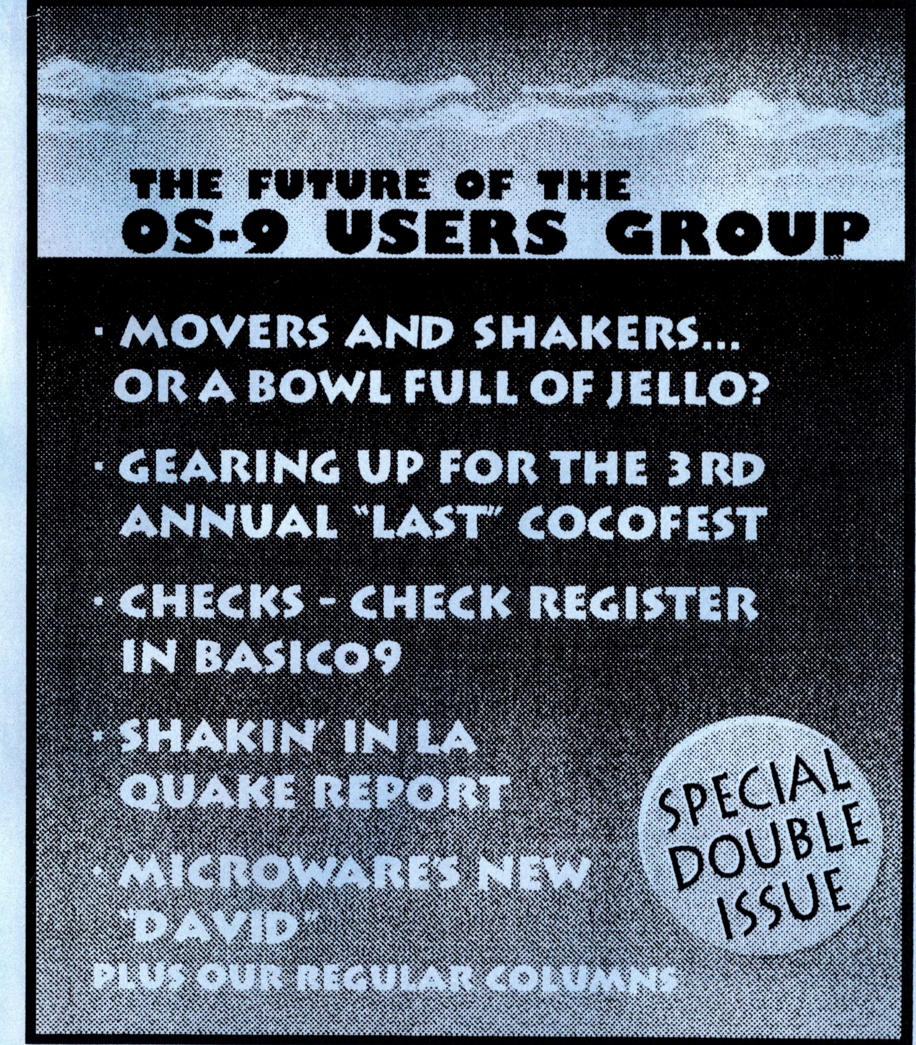
Volume Two, Issue Two

$3.00

The "International" OS9 Underground®

A Fat Cat® Publication

Magazine Dedicated to OS-9/OSK Users Everywhere!



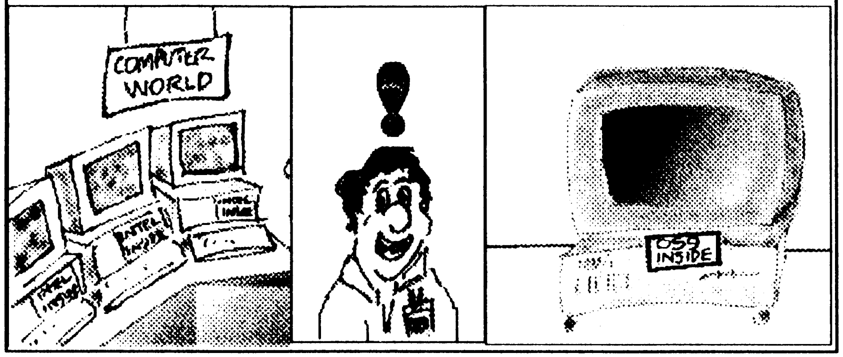
THE FUTURE OF THE OS-9 USERS GROUP

- MOVERS AND SHAKERS... OR A BOWL FULL OF JELLO?
- GEARING UP FOR THE 3RD ANNUAL "LAST" COCOFEST
- CHECKS - CHECK REGISTER IN BASIC09
- SHAKIN' IN LA QUAKE REPORT
- MICROWARE'S NEW "DAVID"

PLUS OUR REGULAR COLUMNS

SPECIAL DOUBLE ISSUE

Shell Game by Alan Sheltra



Advertisers Index

Vendor	Page
ColorSystems	IFC
Glenside Color Computer Club	3
BlackHawk Enterprises	11
Bob van der Poel Software	13
JWT Enterprises	16
ARK Systems USA	17
AniMajik Productions - Software	19
Peripheral Technologies	19
Northern Xposure	21
Sub-Etha Software	23
Dirt Cheap Computer Stuff Co.	29
SoftCopy	34
AniMajik Productions - Graphics	37
Farna Systems	46
CoNect	50
Computer Design Services	58
09-Online	58
FAT CAT Publications Bookshelf	IBC
DELMAR Co.	BC

Let these fine vendors know you saw it in The Underground!

The "International" OS9 Underground® Magazine

Contents

HOW TO REACH THE OS9 UNDERGROUND:

For subscription information, questions, call or write to:

Fat Cat Publications
4650 Cahuenga Blvd., Ste #7, Toluca Lake, CA 91602
(818) 761-4135 (voice) · (818) 365-0477 (fax) · (818) 769-1938 (BBS)
or by email to: **Editor@zog.wa.com or EditorOS9U@AOL.com**

Article Submissions may be sent to the above addresses or to the following special internet account set up for that purpose: **submit@zog.wa.com**
Submissions should be sent as plain text and unformatted.
Join us on the New OS9 Underground Email, Mailing List: **FatCat@zog.wa.com**
(The FATCAT mail list has moved to a new permanent address)
To Subscribe, just send email to listserv@zog.wa.com and in the message type:
"subscribe fatcat *your name*"
your name – your real name

Fat Cat Publications

Comments to our columnists may be sent to:

jev@zog.wa.com	Jim Vestal
cos@zog.wa.com	Steve Secord
leonard@zogwa.com	Leonard Cassady
bvdp@zog.wa.com	Bob van der Poel
wayne@zog.wa.com	Wayne Campbell
smgee@wa.com	Scott McGee
subetha@wa.com	Allen Huffman
blkhawk@zog.wa.com	David Graham

OS9 Underground Magazine Member Card

Participating Vendors

These vendors will offer the following discounts for OS9 Underground Member Card Holders

· CoNect	10% Off any order
· Sub-Etha	10% Off any order
· Canaware	10% Off WristSavers
10% Off WristSaver MousePad & 15% Off ENC9	
· AniMajik Productions	20% Off all Software
· Fama Systems	10% Off any order
· Bob van der Poel Software	10% Off OSK Software

Software/Hardware Vendors...you can be listed here FREE! Contact the OS9 Underground for details. You need not be an advertiser for this Free service.

Be sure to give your card number when you place an order with these fine vendors. Not responsible for typos or mis-prints

```
/* st - a control char strip routine by
      Jim Vestal, features include:
1) strip or filter out all unwanted control
   characters from standard input
2) will not affect standard carriage returns
3) will convert Unix style end-of-line to OS-9
    style end-of-line by changing linefeeds to
    carriage returns, and at the same time,
4) will convert MS-DOS style end-of-line to
    OS-9 style end-of-line by stripping out the
    linefeeds followed by the carriage returns.
5) will convert ascii tab control character to 5
    spaces
*/

/* include standard input/output header file, used for
high level functions such as getchar/putchar */

#include <stdio.h>

/* beginning of main function */

main()
 {
 /* Declare Variables: */
 /* in, ascii value of charactor read from
    standard input */
 /* lfflag, flag for carriage return/linefeed
    input (0=false/1=true) */

 int in,lfflag;

 /* initialize variables */

 /* set linefeed flag to false */
 lfflag = 0;

 /* main while loop */

 while (in != EOF) /* loop until end of file is
true */
  {
  /* read standard input and store the value of
     the character into in */
  in = getchar();

  /* convert characters with high bit set into
     their high bit reset value */

  if ( in > 127 )
  in=in-128;

/* print non-control characters to standard
   output */

if ( in > 31 )
  putchar(in);

/* process tab by printing 5 spaces to
   standard output */

if ( in == 9 )
  printf("     ");

/* Process OS-9 EOL by printing carriage
   return to standard output */
/* and setting lfflag to true. */

if ( in == 13 )
  {
  putchar(in);
  lfflag = 1;
  }

/* Process Unix EOL by changing linefeed
   to carriage return and printing carriage
   return to standard output. */

if ( ( in == 10 ) && ( lfflag == 0 ) )
  {
  in = 13;
  putchar(in);
  }

/* Process MS-DOS EOL by stripping (not
   printing) the linefeed following the carriage
   return and setting lfflag to false. */

 if (( in == 10 ) && ( lfflag == 1))
   lfflag = 0;

} /* endwhile */

/* end main */.
```

(EOF)

Dedicated to OS-9/OSK Users Everywhere

Volume Two, Issue Two/Three

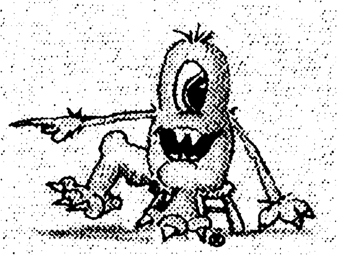
Editor/Publisher:
Alan Sheltra
Assistant Editors:
Jim Vestal
Steve Secord
Technical Editor:
Leonard Cassady
Contributing Editors:
Wayne Campbell
Allen Huffman
Scott McGee
David Graham
Eric Levinson
Boisy Pitre
Bob van der Poel
Typesetting/Layout/Artwork:
AniMajik
Productions

Fat Cat Publications and The "International" OS9 Underground Magazine and it's logotypes are registered trademarks. Subscription rates are $18.00 for 12 issues ($22.00 Canada, $27.00 overseas US Funds). Single or back-issues are available at the cover price (please call or write for availability). Fat Cat Publications is located at 4650 Cahuenga Blvd., Ste #7, Toluca Lake, CA 91602 · (818) 761-4135 (Voice), (818) 365-0477 (Fax) or (818) 769-1938 (Modem).

Under It All

(Editor's Column)

by Alan Sheltra (ZOG)



Shakin' and Bakin' in CA

QUAKE As you are all aware, January 17, 1994 is a date I and many in the San Fernando Valley (*a very large residential area of Los Angei*) won't forget for quite a while.

I, and many other fellow OS-9ers, living in the SF Valley, suffered a lot of damage. Luckily everyone's computers survived (of the OS-9ers that I talk with).

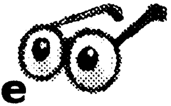
The damage and set-back this quake caused is the reason why the February was delayed (and never made it) and why it was decided to combine the February and March Issues into a double issue to get the schedule somewhat back on track.

This issue has taken enourmous amount of time to put together since I pretty much did the whole thing. Many of the people who have helped to proof-read the text, were not able to in this issue, due to their own problems caused by the earthquake.

