaps, for FOR statements. (Imagine its putting up overlay windows with messages such as "You haven't used this variable before: what type should it be?" or "You're inside a FOR loop using the same control variable.")

You get the idea, though I've skipped over things it could do in the event of errors, or whether it should graphically represent control flow, or show a top-level view of the procedure and let one zoom in on portions of interest (a feature that various language-specific editors provide).

Icon BASIC is a neat idea, but it doesn't go far enough.

-James Jones

Programming with AWK

Part 1

by Zack C. Sessions

This is the third in a series of articles on the AWK Programming Language. In this article, I will concentrate on pure awk programming. I will use several example programs and also show how an awk program can be improved upon. In using awk programs as all of the examples, I will also be introducing other features and capabilities of awk which I have not previously touched on.

Since awk's real claim to fame is file processing, we'll need a few data files to play with. Create the following data files:

(Program 1)

```
$ build parts.dat
? p1 nut red 12 london
? p2 bolt green 17 paris
? p3 screw blue 17 rome
? p4 screw red 14 london
? p5 cam blue 12 paris
? p6 cog red 19 london
? <RETURN>
$ build emp.dat
? Beth  4.00  0
? Dan    3.75  0
? Kathy 4.00  10
? Mark  5.00  20
? Mary  5.50  22
? Susie  4.25  18
? <RETURN>
$
```

Let's say we want to know how many times the character "o" appears in the file parts.dat. Consider the following awk program:

```
{ for (i = 1; i <= length($0); ++i)
        if (substr($0,i,1) == "o")
         ++total
}
END { print "There were", total, "o's in the file." }
```

If this awk program were used to process the parts.dat file, the output would look like:

```
There were 9 o's in the file.
```

This program introduces the builtin functions, of which both length() and substr() are members of. Both of these normally operate on string variables.

Others operate only on numeric variables. In either case, if a variable is the wrong default type, its value is first converted to the appropriate value.

Program 2

Consider the standard word counting utility wc. When run against the parts.dat file, its output would be:

```
6   30   131 parts.dat
```

Let's see how hard this is to do with awk. Take a look:

```
{ nc += length($0) + 1
    nw += NF
}
END { printf("%7d%7d%7d %s\n",NR,nw,nc,FILENAME) }
```

There are two awk program lines in the awk program. The first program line does not have any pattern part, thus the action is applied to every record in the standard input stream. The second program line has a special pattern, the keyword "END", which means its action is performed just once, and after all input records have been processed.

This produces output identical to the wc command above. Note how I use the += additive operator, a la C. Remember, awk always initializes variables the first time they are referenced, thus the variables nc and nw are initialized to zero when the first record is processed. Remember also that NF is a builtin variable which contains the number of fields in the current input record.

Note also that the length of the entire record does not include the "newline" character at the end of every text record. That is why a 1 is added to the character count variable nc with each record processed.

Notice that the syntax of the printf() function is identical to the function of the same name in the C Programming Language. Also, a new builtin variable is being used, FILENAME. As you might expect, it contains the name of the file being processed.

Program 3

Consider the file parts.dat, it has one record for each part which is carried in each city of a multi location parts dealer. Let's say we want to know the number of parts in each of the cities. Take a look at the following awk program:

```
{ tot[$5] += $4 }
END { for (city in tot)
printf("City %s has %d total parts.\n",city,tot[city])
}
```

Here's what happens when we run it against the parts.dat file:

```
$ gawk -f Program.3 emp.dat
City rome has 17 total parts.
City london has 45 total parts.
City paris has 29 total parts.
```

This program introduces an entirely new concept for awk programs. This concept is known as the "associative array", and is one of AWK's most powerful tools.

Here, the subscript for the array tot[] is the name of the city, a character string. Since there is no pattern, each input record is processed by the action. The number of parts in the city rome are accumulated in the array variable tot["rome"].

Once all of the input records have been processed, we can dump out the totals in the END action. To access elements in an associative array, the awk programmer uses a special form of the for() loop, for (item in array), where array is a previously defined associative array, and item will be a variable which will be assigned the value of each index in the array, one at a time. Thus, the variable item will be a character string variable. The value of each element in the array can be either numeric or alphabetic, in our example they are numeric.

Think of the C code it would be required to perform such a feat! *Note*: awk processes ALL arrays as associative arrays, that is, even if the index of an array is numeric, the same awk internal processing code, the code which processes associative arrays. will process all references to members of that array.