I am happy to say that things are getting back to normal once again, and starting with the April issue, the Underground's schedule should be back to normal once again.

Remember, this issue is a double issue and counts as 2 issues.

New Style, New Type



Starting with this double issue, you will notice that things are a little different around here.

The format has been changed slightly, as well as the font used for body copy. This san-serif type is called "Antique Olive".

I think this type style will easier on the eyes, and more legible than before. Those of you with impaired vision will be better able to read this text now.

I hope you will find the changes pleasing as well as functional.

The "‹–", "›–", "‹", and "›" are also valid in C.

In C the use of braces "{" and "}" are used in a similar fashion to the "BEGIN" and "END" statements in Pascal. IF statements do not have a "THEN" but instead should have a "{" following the condition. In place of an "ENDIF" statement a "}" must follow.

That pretty much covers everything for this month. If you have the book "The C Programming Language" by Kernighan and Ritchie please read through the first 3 chapters. Some of the material in the book will be covered next month. Chapter 3 is the chapter on loops and control flow, If-Else, Else-If, Switch, While and For loops will all be used in the code for next month's lesson and the lessons following.

If you have any questions about this month's article you can call me or leave email about it. I'll be glad to discuss or answer any questions. My internet email addresses are:

jev@zog.wa.com
sysop@narnia.citrus.sac.ca.us
jevestal@devo.sfasu.edu

```
/*  Listing #1
alf - a filter that adds linefeeds following
carriage returns - also filters out controls
 include standard input/output header file,
used for high level functions such as
getchar/putchar
*/


#include <stdio.h>

/* beginning of main function */
```

```
main()
 {
 /* Declare Variables: */
 /* in, ascii value of charactor read from
    standard input */

 int in;

 /* initialize variables */

 /* main while loop */

 while (in != EOF) /* loop until end of file */
 {
  /* read standard input and store the value of
    the character into in */

  in = getchar();

  /* convert characters with high bit set into
     their high bit reset value */

  if ( in > 127 )
  in=in-128;

  /* print non-control characters to standard
    output */

  if ( in > 31 )
   putchar(in);

  /* Process OS-9 EOL by printing carriage
   return to standard output  and then printing
   a linefeed following any carriage return */

  if ( in == 13 )
    {
    putchar(in);
    putchar(10);
    }

 } /* endwhile */

} /* end main */
```

There are many programs that are written as filters such as MORE and SORT. These will read from the standard input path, process information until <eof> while sending processed output to the standard output path.

One of the more common uses for filters is to strip linefeeds control characters from text files. I wrote such a program called ST that not only strips control characters from files but converts linefeeds from Unix formatted text files to carriage returns needed for OS-9 text files, while at the same time strips the linefeeds from an MS-DOS formatted text file. As a bonus I had the program process ascii tab characters, replacing them with 5 spaces for each tab (ASCII 9). The S source code follows this article.

I have friends that use MS-DOS systems and at college I use them too on occasion. I had a need for a filter to add linefeeds to OS-9 formatted text files so that MS-DOS users could use them without having to worry about their utilities crashing because of abse linefeeds following carriage returns. I took the code from ST.C and deleted the unwanted code and added the code to add linefeeds upon receiving a carriage return. This program is called ALF and the ALF.C follows this article.

Next month I will present new versions of ST.C that adds command line options and other neat features. Thus building on this month's lesson.

Reading the C source is easy, I have commented most everything.

As in Pascal every progra statement must end with a ";". "/*" and "*/" are comment delimiters, similar to "(*" and "*)" in Basic09 and Pascal.

As in Pascal all variables must be defined at the beginning of each C function. In ST.C there are two variables declared with the following line:

```
int in,lfflag;
```

This is similar to the BASIC09 line:

```
DIM in,lfflag:INTEGER
```

The statement:

```
while ( in != EOF)
```

can be read as "while variable in is not equal to EOF",

EOF is defined in the stdio.h header file as a -1, which inicates the end of file condition.

The code: in = getchar()

reads 1 character of input from the standard input path and stores it's ascii value into the integer variable "in".

Toward the end of the code are 2 compound IF statements. One of them reads:

```
if (( in == 10 ) && ( lfflag == 1))
```

This IF statemant can be read as "IF variable in is equal to 10 AND variable lfflag is equal to 1 THEN".

In Basic09 and Pascal it is common practice to use a ":=" for variable assignment and a "=" for comparisons. In C, comparisons are coded with the "==" and variable assignment are coded with the "=". The "!=" means NOT EQUAL, the "||" means OR and the "&&" means AND.

FATCAT LIST HAS MOVED

FatCat Publications Email List has moved from it's previous location on AOL to it's new permanent location on an OS-9 machine. This , I beleive is the first mail list run on an OS-9 machine.

This mail list is broadcast to all it's subscribers, much like this magazine is mailed (snail mail) to it's subscribers.

The FatCat List is for discussion about and feedback to the OS9 Underground, comments, good or bad. Most of our authors read this list.

How do you subscribe you ask? Well, it's real simply, but the catch is, you must have an internet address or have access through a provider such as Delphi, CIS, GEnie or AOL.

To subscribe simply send email to:

```
listserv@zog.wa.com
```

In the body of your message, type:

```
subscribe fatcat your name
```

"Your Name" should be your real name, not your handle or internet address.

To send a message to the list (so others may see it), send mail to the mail list's name.

```
ex: fatcat@zog.wa.com
```

Besides, the fatcat list, there are currently 2 other lists available, those are: "os9ers" and "sci_fi". You may subscribe to those at any time as well.

The listserv is a robot account which interprets commands sent to it. You may request a list of those commands by sending the command 'help' in the body of your message.

Some late-breaking news from Glenside

Tony Podraza of Glenside, had some last minute news. Since the submission of his article (page 36), 4 more vendors have confirmed attendance at the Fest, bringing the total to 15. They are:

1) Barsoft, Dave Barnes
2) ColorSystems, Zack Sessions
3) DELMAR, Ed Gressick
4) Hawksoft, Chris Hawks
5) Farna Systems, Frank Swygert
6) Crystal Palace BBS, Nelson Howard et al
7) DISTO, Tony Distefano
8) Budgetware, Brian Kitt
9) JWT Enterprises, Jordan Tsvetkoff
10) Adventure Survivors, L. E. & Nan Padgett
11) The Chicago OS9 User's Group
12) The National OS9 User's Group
13) StrongWare, John Strong et al
14) SBUG, Andre Levalle
15) Blackhawk Enterprises, David Graham

In addition to confirmation of these vendors, several seminars have been penciled in.

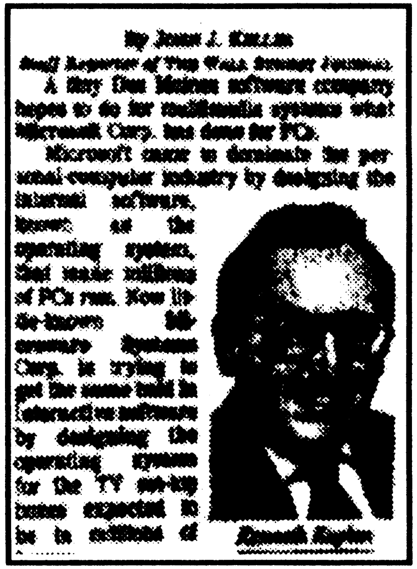
These include:

- Networking and the Internet,
  by Scott Griepentrog
- Nitros9
  by Bill Nobel & Colin McKay
- ACBBS
  by Nelson Howard & Friends

Other as yet, un-named seminars, will be held by, Allen Huffman and John Strong. The Rev Bros. Jeremy, will give a meditation, praise and worship service on Sunc moming of the fest.

Aiming to be a Giant

Microware Systems Corp, Des Moine, IA, is looking to be the Multimedia giant. in interactive software.



By John J. Keller
Staff Reporter of The Wall Street Journal
A tiny Des Moines software company hopes to do for multimedia systems what Microsoft Corp. has done for PCs.
Microsoft came to dominate the personal-computer industry by designing the internal software, known as the operating system, that make millions of PCs run. Now little-known Microware Systems Corp. is trying to get the same hold in interactive software by designing the operating system for the TV set-top boxes expected to be in millions of

Kenneth Kaplan

Ken Kaplan, President and fouder of Microware Systems Corporation

The OS/9 operating system will be used in the next generation of interactive TV set-top boxes.

According to the Wall Street Journal, *"Little known Microware has just landed a coveted piece of business, Bell Atlantic Corp., the regional Bell company, expects to distribute an interactive service to 10 million households by the turn of the century".*

This deal with Bell Atlantic should bring our favorite OS in the homes of the masses, even if they don't know it's there.

OS/9 is already used in the real world, running toll booths, traffic lights, to controlling the particle accelerator at the University of Cern, Switzerland, smashing atoms, and running space shuttle for NASA

BIG BAD MICROSOFT

Microsoft at one point (in 1986), had approached Microware with the intentions of buying them. Microsoft, not wanting to reinvent the wheel by craeting a new operating system, wanted Microware's Microware refused.

Later, Microsoft, wanting to get it's hands into multimedia, teamed up with Intel and General Instrument Corp., but suffered delays getting their simular set-top product to the market. Microsoft also wanted a piece of the pie, by charging the end-user a fee everytime a transaction was made throu their set-top box, which cat companied balked at. In a head to head test conducted by Bell Atlantic, Microware beat out Microsoft

BASIC TO C

by Jim Vestal

Welcome to the all new Basic to C column. This is a column for the BASIC programmer who wants to learn C. I am in the ocess of learning C, so I'll teach you what I learn as I learn it!

For a year now I've been reading everything I can about the C language, I have learned alot, but just reading about C doesn't allow you to learn as much as actually programming in C. So I decided that I'm going to jump in head first and start programming in C, hopefully some of you can learn from my programming.

What is a filter?

A filter program is any program that reads input from stdin and sends output to stdout. It can be used with redirection or with pipes from the command line to process information in a file. Here are examples of how to use a filter program from the OS-9 command line:

```
list file.txt ! filter › new.file
filter ‹ file.txt › new.file
merge file.txt ! filter › /p
```

I have seen instances where the use of a "goto" statement actually makes the current coding more efficient.

iteration. As with everything, too much of anything is not necessarily good. Multiple nested "if" statements may be replaced with the "switch" statement and tend to improve execution speed.

GOTO

The use of the "goto" statement has been argued extensively. I have never run into a program task that absolutely required the use of a "goto" statement, however I have seen instances where the use of a "goto" statement actually makes the current coding more efficient.

Regardless of how you may feel about the "goto" statement, there IS a reason for its inclusion in the language. Prolific use of the "goto" statements should be avoided as it tends to lead to "spaghetti" coding and the program logic becomes difficult to follow.

Whichever flow control method you choose, readability should be the most important factor.

-Leonard Cassady

Comments may be addressed to:
(Leonard@zog.wa.com)

WRead - Continued from page 31

```
08D0   PRINT "79 characters each (NOT
        including EOL characters)"
0905   PRINT
0907   PRINT "once running, use the
        following keys to move around:"
093F   PRINT
0941   PRINT "        [UP]/[DOWN] Arrow
        - move up or down one line"
097A   PRINT " [SHIFT]-[UP]/[DOWN] Arrow
        - move up or down one screen"
09B5   PRINT "           [T]/[B] Keys
        - move to top or bottom of file"
09F3   PRINT "              [Q] Key
        - quit wread"
0A1E   END
```

-Wayne Campbell

Change of Address:
If you anticipate a change of address, you must notify Fat Cat Publications at least 30 days prior to your move to continue uninterupted service. We DO NOT re-send returned mailed. To re-claim returned mail, you must send $1.00 postage and handling per issue to the address below.

Back Issues:
Back issues are available at the cover price. Call or write for more info.

Fat Cat Publications
4650 Cahuenga Blvd., Ste #7
Toluca Lake, CA 91602

Fat Cat Publications

hands down!

DAVID with an OS-9 Heart

The software that with run these TV set-top decoders is called DAVID (Digital Audio/Video Interactive Decoder). Created by Microware to run under OS/9.

A set-top decoder connects to a subscriber's home with an interactive TV network. It received digital information streams over telephone or cable TV lines from video servers distributed accross a network. The set-top then decodes or decompresses the information into applications such as movies-on-demand or electronic shopping as well as educational services. The DAVID system operated and controls all functions of the set-top decoder.

(Excerps from Motorala Press Release and Wall Street Joural)

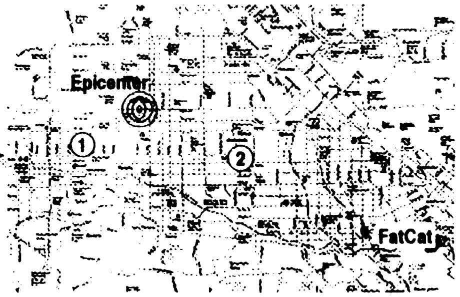
SHAKIN' IN LA

How did OS9-ers fare during the January 17 Quake in So. California? Well, most are more or less back to normal, but repeated aftershocks keep nerves jangling.

Even serveral miles from the quake's epicenter caused structural damage to FatCat, many OS9ers, some who write for this magazine, live even closer and suffered worst damage.

At 4:31 that Monday morning, I was rudely awakened by the most violent shaker I have ever felt (I experienced the 1971 and the Whitter Quake after that). Everything that could fall or topple, did, including TV sets, shelves, speakers, books, dishes and my TC70. The TC70, which sits on a counter top, landed 6 feet away from it's perce, upside-down!

7 hours later after the power had been restored (I was lucky, some had power out for days) and had cleared enough rubble to walk around without tripping on something, I set the TC70 back up on it's perch again. I assumed at that point, the machine was deader than a doornail after the fall it took, but decided to shove the cards back into the K-Buss and fire here up for laughs. I was flabbergasted! It booted and there was the good ol' OSK "$" Prompt. Tough little machine, that '70!



(1) Leonard Cassady, Technical Editor
(2) Wayne Campbell, Staff Writer

The above map of the San Fernando Valley will give you somewhat of an idea as to the relative location of two of our writers to Fat Cat's location. Other OS9ers, not listed here all fared about the same. Luckily everyone of them were okay.

-ZOG

New Discoveries

by Jim Vestal

New Discoveries contains a brief overview of new products, hardware, software (both commercial and shareware/public domain). If you are a vendor or a software author and would like to submit a list of new products please send a general description to Jim Vestal c/o this magazine or via internet email: "Discover@zog.wa.com"

Commercial Products:

The OS-9 Guru 1 - A new book about OS-9

The first in a series of books, Paul Dayan's OS-9 Guru - 1 is a highly detailed introductory and technical referance book on the OS-9 operating system. Available in the United States for $44 (includes shipping) from Windsor Systems, 2407 Lime Kilm Court, Louisville, KY 40222, phone (502) 425-9560.

Smash - Breakout Style arcade game for OS-9 Level II

Written by Alan Dekok, 32 levels, 17 different block types. Available for $29.95 (U.S. funds) from Northern Xposure, 7 Greenboro Cres, Ottawa, ON, Canada K1T1W6, phone (613) 736-0329.

Shareware and Public Domain:

SHANGHAI 1.2: MAHJONGG GAME

Version 1.2 of SHANGHAI for OS-9 Level II. This final version has faster graphics and a faster Dragon Builder. New artwork is included as well as soure code. It will work with your previous data files from version 1.0. The game also reports when there are no more moves for the player to make. Available on Delphi and O9-Online Systems

ANSIFRONT 0.9: ANSI C PROCESSOR

AnsiFront v0.9: Ansi C pre-processor front end to MWC compiler for OS-9/6809. This updated version of Ansifront is now on Delphi and available from O9-Online Systems.

SCRED PATCH

Patch to fix SCRED from crashing in insert mode when typing an ALT char. By David Breeding. Available on Delphi or from O9-Online Systems.

SHANGHAIED (A TILE EDITOR)

A Tile Editor for Shanghai OS-9. Create/edit your own Shanghai tiles! Source code is included. Errata for Shanghai v1.1 is also included. Available on Delphi or from O9-Online Systems.

COCO 2 EMULATOR FOR MSDOS

COCO 2 Emulator for MS-DOS PC. Turns a PC into a TRUE COCO2. Available on Delphi or from O9-Online Systems.

PALM V2.1 - ELM-LIKE MAIL EDITOR

Palm v2.1 is the first non-beta release of Palm.

Palm is "full-featured" Elm-like mail editor to be used with either Rick Adams UUCP package (4.2 is what it was tested on) or Bob Billson's UUCPbb package (2.0). Be sure to read the docs and set up the makefile and palm.h to match your system before compiling. If you have the old beta version of Palm, trash it. Too much has changed. Available on Delphi or from O9-Online Systems.

UUTOOLS

UUE and UUD are a set of enhanced UUENCODE/UUDECODE tools which can dynamically size output files into multiple chunks and automatically reassemble and decode them into object files. Available on Delphi or from O9-Online Systems.

Continued Page 12

Continued from Page 57

or "xxxxx.r" files into a single file. The "linker" is invoked to combine the re-locatable, or "xxxxx.r" files and the system library or "xxxxx.l" files into an executable program. Smaller, complete programs are usually written by one programmer and comprised of a single source file containing all the necessary algorithms or functions as well as the program flow control, (usually a 'main' routine and subsequent functions). Larger programs, written by 'teams of programmers', usually require a 'makefile' to combine the numerous object files into the final, executable form.

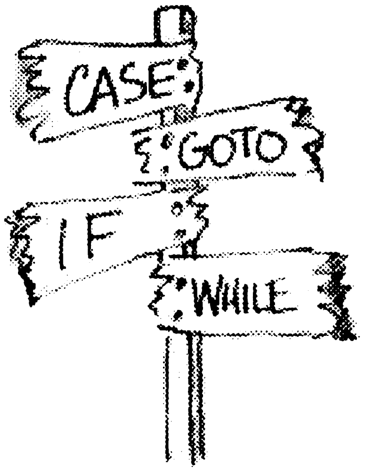
I would encourage the use of the "make" utility and a "makefile" to combine the various program algorithms as this approach tends to be more productive, especially when separate modules go through many revisions regardless of how many programmers are involved.

FUNCTIONS

The most important concept of high-level languages is that of "functions". Also known as "subroutines" or "procedures", they are a collection of the language's basic operations. Usually they perform an operation more complex than any built into the language. However, they should not so complex that they are difficult to understand.

The lower-level functions perform the simplest operations, and higher-level functions are created by combining lower-level functions. Another way to think of a function is an abbreviation for a long, maybe complex, set of commands. A function is like a separate program that accepts data as input, processes it in a defined manner, and returns result/results to the calling program, (the 'void' function returns a special 'void' type result of NULL, which is a different type than the numeric type result 'return(0);' returns).

Functions should be small, yet general. They should be written so that their tasks can be modified by changing the input. As the program develops, it might be necessary to rewrite the algorithms. Even the most experienced programmer can't write perfect code every time.



FLOW CONTROL

Most high-level languages provide mechanisms for efficient flow control through the use of conditional-branching and looping, or

Shareware and Public Domain (continued)

VFY #11: 6809 "FIXMOD" UTIL

An expanded function vfy util, now does the OSK "fixmod" stuff for OS-9/6809. Available on Delphi or from O9-Online Systems.

Online Services

LOWER PRICING FOR EXTENDED SERVICES

CompuServe members billed under the Standard Pricing Plan now pay lower rate for access to extended (hourly priced) services. The new rates for CompuServe' extended services are $4.80 per hour for access at 300, 1,200 and 2,400 bps, a $9.60 per hour for access at 9,600 and 14,400 bps. This pricing change reflect up to a 40 percent reduction from previous rates for extended-service products Communications network and product-specific surcharges still apply.

The monthly basic services rate of $8.95, which provides access to more than 50 product areas, remains unchanged, as do rates for CompuServe members billed under the Alternative ("pay-as-you-go") Pricing Plan.

Through this pricing reduction, CompuServe is passing along cost savings realized by the use of advanced technology and "commodity"-priced hardware in its host computer systems.

For complete information about CompuServe's Standard and Alternative Pricing Plans, to check your current billing option or to change from one billing opti
to the other, GO CHOICE.

OS-9 FTP SITE MOVES

The OS-9 only FTP cabrales.cs.wisc.edu has moved, the new site name is chestnut.cs.wisc.edu.

FATCAT LIST MOVES

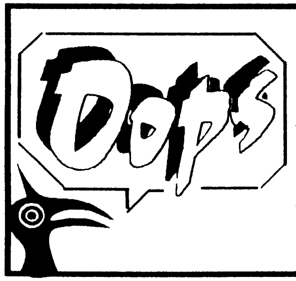
THE OS9 UNDERGROUND's FatCat email list has moved to a new address: The new address is: fatcat@zog.wa.com

To subscribe to this list, simply send email to listserv@zog.wa.com with: "subscribe fatcat your name" (should be your real name) in the body of the message.

This list is running on a TC70 under OS9/68K and is (I beleive) the first mail list to run under OS-9 or OSK. You may subscribe from any internet account or service that provides internet access (such as Delphi, GEnie, AOL or CIS).

Other lists available through zog are: "os9ers" and "sci_fi"

If you're software vendor or programmer and would like to submit your software hardware listed in New Discoveries, send us a brief discription of your product in care of this magazine or email to: DISCOVER@ZOG.WA.COM



Last month's New Discoveries column reported an erroneous phone number for information regarding the Pacific Northwest Fest, here is the correct number:

(206) 871-6535

We apologize for an inconveniences this may have caused.

MACRO SUBSTITUTION

A macro is a name that is associated with a text or numeric string. By convention, macros that represent constants should be in uppercase only. This makes them easy to spot in the source file listing and distinguishes them from variable names, which should be in lowercase. When the preprocessor is run, the actual text string constant is substituted for the macro name. This is known as "macro expansion".

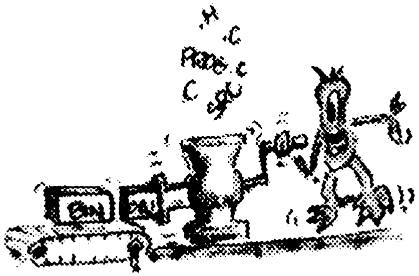
The advantage to macros is that they need only be defined once. Any change to the definition affects all occurrences. They execute more quickly than functions because there is no overhead involved in copying arguments and maintaining stackframes. When program speed is important, routines written in assembly are your best bet. However, if you don't want to learn assembly, then look for heavily used, small functions that can defined as a macro.

The disadvantages of macro substitutions are that they reevaluated at each mention of the macro name. Due to expansion, a program with many large macros will compile into a larger program than one with multiple calls to a single, large function.

It is important to note there is NO type-checking for macros and no storage is allocated for them, although the number of arguments are checked.