Program 4

For our last awk program this time, lets look at that parts file. Let's say that we want to know how many parts of each color are at each city. Consider this awk program:

```
{ total[$5,$3] += $4 }
END { for (idx in total) {
        split(idx,x,SUBSEP)
        printf("There %s %d %s part%s in %s.\n",
        (total[idx] == 1 ? "is" : "are"),total[idx],x[2],
        (total[idx] == 1 ? "" : "s"),x[1])
        }
}
```

Here's what happens when it's run:

```
$ gawk -f Program.4 parts.dat
There are 17 green parts in paris.
There are 17 blue parts in rome.
There are 12 blue parts in paris.
There are 45 red parts in london.
```

This awk program again uses an associative array, and in this case, it is a two dimensional array. First dimension is the city name, and the second dimension is the part color. The value of the elements in the array is the total number of those parts.

Apparently, awk actually processes the multiple indexed array by simply concatenating the two indices together into a single character string where the two values are separated by a special separator character.

One argument for this is that to reference the index values separately in a subsequent for() look, the awk programmer must split up the values by using a new builtin function, split(). Also used is another built-in variable SUBSEP. This variable contains the default character used to separate subscript values for an associative array.

Notice also, that awk supports a powerful C expression, the conditional expression.

Extra!

Lastly, I'll present a "real world" example. Let's say I have a directory tree of many files in several directories. I realize that all of the files have public write access and I do not want that. I want to remove public access from the files. I could do the following:

```
$ dir -sur ! attr -z -npw
```

But this would change the access mask of all of the subdirectory files in the tree! I did not want to do that! Consider the following awk program:

```
$4 !~ /^d/ { print $7 }
```

OK, for any input record in which the fourth field starts with a lower case "d", that record is ignored. If the fourth field starts with anything else, the the seventh field of the record is written to the standard output path. What good is that? Well consider the following OSK command:

```
$ dir -sure
 0.0    92/12/27 1200  d-ewrewr    2E      64 DIR1
 0.0    92/12/27 1150  —w—wr       4    10476 file1
 0.0    92/12/27 1210  —w—wr       3A    1004 DIR1/file2
```

The s option tells dir not to sort the output, this makes the command run faster and sorted output is not really needed for our ultimate purpose. The u option means the output is unformatted, basically this means no header information, just one record of output per file. The r option tells dir to do a recursive search. You'll notice for files in the listing, the full path name is displayed for files below the current level. The e option asks for a full, or entire, directory listing.

Now, notice that field 4 is the attributes mask. For directory files, this mask will ALWAYS have a "d" in the first position. Conversely, all files which are not directory files will have a dash ("-") in that position. Now, notice that the file name is the seventh field in the display. Now that awk program is starting to make sense! What if I entered the following command:

```
$ dir -sure ! gawk '$4 !~ /^d/ {print $7}' ! attr -z -npw
```

Just what the doctor ordered!

Notice also that the same effect could be produced with either of the following awk programs:

```
$4 ~ /^-/ {print $7}

substr($4,1,1) == "-" {print $7}

substr($4,1,1) != "d" {print $7}
```

I'll leave it to the reader to determine which one is more efficient!

More awk programs next time!

Zack Sessions
sessions@seq.uncwil.edu
University of North Carolina at Wilmington (Alumnus)

DYNACALC PATCH FOR OS-9 V2.4

by Dr. Peter Dibble

This patch does two things. It converts an F$Mem SVC into an F$SrqMem SVC and it changes the way dynacalc deals with generated code for testing conditionals (the @if function).

Generated code has trouble with the 040's cache unless the cache is flushed after the code is generated and before it is executed.

The way I put the stuff on the end was to merge the dynacalc file with a file containing the bytes I needed to add, then patch the module length and use fixmod to pull everything together.