ALGORITHMS

An algorithm is a set of steps to to solve a particular problem in a finite number of steps. One could say that the art of programming is designing or choosing algorithms and expressing them in a well defined order of the particular programming language. The development process is very important and often overlooked by beginners and experienced programmers alike.



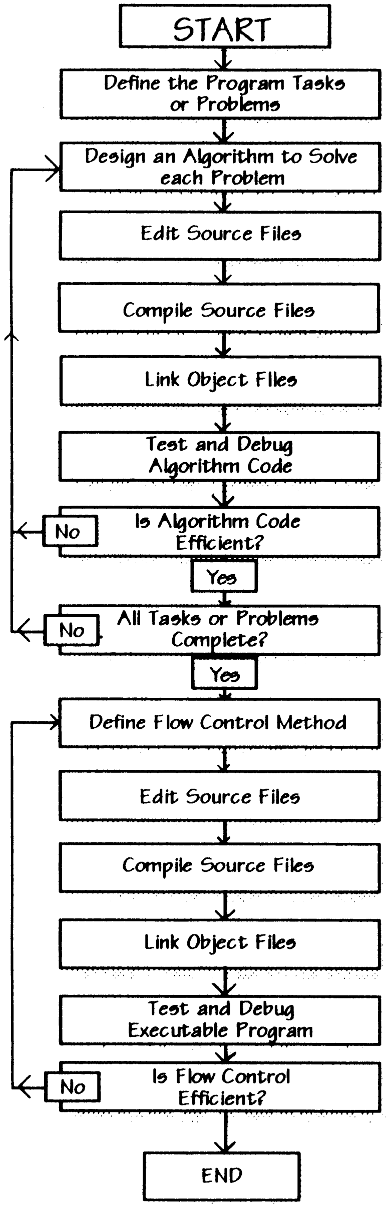
The source code can be spread out into multiple files, but the executable code should be be a single file for each task or problem. Not only does this make program maintenance simpler, it allows the creation of a concise "makefile".

At the end of the design stage, you should have a set of routines, or procedures, each of which solves a particular problem or step necessary in the complete program efficiently, or at least acceptable to the overall program performance, (speed, ease of use, etc.).

After creating and testing the object files by invoking the compiler, we need to combine the object files

Continue Page 59

The stages of program development should generally be as follows:


START
Define the Program Tasks or Problems
Design an Algorithm to Solve each Problem
Edit Source Files
Compile Source Files
Link Object Files
Test and Debug Algorithm Code
Is Algorithm Code Efficient?
No
Yes
All Tasks or Problems Complete?
No
Yes
Define Flow Control Method
Edit Source Files
Compile Source Files
Link Object Files
Test and Debug Executable Program
Is Flow Control Efficient?
No
END

RUNTIME LIBRARY

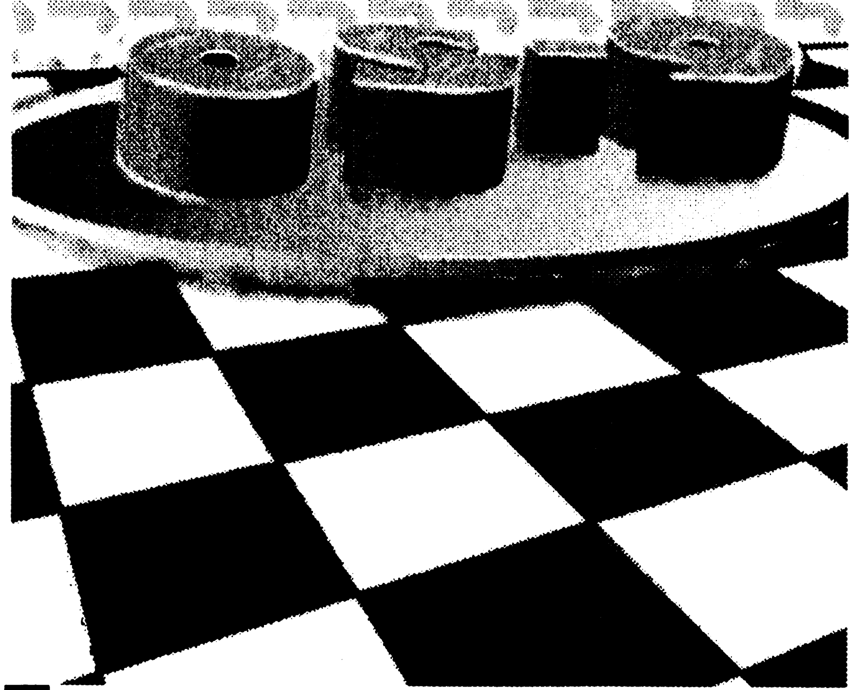
The C language defers many operations to a set of predefined functions sometimes known as the "Standard Library" or "Runtime library". This library consists of a collection of object files each of which perform one of many algorithms. These functions are divided into groups, such as input/output, memc management, mathematical operations, string manipulation, etc. Each should contain the information or definitions needed for the algorithms. By convention, these files end with the ".h" extension.

THE PREPROCESSOR

You should read the documentation for your particular compiler for the specifics on how to compile and link programs. The preprocess should be thought of as a separate program that runs before the compiler. While automatically run when a program is compiled, it has it's own simple syntax, distantly related to the C language.

All preprocessor directives begin with the pound sign, "#", and must be the first character on the line. Unlike C statements, a preprocessor directive ends with a newline, not a semi-colon and may appear anywhere in the source files.

The Future of the OS-9 Users Group



C SOFTWARE ENGINEERING 4

by
Leonard Cassady

PROGRAM DEVELOPMENT

Program development consists of a number of steps. Some will vary from system to system, however we'll attempt to briefly describe the general stages. The first step is to clearly define the problem and design "algorithms" or "functions" to solve it. The second step, determine the most efficient method of flow control. Program speed, system resources, and flexibility should all be considered.

InfoXpress is one of the mc flexible offline reader programs that I have seen.

In addition to mail and forum support InfoXpress comes with several script files set up to capture weather information and sto quotes from Compuserve.

The version of the softwa that I have is 1.1, a new versi 1.2 is due out soon. Then lat on a future upgrade, 2.0 will add support for GEnie, and other neat features such as data library support, which will mean the ability to capture the forum file cat logs, view them offline, and s lect which files you want download on the next onli pass.

Bill Dickhaus is open to suggestions and comments. He has put a lot of thought behind this program. It works so well th many people setup a CRON schedule program to automatica download messages and mail in the middle of the night so th can view their messages when they get up in the morning.

I highly recommend this program if you read forum me sages on either Del Compuserve or if you recei much email on a regular bas This program can save you much time online by automating t mail and message sessior InfoXpress could very well p for itself in the money you sa in forums. It frees up onli time that can be used on mo productive things online such as online conferences or searching the Internet on Delphi.

You can order InfoXpress from the Dirt Cheap Computer Stuff Co, 1368 Old Hwy. 50 East, Union, MO 63084. InfoXpress costs $69.95 for the OS-9/68k version or $49.95 for the OS-9/6809 version.

The OS9 Users Group...

A mover and shaker...or just a bowl of Jello?

Well, by this time, the news should be out. Carl Boll is the new president of the OS9 Users Group. What else has happened, I don't know, for Allen Huffman's 4th annual CoCoFest report contains all I know about the CoCofest this year.

That Carl Boll was elected, and Mark Griffith is on the Board of Directors is most of the news I have, and good news it is! What direction will the group take? Well, word on Delphi has it that support for the 6809 will be continued - though after the events of early last year Microware refuses to release the source to OS9/6809

However, while the 6809 version of the OS is the only version to boast it's own Vice-President, (OSK and OS9000 share one), support is going to increase more in the newer versions of the OS, and less in the 6809 area.

This is natural. For, as you Klingon fans out there know so well - "The Empire that does not grow - DIES!". It is time for OS9 users to start to grow.

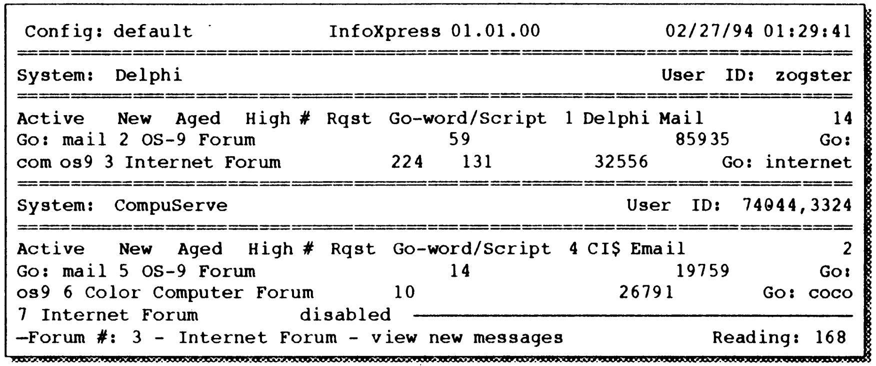
Where to Grow to?

Ask any farmer, he'll tell you, if you want a good crop, you choose one that fits the environment. The OS9 environment is largely industrial. Education provides less than 1% of OS9 sales.

Compuserve in addition to full mail support When setting up InfoXpress keep in mind that forums on Compuserve are extended services available for an hourly surcharge.

InfoXpress has several commands which are either commands to initiate an action online or offline.

The commands that initiate online actions allow you to only process replies and messages that you have posted earlier during an offline session (called requests) or allows you to go online and process requests and gather new mail and forum messages. You can disable/ enable any selected forum on the fly if you choose not to get messages for a particular forum. You can choose to go online Delphi or Compuserve or both services. When you choose an online command InfoXpress logs in, sends outgoing mail and captures any mail waiting to be read. Then InfoXpress goes from one forum to the next processing replies and new messages posting them directly to the forum. After posting messages all new messages are captured directly to InfoXpress message base.



```
 Config: default          InfoXpress 01.01.00         02/27/94 01:29:41
========================================================================
System:  Delphi                                          User  ID:  zogster
========================================================================
Active   New  Aged  High # Rqst  Go-word/Script  1 Delphi Mail        14
Go: mail 2 OS-9 Forum                 59                 85935       Go:
com os9 3 Internet Forum         224   131          32556     Go: internet
========================================================================
System:  CompuServe                                  User  ID:  74044,3324
========================================================================
Active   New  Aged  High # Rqst  Go-word/Script  4 CI$ Email           2
Go: mail 5 OS-9 Forum                 14                 19759       Go:
os9 6 Color Computer Forum       10              26791         Go: coco
7 Internet Forum         disabled ------------------------------------
-Forum #: 3 - Internet Forum - view new messages           Reading: 168
```

This process works unattended keeping time spent in forums to a minimum. It is really cool watching InfoXpress login and issue all the commands.

In addition to automated online sessions, InfoXpress has a manual online mode that will automatically login to Delphi or Compuserve and lets you select which forum to goto. You can then use InfoXpress as if it were a terminal program.

InfoXpress is also a really good offline reader program.

You can read messages and reply to them offline using your favorite text editor, or compose a new message offline. It works equally well for both mail and forum messages. InfoXpress keeps all messages in it's own message base that you can manage manually or have the system "age" messages that will be deleted from it's message base. You have options to archive all or just the messages to/ from you. When reading messages you can save interesting messages to text files. When you reply to a message InfoXpress gives the option of quoting the message.

Review:

Review by Jim Vestal

InfoXpress

by William Dickhaus

Back when I bought my first modem (a 300 baud manual modem) I had a Compuserve account, they were the only "affordable" pay bulletin board. Affordable meant $12.80 an hour, and at 300 bits per second you couldn't download much in a hour. But we enjoyed our hacker's lifestyle in spite of the monthly bill.

A few years later, a new service was offered called Delphi. Prices were even more affordable, a mere $6 per hour. Again most of us still had our trusty 300 ba modems, others have upgraded by this time to the new 1200 baud "standard".

Modems have gotten faster and prices for online services have become more competitive. We now have a greater choice of forums to choose from on both Delphi and Compuserve and with the advent of Internet mail, there is a need for an automated system for both forum and mail messages.

InfoXpress is the answer to automated forum messages and email for all OS-9 users.

Installation of the program is rather simple. I had to create a text file called default.ini which is the initialization control file (the key file), a sample default.ini is printed in the documentation and partial sample files are included on the distribution disk. The default.ini file is read when InfoXpress is first started up. This file contains the names of the forums that you desire to read on a regular basis for either Compuserve and Delphi or both. You can set up multiple *.ini files for just Delphi or just Compuser InfoXpress supports any readable forum on Delphi and

The personal market is even smaller. That is our present 'environment'. Can it be changed? Sure! But it will not change without careful regard to where we are, where we want to go, and how to get there.

The first thing the OS9 Users Group needs, is active participation from the entire spectrum of users. The way to get that is by providing support to all of the available market segments. So, expect to see efforts to recruit industrial users by providing service to them. Expect to see efforts to support small OS9 businesses, so that they can achieve greater penetration in the market place. Expect to see more support for users of personal systems running OSK and OS-9000.

Plant care, and fertilization....

The common factor in all market segments is SERVICE. A users group exists solely to provide services to a product's users.

A group is driven and fueled by it's membership. So, join your users group, and contribute to it!

What services does the User Group need to provide? Well, that is going to be up to the membership, and depends on a few things. One, is what the members want. Write to your officers FREQUENTLY. Tell them what you want

The Users group needs time and skills. Without skilled labor applied to the projects the membership wants to see done, nothing happens. Your officers can direct more projects than they can do themselves. So, it is up to the officers to seek volunteers, and those with the talent to accomplish things. Then, officers and members together must MAKE time to reach a successful project conclusion, or maintain an active program!

It is equally important to remember that each market segment, like different plants, is going to require different care and different formulas of fertilizer. This means that it will be up to the officers and board of directors to develop detailed programs to deal with the needs of each of OS9's unique market segments.

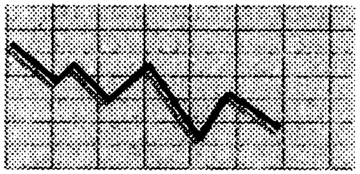
What plan?

Which programs?

What programs actually evolve to meet the needs of each area of the marketplace will depend on what the users in that segment see as their most pressing needs. But we can logically imply some needs from the current state of OS9. Both the industrial users and personal system developers need a means of locating capable OS9 programmers.

OS9 programmers need a convenient means of locating customers. That is a program that meets the needs of two birds with one stone! So, it would be a sma move to develop databases that contain information about OS companies and OS9 programmers, and provide assistance to and information about both.

I don't know that many industrial users, and won't presume to offer further comments on potentially useful programs at this time. However, I do know somethir about the needs of small businesses deriving their income from OS9, and I intend to propose the beginning of a rather comprehensive program to support them. Here goes.



Small Business and Programmers.

These days, with th economy stalled, large businesses cutting back, and uncertainty in the wind, small business has a window of opportunity. This is n strange. Small business often, if not always, does better when big business stalls. Business usually goes to the person willing to race, rather than wait on the starting line! But, still, there are plenty of businesses out there that can provide computer solutions.

And compatibility with DOS, DBASE, Lotus 123, CD-I, GIF, PCX, TIFF or ABC is very important in today's business environment. Standards are available. The Green Book, the White Book, ISO-9660, ANSI C, and so on, you can get them all - for a price. You are stuck with developing your own standard, and that shoots the benefits of compatibility out the window!

Another issue is, who has time to waste developing standards? OS9 needs applications. Period.

Now, the situation is better now than 2 years ago, don't get me wrong. TeX, Lout, GhostScript, CheckBook+, VED, VPRINT, Write Right!, SC, those are not insignificant improvements in available applications. But they are still a drop in the bucket compared to the competition. Can we fix this problem? Probably. Not to the point that we equal the number of programs that are available in the MS-DOS or Macintosh worlds, but certainly to the point that we can say we support all the major standards.

Raise the Standard!

How can we do this?? The plan is simple. First, we provide financing. This can be done by filling the need for a cheap and complete

```
19EA       end_rec:=end_rec+14
19F5      ENDIF
19F7      start_rec:=end_rec-13
1A02      IF start_rec<0 THEN
1A0E       start_rec:=0
1A15      ENDIF
1A17      screen_pos:=start_rec
1A1F      GOSUB 120
1A23     ENDIF
1A25     (*
1A28     (* Display the Last 14 Records
1A46     IF SUBSTR(key,"Uu")>0 AND
          control.num_recs>0 THEN
1A61      IF start_rec-14<0 THEN
1A70       start_rec:=0
1A77      ELSE
1A7B       start_rec:=start_rec-14
1A86      ENDIF
1A88      end_rec:=start_rec+13
1A93      IF end_rec>control.num_recs-1
           THEN
1AA6       end_rec:=control.num_recs-1
1AB4      ENDIF
1AB6      screen_pos:=start_rec
1ABE      GOSUB 120
1AC2     ENDIF
1AC4     (*
1AC7     (* Move to the Top of the File
1AE5     IF SUBSTR(key,"Tt")>0 AND
          control.num_recs>0 THEN
1B00      start_rec:=0
1B07      IF control.num_recs<14 THEN
1B16       end_rec:=control.num_recs-1
1B24      ELSE
1B28       end_rec:=13
1B2F      ENDIF
1B31      screen_pos:=start_rec
1B39      GOSUB 120
1B3D     ENDIF
1B3F     (*
1B42     (* Move to the Bottom of the File
1B63     IF SUBSTR(key,"Bb")>0 AND
          control.num_recs>0 THEN
1B7E      end_rec:=control.num_recs-1
1B8C      start_rec:=end_rec-13
1B97      IF start_rec<0 THEN
1BA3       start_rec:=0
1BAA      ENDIF
1BAC      screen_pos:=end_rec
1BB4      GOSUB 120
1BB8     ENDIF
3323     (*
3326     (* Quit Program
3335    EXITIF SUBSTR(key,"Qq")>0 THEN
3346    ENDEXIT
334A     GOSUB 120
334E    ENDLOOP
3352    (*
3355    (* End of Program
3366 60  IF file_open THEN
3372     CLOSE #file
3378    ENDIF
337A    GOSUB 110
337E    er:=ERR
3384    IF er>0 THEN
3390     PRINT
3392     PRINT "ERROR #"; er
33A1    ENDIF
33A3    END
3BD2    (*
3BD5    (* The remarked REPEAT loop is
         the INKEY routine for OSK.
3C0E    (* Unremark it and remark the RUN
         Inkey statement if using OSK.
3C4D 100  REPEAT
3C52     (* REPEAT
3C5B     (* count:=INKEY(#0)
3C6E     (* UNTIL count>0
3C7E     (* GET #0,key
3C8B     RUN Inkey(key)
3C95    UNTIL
         SUBSTR(key,"YyNn"+CHR$(13))>1
3CAC    RETURN
3CAE    (* Clear Screen
3CBD    (* Unremark the ansi version and remark
3CEE    (* the other if using an ansi terminal.
3D0B 110  (* PUT #1,ansi_cls
3D20    PUT #1,cls
3D29    RETURN
3D2B    (* Home Cursor
3D39    (* Unremark the ansi version and remark
3D6A    (* the other if using an ansi terminal.
3D87 120  (* PUT #1,ansi_home
3D9D    PUT #1,home
3DA6    RETURN
```

(EOF)

Moving?

If you are planning to change your address, you must let us know at least 30 days in advance in writing. Send a postcard or letter to **Fat Cat Publications** with your current address *and* your new address so we can update your records and keep your subscription uninterupted.

*We are **not** responsible for your subscription, if we are not notified in time.*

Fat Cat Publications
4650 Cahuenga Blvd., Ste #7
Toluca Lake, CA 91602

ANSI compatible C compiler.We have the compiler, but the library that makes GCC complete is just not there. By porting the GNU C library as a cooperative venture, we give the OS9 User Group an asset that can be distributed in return for a reasonable fee.

long code = CASH$;

Using those fees, the User Group can purchase other standards, and those standards can be used by member companies and programmers to develop libraries that will speed application development. By making the development of these libraries cooperative ventures with the rights vested in the User Group, we can shorten development cycles significantly, and greatly increase the number of types of MS-DOS and Macintosh files usable on OS9 computers.

Who cares if the file is DBASE IV type - the program needn't be a direct clone, and once you have file compatibility, the user interface becomes the important factor in selling a program. So, programmers can afford to share development of the initial code base, get an application to market, and then optimize the base code as a competitive measure later, if necessary. This will substantially shorten the length of the development cycle. It means more applications, in shorter time!

The Standards and Development Organization would be responsible for collecting data indicating which standards would 1) be most in demand, and 2) be most cost effective for the User Group to purchase, given fund availability, and purchasing said standards.

The OS9 User Group Standards Committee would further be responsible for or promoting representation for, OS9 at various international standards meetings that conveys a consensus of the community on that particular matter. For example, ANSI has committees writing standards on HyperMedia, MIDI, and other subjects that effect OS9 users. The committee would provide OS9 using members of such committees with a channel for communication with the general body of OS9 users. Basically, we would begin by dedicating the efforts of one or more members of the committee to channeling comments from the OS9 community to the ISO or ANSI committee member. This could be done by establishing a directory of such members, which could be sold at cost. We could also publish reports of those committee meetings in MOTD. Readers of the C Users Journal should be familiar with the value of such reports from reading PJ Plauger's monthly column on the ANSI C Standard.