Patch is for dynacalc edition 112

Differences

	(hex)		(ascii)	
byte	#1	#2	#1	#2
========	==	==	==	==
00000007	02	3e	^B	>
00000009	00	01	^@	^A
0000000b	00	01	^@	^A
0000002f	c7	fb		
000009ff	ae	40	@	
00000a00	9c	00	^@	
00000a01	78	2 8	x	(
00000a02	2a	64	*	d
00000a03	00	06	^@	^F
00000a04	66	30	f	0
00000a05	0a	01	^J	^A
00000a06	20	4e	N	
00000a07	3c	ae	<	
00000a08	00	9c	^@	
00000a09	00	54	^@	T
00000a0a	00	2d	^@	-
00000a0b	ed	44	D	
00000a0c	4e	8b	N	
00000a0d	ae	66	f	
00000a0e	9c	2d	-	
00000a0f	54	4a	T	J
00000a10	2d	85	-	
00000a11	44	88	D	
00000a12	8b	22		"
00000a13	66	0a	f	^J
00000a14	2d	d0	-	
00000a15	4a	81	J	
00000a16	85	2d		-
00000a17	88	40		@
00000a18	20	85		
00000a19	0a	a0	^J	
00000a1a	d0	2d		-
00000a1b	85	40		@
00000a1c	2d	85	-	
00000a1d	40	a4	@	
00000a1e	85	2d		-
00000a1f	a8	40	@	
00000a20	2d	85	-	
00000a21	40	a8	@	
00000a22	85	20		
00000a23	a0	00	^@	
00000a24	2d	20	-	
00000a25	40	00	@	^@
00000a26	85	20		
00000a27	a4	00	^@	
00009b6e	4e	61	N	a
00009b6f	ae	00		^@
00009b70	80	55		U
00009b71	76	92	v	
0000db29	38	36	8	6
0000db2a	36	34	6	4
0000dc22	2f	2e	/	.
0000dc23	c3	35	5	
0000de34	02	d2	^B	
0000de35	07	d0	^G	
0000de36	04	02	^D	^B
0000de37	1b	d3	^[	
0000de38	01	31	^A	1
0000de39	18	d1	^X	
0000de3a	60	d2	`	
0000de3b	05	3d	^E	=
0000de3c	d0	e2		
0000de3d	12	20	^R	
0000de3e	00	39	^@	9
0000de3f	08	01	^H	^A
0000de40	d7	e5		
0000de41	0e	f1	^N	
0000de42	f1	88		
0000de43	99	e6		
0000eb02	54	4e	T	N

```
0000eb03   9c   ab
0000eb04   ac   ab
0000ec01   00   f9    ^@
0000ec02   01   00    ^A    ^@
0000f0ff   99   70          p
0000f100   b8   bf
0000f101   b4   c5
```

dynacalc is longer

Here is the end of the new dynacalc: starting at f102:

```
0000f102 48e7 8080 42e7 5d4f 3eae 8076 2f7a
0000f110 0026 0002 203c 0000 0004 4e40 005a 203c
0000f120 0000 0020 4e40 005a 44ef 0006 4e97 504f
0000f130 4cdf 0101 4e75 4283 4e75 001c d244
```

Dr. Peter Dibble

OS-9 User Group Tee-Shirts

The Demand for our "**Kick Butt in Real Time**" have surged We've just ordered another run of the popular in-your-face shirts Please hurry, quantities are limited

All shirt are made of 50% cotton, 50% Polyester and feature the "Kick Butt in Real Time" in front, and the OS-9 Users Group Logo on the back

Size	Member Price	Non-Member Price
L, XL	$13 00	$15 00
XXL	$14 50	$16 50

Send your check or M O to:
The OS-9 Users Group
P O Box 71131
Des Moines, IA 50325
(Please state Quantity and Size)

Los Endos...

Zack Sessions, our User Group Librarian has just recently compiled the new UG Library disks. Please check out the back cover to see what offerings we have available. These are *ONLY* available to User Group Members. I'm sure this list will be appreciated, as we have received numerous requests for these.

This concludes another issue of the MOTD. We hope to see some of you at the upcoming Mid Iowa Fest. Keep "Kick'n Butt"...

Alan Sheltra (MOTD Editor)

The OS-9 Users Group Library

•Package #1. The *Original* Users Group Library

Consists of the group's FileMaintenanace (2), Finance (1/2), Communications (2), File Processing Filters (1), System Software (1), Text File Processing (6), Games (1), Graphics (1), Database Management (1), Programming Aids (1), Binary File Processing (1/2), Mathmatics (1), Word Processing (1/2), System Utility (2), Languages (1), Text File Output Routines (1). Each group consists of 1/2 to 6 *Units*. Cost is $5.00 per 3 Units, $9.00 per 6 Units, $13.00 per 9 Units, or purchase the *entire library* for **just $30.00!**

•Package #2. The OS-9 Project (TOP) Disks

Get the entire set of TOP disk for just $30.00!

•Package #3. The European Forum for OS-9 (EFFO) Disk

This entire library is also available for just $30.00!

Remember! The OS-9 Users Group Library exists for and by it's members!
If you you have something worthwhile to contribute to the library please send it to:

The OS-9 Users Group Library
P.O. Box 540
Castle Hayne, NC 28429

To place an order, send your check or M.O. to:

The OS-9 Users Group
P.O. Box 71131
Des Moines, IA 50325

*Please specify the format you wish to have your software returned to you in. Available formats are Universal, MM/1, Atari ST and Color Computer. 5.25" only for Color Computer format. MM/1 is a High Density format, all of the others are Double Density. The TOP and EFFO Packages are specific to OS-9/68000 ONLY! The original library has a combination of OS-9 Level 2 and OS-9/68000 software. Please include $3.00 per package to cover the cost of shipping. Sorry, you MUST be a member of the **OS-9 User Group** to take advantage of these deals.*
(Iowa residents please add 5% salea tax.)