Another function of this committee would be to certify programs as meeting or not meeting new standards. For example, Jc Hegberg of SubEtha Software has proposed releasing the format of the Printer Object Files used by his

```
0AF3      IF entry.check THEN
0AFF        PRINT USING "s5>"
             ,entry.number;
0B0F      ENDIF
0B11      PRINT TAB(8); entry.date;
0B1E      PRINT TAB(17); entry.desc;
0B2B      IF entry.clear THEN
0B37        PRINT TAB(46); "*";
0B41      ENDIF
0B43      IF entry.deductable THEN
0B4F        PRINT TAB(48); "*";
0B59      ENDIF
0B5B      IF entry.amount>.0 THEN
0B6E        IF NOT(entry.deposit) THEN
0B7B          PRINT TAB(50);
0B81          PRINT USING "r9.2>"
               ,entry.amount;
0B93        ELSE
0B97          PRINT TAB(60);
0B9D          PRINT USING "r9.2>"
               ,entry.amount;
0BAF        ENDIF
0BB1      ENDIF
0BB3      PRINT TAB(70);
0BB9      PRINT USING "r10.2>"
         ,entry.balance
0BCB    NEXT counter
0BD6  ENDIF
0BD8  (*
0BDB  (* Main input routine
0BF0  REPEAT
0BF2    (* REPEAT
0BFB    (* count:=INKEY(#0)
0C0E    (* UNTIL count>0
0C1E    (* GET #0,key
0C2B    RUN Inkey(key)
0C35  UNTIL SUBSTR(key,
   "AaBbDdEeFfHhIiNnPpQqRrSsTtUuYy")>1
17C2  (*
17C5  (* Find Record
17D3  IF SUBSTR(key,"Ff")>0 AND
       control.num_recs>0 THEN
17EE    GOSUB 110
17F2    PRINT
17F4    (*
17F7 50    INPUT "Record # to Find: "
         ,rec_num
```

```
1814      er:=ERR
181A      IF rec_num<0 OR rec_num
           >control.num_recs THEN 50
1834      start_rec:=rec_num-1
183F      IF start_rec+14
           >control.num_recs THEN
1852        end_rec:=control.num_recs-1
1860        IF start_rec>end_rec-13 THEN
1870          start_rec:=end_rec-13
187B          IF start_rec<0 THEN
1887            start_rec:=0
188E          ENDIF
1890        ENDIF
1892      ELSE
1896        end_rec:=start_rec+13
18A1      ENDIF
18A3      screen_pos:=rec_num-1
18AE  .   (* Home Cursor Routine
18C4      GOSUB 120
18C8    ENDIF
18CA    (*
18CD    (* Move Pointer to Next Record
18EB    IF SUBSTR(key,"Nn")>0 AND
         control.num_recs>0 THEN
1906      IF screen_pos<end_rec THEN
1913        screen_pos:=screen_pos+1
191E      ENDIF
1920      GOSUB 120
1924    ENDIF
1926    (*
1929    (* Move Pointer to Previous
         Record
194B    IF SUBSTR(key,"Pp")>0 AND
         control.num_recs>0 THEN
1966      IF screen_pos>start_rec THEN
1973        screen_pos:=screen_pos-1
197E      ENDIF
1980      GOSUB 120
1984    ENDIF
1986    (*
1989    (* Display the Next 14 Records
19A7    IF SUBSTR(key,"Dd")>0 AND
         control.num_recs>0 THEN
19C2      IF end_rec+14
           >control.num_recs-1 THEN
19D8        end_rec:=control.num_recs-1
19E6      ELSE
```

```
0623      PUT #file,control
062D      SEEK #file,0
0636     ENDIF
0638   ENDIF
063A   (*
063D   (* Set starting values for
        records displayed
0669   ON ERROR GOTO 60
066F   GET #file,control
0679   IF control.num_recs>13 THEN
0688     start_rec:=control.num_recs-14
0696     end_rec:=control.num_recs-1
06A4   ELSE
06A8     IF control.num_recs=0 THEN
06B7       start_rec:=0
06BE       end_rec:=0
06C5     ELSE
06C9       start_rec:=0
06D0       end_rec:=control.num_recs-1
06DE     ENDIF
06E0   ENDIF
06E2   screen_pos:=end_rec
06EA   GOSUB 110
06EE   (*
06F1   (* Main LOOP
06FD   LOOP
06FF     (*
0702     (* Display the upper screen
071D     PRINT "  Check Register for ";
          control.year; "     [A]dd
           [E]dit  [D]own  New [Y]ear
           [U]p  [Q]uit"
0775     PRINT TAB(32); "[T]op  [F]ind
          [N]ext  [B]ottom
          [P]revious"
07AA     PRINT TAB(32); "Reconcile
          [S]tatement  [I]nsert
          [R]emove"
07DD     PRINT "Total Entries  Display
          Entries"; TAB(42); "[H]ard
          Copy"; TAB(66); " Start
          Balance"
0826     PRINT "=============
          ===============    C=Cleared
          T=Tax Deductable
          ============="
```

```
0879     PRINT USING "i13>,x2,i6>,' -'
          ,i6<,x5,'Current Entry =',i5<
          ,x15,r10.2>",control.num_recs
          ,start_rec+1,end_rec+1
          ,screen_pos+1
          ,control.start_balance
08D9     PRINT
08DB     PRINT "  Num  Date
          Description              C
          T Withdrawl Deposit   Balance"
092B     PRINT " ===== ========

  =========================== =
          = ========= =========
          =========="
097E     (*
0981     (* Display the lower screen
          (records)
09A6     (* If no records, display the
          sample record
09D1     IF control.num_recs=0 THEN
09E0       PRINT "*";
09E6       PRINT USING "s5>","0";
09F3       PRINT TAB(8); "11/21/93";
0A04       PRINT TAB(17); "Sample";
0A13       PRINT TAB(46); "*";
0A1D       PRINT TAB(48); "*";
0A27       PRINT TAB(50);
0A2D       PRINT USING "r9.2>",.0;
0A3F       PRINT TAB(60);
0A45       PRINT USING "r9.2>",.0;
0A57       PRINT TAB(70);
0A5D       PRINT USING "r10.2>",.0
0A6F     ELSE
0A73       (*
0A76       (* Otherwise, display the
            designated records
0AA2       FOR counter:=start_rec TO
            end_rec
0AB4         SEEK #file,counter*SIZE
              (entry)+SIZE(control)
0ACA         GET #file,entry
0AD4         IF counter=screen_pos THEN
0AE1           PRINT "*";
0AE7         ELSE
0AEB           PRINT " ";
0AF1         ENDIF
```

Write Right! word processor.

Suppose Joel submitted that format to the OS9 UG as a standard, with copies of a variety of printer object files, and that a developer wished to use that format in his new Data Right! DBM.

Joe Programmer could send his product to the committee, t committee members would test his program using available object files, and issue a certificate of compliance, for a fee. This nominal fee would finance committee activities, the certificate would provide the developer with a marketing tool, and the OS9 consumer with assurance that they were receiving a product that would be compatible with files currently in use on their systems, reducing system clutter and enhancing maximum use of the printers capabilities. This committee would also serve as caretaker for imported standards such as termcap and curses.

Obviously, this is a heavy undertaking. But, with participatio from the community, users an developers alike would benefit

This editorial is an invitation. I'm inviting all OS9 users, vendors, and developers to write to me. Share your opinion on this matter. Volunteer for whatever you think you can do. As I write, I'm looking at a message just received from Carl Boll. Carl has asked tha develop a plan for such a standards organization. He is interested in seeing a list of people who will serve on such a committee, a goals list and a time table for goal achievement. When he has these items, he will take them to the Users Group Board of Directors for approval. So get writing! This invitation specificly includes ALL vendors. I'm looking forward to working with those who have contributed in this regard in the past, and building on their work, for a stronger future for OS9!

-David M. Graham

Comments to David regarding this article may be sent to:
BlackHawk Enterprises, Inc.
P.O. Box 10552
Enid, OK 73706-05552
Internet : nimitz@delphi.com
Delphi :(NIMITZ), or c/o this
magazine (see page 65 of this issue)

(EOF)

```
016E  (*
0171  (* File Control and Record Entry
       Strutures
019B  TYPE
CTRL=num_recs,year:INTEGER;
       last_year:BOOLEAN; start_balance
01B4  TYPE REC=number:STRING[5]; date
       :STRING[8]; desc:STRING[28];
       check,deposit,clear,deductable
       :BOOLEAN; amount,balance
01F4  DIM control,lcontrol:CTRL; entry
       ,last,new:REC
0211  (*
0214  DIM file,path,form_feed:BYTE
0223  (*
0226  (* Clear Screen and Home Cursor
       Variables
024F  (* Remark this statement if using
       ansi codes
027B  DIM cls,home:BYTE
0286  (*
0289  DIM er,count,lcount,counter
       ,start_rec,end_rec,screen_pos
       ,rec_num:INTEGER
02AC  DIM temp_amount,temp_withdrawls
       ,temp_deposits
02B9  DIM file_open,found,no_checks
       ,no_withdrawls,no_deposits
       :BOOLEAN
02D0  DIM char,key:STRING[1]
02E0  DIM temp_rec:STRING[4]
02EC  DIM temp_num:STRING[5]
02F8  DIM temp_date:STRING[8]
0304  DIM temp_desc:STRING[28]
0310  DIM filename:STRING
0317  (*
031A  (* Ansi Structure Assignments
0337  (* ansi_cls.escc:=$1B
034C  (* ansi_cls.cmdc1:=$5B
0362  (* ansi_cls.codc:=ASC("2")
037C  (* ansi_cls.cmdc2:=ASC("J")
0397  (* ansi_home.esch:=$1B
03AD  (* ansi_home.cmdh1:=$5B
03C4  (* ansi_home.codh1:=ASC("0")
03E0  (* ansi_home.cmdh2:=ASC(";")
03FC  (* ansi_home.codh2:=ASC("0")
0418  (* ansi_home.cmdh3:=ASC("H")
```

```
0434  (*
0437  (* TC-70 Clear-Screen and
       Home-Cursor Codes
0462  (* cls:=$1A
046D  (* home:=$1D
0479  (*
047C  (* CoCo3 and MM/1 Clear-Screen
       and Home-Cursor Codes
04B0  cls:=12
04B7  home:=1
04BE  (*
04C1  form_feed:=12
04C8  control.num_recs:=0
04D3  control.year:=1990
04DF  control.last_year:=FALSE
04E9  control.start_balance:=0
04F5  file_open:=FALSE
04FB  key:=" "
0503  filename:="checks.dat"
0514  (*
0517  (* Open 'checks.dat' file
0530  ON ERROR GOTO 10
0536  OPEN #file,filename
0540  file_open:=TRUE
0546  (*
0549  (* Create file if it doesn't
       exist
056B  IF NOT(file_open) THEN
0575 10   er:=ERR
057E    IF er=216 THEN
058A      CREATE #file,filename
0594      file_open:=TRUE
059A      (* Clear Screen Routine
05B1      GOSUB 110
05B5      PRINT
05B7      PRINT "Current Year: ";control.year
05D0      PRINT
05D2      PRINT "Change? (y/N) ";
05E5      GOSUB 100
05E9      PRINT
05EB      IF key="Y" OR key="y" THEN
0600        PRINT
0602        ON ERROR
0605        INPUT "New Year: "
             ,control.year
061B        er:=ERR
0621      ENDIF
```

Tandy's Little Wonder

the most complete reference ever written for the Color Computer!

This 140 page softbound book contains:

- History of the CoCo
- Club and BBS Listings
- Current Supporting Vendors
- Peripheral Details
- Operating System Descriptions
- Programming Languages
- Repair/Upgrade/Modification Procedures
- Schematics (reprinted w/permission of Tandy)
- MUCH, MUCH MORE!

ONLY $25 (+ $2.50 S&H)

(Canadians add $2 for air mail, overseas add $4)

Introducing a NEW MAGAZINE for CoCo/OS-9/OSK users:

the world of
68' micros
Tandy Color Computer, OS-9, OSK

Where does one now go for CoCo support since "the Rainbow" ceased publication? "the world of 68' micros is dedicated to producing a quality publication supporting the CoCo, Disk BASIC, 6809/OS-9, and even OSK (OS-9 /68000)! Top writers and articles will be featured, including a hardware column by the infamous **Dr. Marty Goodman**. Upcoming features will include:

Repackaging the CoCo (even a transportable!)
C Programming for Beginners
Beginning OS-9... from the box!
CoCoFest Reports... FOUR this year!
MicroNews... new products and information (w/ photo of the B&B "Rocket")
Swap Shop... classified ads! (Subscribers only, buy,sell trade... even software!)

Subscriptions are **$23/year for 8 issues** (every 6 weeks), or **$12 for a 4 issue trial subscription** ($30/$16 for Canada, $33/$17 overseas). A disk service, **"microdisk"**, is **$40/year or $6 per issue** ($44/$7 Canada, $54/$8 overseas). **First issue will be delivered in August...** ***DON'T MISS IT !***

FARNA Systems PB
P.O. Box 321
Warner Robins, GA 31099-0321
Phone 912-328-7859

Sub-Etha Software

MultiBoot by Terry Todd & Allen Huffman

Now have up to SIXTEEN bootfiles on your startup disk!

Hot of the assemblers and compilers is a great must-have utility which lets you have up to 16 bootfiles on one disk! No more boot disk floppy-swapping! MultiBoot will install itself to a cobbled boot disk and, upon typing "DOS", will greet you with a scrolling menu of available bootfiles!

OS-9 Req: CoCo 3, OS-9 Level $19.95

Towel by Allen C. Huffman

The first EthaWin program - a disk utility for OS-9.

A program no intergalactic hitchhiker should be without! Use a mouse or keyboard hot-keys to perform common file and disk commands from pull-down menus. Tag multiple files for Delete, Copy, Rename, etc., and even have point 'n click disk Backup, Cobbler, Dcheck and other commands. User menu lets you specify up to seven of your own commands to execute. Runs under the EthaWin interface on a high-speed text screen. All commands/colors configurable.

OS-9 Req: CoCo 3, OS-9 Level 2 .... $19.95
OS/K Req: MM/1 or K-Windows Compatible..........................$24.95

1992 CoCoFest SIMULATOR by Allen C. Huffman

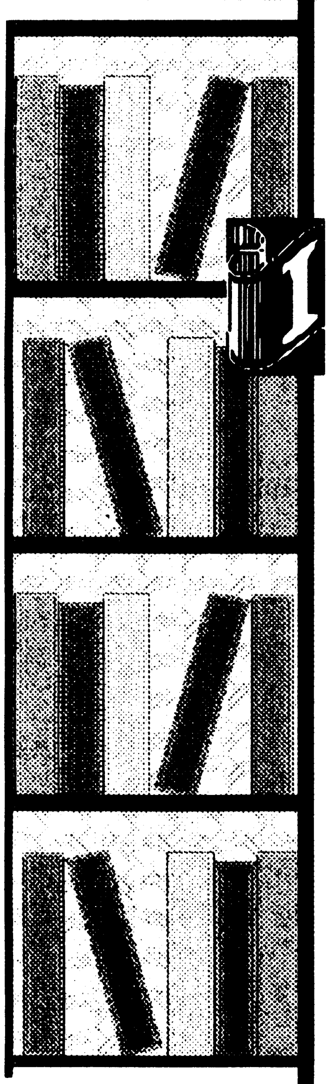
Graphics "adventure" based on the 1992 Atlanta CoCoFest

The next best thing to having been there! Digitized graphics of the event and a text command parser (ie, "get the box of disks") let you see all the vendors and even run into some famous faces of the CoCo Community. The show area, seminar room, and portions of the hotel are all represented. No true "goal", but you do have to figure some things out, like how to get into the show and how to buy items from vendors. Runs on a 640x192 hi-res graphics screen.

OS-9 Req: 512K CoCo 3, OS-9 Level 2, 490K Disk Space $9.95
OS/K Req: MM/1 or 100\% K-Windows Compatible $14.95

Send US funds plus $2.50 shipping to:
Sub-Etha Software
P.O. Box 152442
Lufkin, TX 75915
(815) 748-6638
More items available! Contact us for a complete product listing!

Building Your Own C Libraries - Part 3



by Bob van der Poel

In the last two installments of this column we have been discussing the C library. In part 1 we discussed the purpose of a libiary and how one is created; in part 2 we got into some of the details of a makefile for creating a library and showed some sample modules This month we will continue with more modules you can add to your own libraries.

The first module for this month is replacement for the memchr() function in Microware's 68K library. From a quick disassembly, it appears that Microware's version was coded in C. Our assembler replacement is about three times faster than the original; plus it is a few bytes shorter.

You might be wondering about how to go about using a function with the same name as an existing one. It is really quite

display 14 entries from and including the current entry. If there are less than 14 entries in the register, the screen remains unaffected. The pointer is placed by the first entry displayed. If there are less than 14 entries from the current pointer position, the first 14 entries will be displayed.

Down : Selecting this function will move the entry pointer 14 entries towards the bottom of the register from the current pointer position and display 14 entries from and including the current entry. If there are less than 14 entries in the register, the screen remains unaffected. The pointer is placed by the last entry displayed. If there are less than 14 entries from the current pointer position, the last 14 entries will be displayed.

Next : Selecting this function will move the entry pointer to the next entry. If the current entry is the last one displayed, the pointer remains where it is.

Previous : Selecting this function will move the entry pointer to the previous entry. If the current entry is the first one displayed, the pointer remains where it is.

Find : Selecting this function allows you to find a specified entry number. You will be prompted to enter an entry number to find. When found, the screen will be updated and the pointer placed by the specified entry. This function requires a numeric in put, and you must enter an entry number once you have selected it.

A Note About The Entry Pointer:

The entry pointer is directly tied to the 'Current Entry – ‹num›' area of the upper display area. Since there was no convenient way to place an entry number in the entry display area, I decided it would be best to do it this way. As the entry pointer is moved, the 'Current Entry – ‹num›' area is updated to reflect the current entry number. If you aren't sure what the entry number is of the entry you wish to deal with, simply follow these steps:

1. **Look at the current entry number.**
2. **Estimate how many entries away the desired entry is from the current entry.**
3. **Find the entry number you have estimated.**
4. **Repeat these steps.**

Also, the 'Displayed Entries' area of the upper display is updated each time you move Up, Down, Top or Bottom, or Find an entry. This field specifies which entries are currently displayed. You can use this field to determine if your estimate is within the range of the displayed entries.

This concludes part one of the checks program. In part two, we cover entry addition, editing, insertion, and removal. See you next time.

Code to part one:

```
PROCEDURE checks
0000  (* Checks - Check Register
       Program
0022  (* Copyright (c) 1993 by Wayne
       Campbell
0049  (*
004C  (* Ansi Clear-Screen and Ansi
       Home-Cursor Structures
0080  (* If you wish to use these,
       un-remark them and remark the
00BA  (* cls and home variable
       assignments designated below
00EF  (* TYPE ANSC=escc,cmdc1,codc
       ,cmdc2:BYTE
0116  (* TYPE ANSH=esch,cmdh1,codh1
       ,cmdh2,codh2,cmdh3:BYTE
014A  (* DIM ansi_cls:ANSC; ansi_home
       :ANSH
```

you can remark this DIM statement, as these variables will not be used.

After the dimensioning statements are the initial assignment statements. The first ones are remarked and assign the values of the ANSI clear screen and home cursor data types. Un-remark them if you wish to use the ANSI codes.

Next are the remarked codes for the TC-70. After them are the assigments for cls and home. Remark these assignments. Now, change all of the statements that begin with 'PUT #1' to read '(* PUT #1', making them remark statements. Next, change all of the statements that begin with '(* PUT #1,ansi' to read 'PUT #1,ansi' to remove the remarking.

You are now ready to use ANSI-codes to clear the screen and home the cursor.

Using other terminal codes:

If the terminal you are using doesn't work with any of the above codes, you will have to modify the program to account for them. Checks uses the PUT statement to clear the screen or home the cursor. The steps you need to take are:

First, determine if the terminal you have uses a single character code for clear screen and/or home cursor. If it does, the changes will be easy. Just remark the assignments for 'cls' and 'home', and add new assignments that reflect the single character code (a number, NOT a text character) for your terminal. If the value of the code is greater than a BYTE variable can store, find the DIM statement that dimensions 'cls' and 'home' and change the type from BYTE to INTEGER.

If your terminal requires a series of codes (more than two bytes), you'll have to create a new data type using the TYPE and DIM statements, and then assign the types the correct values. Then you'll have to change all of the PUT #1 references to reflect the new data type.

Study the ANSI data types for a better understanding of their use, and this should guide you to creating your own data types.

The functions included in this segment of the code are:

[T]op [P]revious [U]p [F]ind
[B]ottom [N]ext [D]own

The following is a discussion of these functions.

Top : Selecting this function will move the pointer to the top of the register and display the first 14 entries in the register. If there are less than 14 entries in the register, all of the current entries are displayed. (This means that you won't see any change in the display.) The pointer is placed by the first entry.

Bottom : Selecting this function will move the pointer to the bottom of the register and display the last 14 entries in the register. If there are less than 14 entries in the register, all of the current entries are displayed. (This means that you won't see any change in the display.) The pointer is placed by the last entry.

Up : Selecting this function will move the entry pointer 14 entries towards the top of the register from the current pointer position and

simple–when you link the program you use a line like:

```
cc program.c -l=/dd/lib/bvlib.l
```

Assuming that the new function is in bvlib.l the replacement will be used. When the main library (/dd/lib/clib.l) is processed the memchr function in it will be ignored since it has already been found in bvlib.l. Note that any functions in clib.l which call memchr (there aren't any...) would use the new function, too!

Our code for memchr() follows:

```
/* memchr.c

This duplicates the MW version of the
same function. However, this is a tad
shorter and faster. (I think the MW version
uses a cast in the compare....)

This started out as the following C
program. However, it's even faster
when the assembler source has been
hand optimized....

   memchr(p, value, count)
   register u_char *p;
   int value;
   register int count;
   {
      register u_char c=value;

      while(count–)
      {
         if(c == *p++) return –p;
      }
      return 0;
   }
 */
#asm
memchr:
 movem.l d1/a2,-(sp)
 movea.l d0,a2        move start ptr to a2
 move.l 12(sp),d0      count to d0
 beq.b notfound        exit if count==0
 subq #1,d0          adjust count for dbeq (exit==-1)

loop
 cmp.b (a2)+,d1       do we have a match?
 dbeq d0,loop

beq.b found  exit if match (DBxx does not affect CC!)

 addq.w #1,d0        now check to see if counter
 subq.l #1,d0   is fully decremented...this is needed
 bcc loop           since DBxx only works on 16 bits

notfound
 moveq.l #0,d0        set return reg. to 0
 bra.b done

found
 subq.l #1,a2       dec ptr back to match or end pos
 move.l a2,d0         set return register to match pos

done
 movem.l (sp)+,d1/a2
 rts

#endasm
```

You might notice a few things about the above source: First, I wrote the code first in C and then hand optimized it. I left the original C code in the source file as a comment. Second, I have used the DBeq 68xxx instruction. This is a bit tricky to use (so be careful), but it is very fast. When I first wrote the function I didn't use DBeq–the latest version is about a third faster. Third, I have again set this up as a C file and used the #asm directive to take care of generating the necessary psects, etc.

The next function converts a string to all lower case letters. Conversions of this type are simple to handle in C since all strings end in a 0 byte....well, they should!

```
/* lowstr.c

  Convert string to all lowercase
  Usage: tolower(string)
*/

#define u_char unsigned char

lowstr(s)
register char *s;
{
  register u_char c;

  while(c=(u_char)*s++)
  {
     c-='A';
     if(c<='Z'-'A') *(s-1)+=('a'-'A');
  }
}
```

If you are new to C you might be confused by some of this code.... First, notice that we use a cast to convert the char in the string (range -128 to 127) to an unsigned char (range 0 to 255). Second, we do the assignment to 'c' in the while() loop–when a 0 is assigned the loop is exited. Third, notice how the comparison to determine if the character to convert is a uppercase letter to start.

This function relies on two important conditions (which makes it not at all portable)–it requires a unsigned char type (which the 6809 compiler does not have) and it assumes that the ASCII character set is being used. An argument could easily be made that another way of doing this would be better. On the other hand, this code is quite a bit faster than the alternates. Be aware!

The final function for this installment is the companion lowstr(). Upstr() just converts a string to uppercase. The same warnings apply to it as lowstr().

```
/* upstr.c

  Convert string to all uppercase

  usage: upstr(string)
*/

#define u_char unsigned char

upstr(s)
register char *s;
{
  register u_char c;

  while(c=(u_char)*s++){
     c-='a';
     if(c<='z'-'a') *(s-1)-=('a'-'A');
  }
}
```

In the next issue we'll present a few more functions for your libraries. Users of 6809 versions of OS-9– don't get discouraged, the code in the next issue will work for all C compilers. Keep those cards and letters coming!

-Bob van der Poel

You can write me at:
PO Box 355, Porthill, ID 83853 or
PO Box 57, Wynndel, BC, Canada
V0B 2N0 or, electronically,
Compuserve 76510,2203.
(or in c/o this magazine, see pg 68)

Inkey: The inkey routine is devised to allow you to run this program under OS9 Level 1 & 2, and OS9/68000. Since the INKEY function in Microware Basic is different from the Inkey sub-routine in Basic09, the syntaxs are therefore different, and must be dealt with differently.

The GOSUB subroutine for Inkey is for getting a yes/no input. There are other inkey functions in the code that are separate from this suboutine. In all cases, the routine is set for Basic09. To use the OSK version, you'll have to make some changes to the code. The routines follow the format:

```
REPEAT
 (* REPEAT
 (* count:=INKEY(#0)
 (* UNTIL count>0
 (* GET #0,key
 RUN Inkey(key)
UNTIL SUBSTR(key," YyNn"+CHR$(13))>1
```

The last UNTIL statement reflects the GOSUB subroutine. The differences with the other uses in the code are basically in that UNTIL statement, except for one, which is identical to the GOSUB except for the variable used to get the character ('char' instead of 'key').

To use the OSK version of Inkey, change all of the RUN Inkey statements to remark lines, and change all of the remarked REPEAT loops and GET statements to valid statements (non-remarked). NOTE: Attempting to use this function under OS9 Level 1 & 2 will result in an error.

```
REPEAT
 REPEAT
  count:=INKEY(#0)
 UNTIL count>0
 GET #0,key
 (* RUN Inkey(key)
UNTIL SUBSTR(key," YyNn"+CHR$(13))>1
```

Screen Clearing and Cursor Homing: The screen clearing and cursor homing subroutines are setup to allow you to select the code you wish to use. Internally, you may choose between OS9 Level 1 & 2 (ASCII code 12, which I think is also valid on the MM/1), the TC-70 (by changing an assignment statement), or ANSI-code for terminals that support those codes.

Setting up the TC-70 codes:

Find the assignments at the beginning of the program that set the variables 'cls' and 'home' for the TC-70 and un-remark them. Then remark the 'cls' and 'home' assignments that setup for OS9 Level 1 & 2.

Setting up the ANSI-codes:

At the beginning of the program are all of the dimensioning statements used in the program. The first three statements are TYPE statements and a DIM statement for setting up the ANSI codes. They are remarked, so you'll have to unremark them to use them. There is a DIM statement furthur down that dimensions the variables 'cls' and 'home'. If you use the ANSI codes, no set format for entry of the date, but there is an 8 character field size limitation. I use the format MM/DD/YY in my register (which means that 1/2/93 would be entered as 01/02/93). You may use whatever format suits you, but remember that the format you use in this field must be the same as the format you use when reconciling statements. (More on this later)

Description: This field holds the information concerning the nature of the transaction. (ie. if a check, the entity to whom the check was written). This field holds a maximum of 28 characters.

Cleared (C): When a statement is reconciled, this column provides a visual aid to determine which entries were cleared, denoted by an asterick (*).

Tax Deductable (T): Most registers have a single column that may be used to denote cleared entries, tax deductable entries, or whatever you wish to use it for. I decided to create a separate column for tax deductable entries so you could tell which items to include in your deductions each year. "Checks" DOES NOT do anything with this information. It is for your information only. Entries with this set show an asterick (*).

Withdrawl/Deposit: These columns hold the amount of the transaction. Placement of the data is automatic. (ie. deposits always show up in the Deposit column)

Balance: This column reflects the current balance after accounting for the transaction. Deposits are added to the bal withdrawls and checks are deducted from the balance.

The available functions listed in the upper portion of the display are:

[T]op : Move to the top of the file and display the first 14 entries.

[B]ottom : Move to the bottom of the file and display the last 14 entries.

[N]ext : Move the pointer to the next entry.

[P]revious : Move the pointer to the previous entry.

[U]p : Move the pointer 14 entries toward the top of the file and display the entries.

[D]own : Move the pointer 14 entries toward the bottom of the file and display the entries.

[F]ind : Find the specified entry.

[A]dd : Add an entry to the register.

[E]dit : Edit the specified entry.

[I]nsert : Insert an entry at the specified position.

[R]emove : Remove (Delete) the specified entry.

Reconcile [S]tatement : Reconcile a Bank Statement.

[H]ard Copy : Print a copy of the current register.

New [Y]ear : Create a new register.

[Q]uit : Quit checks.

Before getting into the functions included in this segme of the source code, I feel it necessary to cover the 3 sub-fur tions included in this part. They are the inkey routine, the screen clearing and cursor homing routines.

WREAD

BY WAYNE CAMPBELL

If you're like me, and you don't have a working printer, then the only way to read a text file is to use the following command sequence:

tmode pause; list ‹file›; tmode -pause

or list the file and [CTRL]-[W] to pause the listing here and there. Not only is this a pain to work with, it also means that if you want to re-read the file you have to start at the beginning all over again. You can't just go backwards.

I decided to write a routine to let me read a file, mainly because I grew weary of re-listing 40K testfiles over and over to check them over. I named the routine WRead for two reasons. First, I liked the idea of typing 'read filename' because it's syntax is closer to what I am doing with the file. Second, I added the 'W' to the name to avoid conflict with other routines. WRead is pronounced 'read', just as Wren is pronounced 'ren'.

WRead has built-in help, so you can find out the available keys by entering the command with no options. However, WRead is not a completely efficient routine either. WRead is minimally error trapped, basically just enough to catch a bad file/pathname.

It could use improvements also. I basically wrote it to handle the textfiles I need to read. If you WRead a file that has a bad sector, WRead WON'T trap it. WRead turns the echo off, so if an error occurs during a WRead of a file, you'll have to turn the echo back on manually (*tmode echo*).

The text format is 80 characters per record, including carriage return. WRead actually reads

79 characters into a string, and the carriage return is the end-of-line. WRead will read a file with a maximum of 4,096 lines. The reason for this is that a sequential text file can only be treated like a random access file if you first find out where each line begins in the file, from top to bottom. In WRead this is done by an initial p through the file, storing all of the line start positions in an array of REAL variables.

NOTE: The REAL is used instead of the INTEGER because a file greater than 32,767 bytes could not be read with an INTEGER. This is due to the wrap to a negative value that occurs with an INTEGER.

Once these positions stored, WRead then clears the screen and lists the first screen of lines (set to 24 lines; if you have a 25 line grfdrv module, you'll have to modify WRead to use the 25th line). The cursor is placed at the beginning of the bottom line of the screen. Then you're choices are:

WRead was developed on a Color Computer 3 running OS-9 level 2. It uses the window cursor codes for inserting a line (insert), positioning the cursor (curxy), homing the cursor (home), clearing the screen (cls), and erasing to the end of the line (ereol) . These codes may not be compatible with other platforms.

Here's the code:

```
PROCEDURE wread
 0000   (*
 0003   (* wread - textfile reader
 001D   (* copyright (c) 1993 by Wayne
         Campbell
 0044   (*
 0047   (* textfile name parameter
 0061   PARAM filename:STRING[80]
 006D   (* insert line code
 0080   TYPE INS=icode,icommand:BYTE
 008F   DIM insert:INS
 0098   (* cursor x/y code
 00AA   TYPE CXY=ccode,xcor,ycor:BYTE
 00BD   DIM curxy:CXY
 00C6   (* line start position pointers
 00E5   TYPE FIL=line_start
 00EE   DIM file(4096):FIL
 00FC   (* path number, home cursor code,
         clear screen code
 012F   DIM path,home,cls:BYTE
 013E   DIM total_lines,counter
         ,current_line,screen_line:INTEGER
```

Arrows:
[UP] - move up a line
[DOWN] - move down a line
[SHIFT]-[UP] - move up one screen
[SHIFT]-[DOWN] - move down one screen

Letters:
[T] - move to the top of the file
[B] - move to the bottom of the file
[Q] - quit WRead

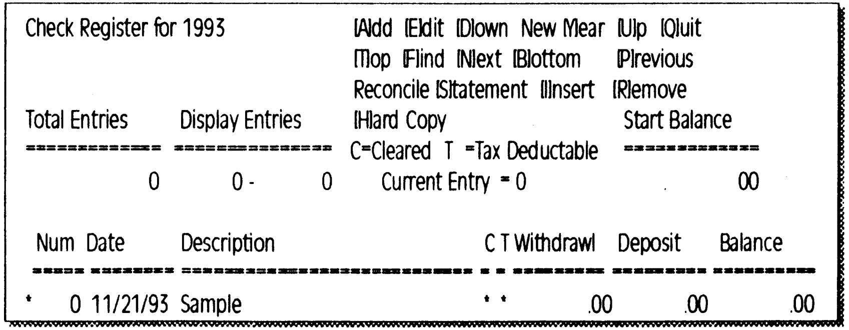
Continued Page 30

that you do in your physical register. When first run, checks creates a data file named 'checks.dat' in the current directory. You are asked if you wish to establish the base year using the default value. If you answer with n(o), you will be prompted to enter the year to use. This allows you to start with previous years and bring the register up to current from a prior starting place.

This is advised, because the register is established with a starting balance of .00 (no dollars and no cents). This allows you to start the first register with the initial deposit that opened the account originally. If this is not desired, you may alter the source code to start with a different starting balance. This only applies to the first register created. Each subsequent register created will use the last balance of the current register as the starting balance of the new register. (More on this later)

WARNING: Use caution when running checks. If you are NOT in the directory where 'checks.dat' is located, checks will create a new register, assuming it to be the first register!

After the register is created, the primary screen will be displayed:



```
 Check Register for 1993            [A]dd [E]dit [D]own  New Mear [U]p [Q]uit
                                    [T]op [F]ind [N]ext [B]ottom   [P]revious
                                    Reconcile [S]tatement [I]nsert [R]emove
 Total Entries     Display Entries  [H]ard Copy                   Start Balance
 ============  ===============  C=Cleared  T =Tax Deductable  ============
             0       0 -       0      Current Entry = 0                  .00

  Num Date       Description                     C T Withdrawl  Deposit   Balance
 ===== ======== ============================== = = ========= ========= =========
 *   0 11/21/93 Sample                          * *       .00       .00       .00
```

The upper portion of the display describes the current register year, the functions available, the total number of entries in the current register, the beginning and ending numbers of the entries currently displayed, the entry currently pointed to be the asterick (*), and the starting balance of the current register. The lower portion of the display is the register entry display. When first created, there are no entries in the current register, so a sample entry is displayed.

The fields of the entry display are described as follows:

*: This denotes the entry currently pointed to. When you use the functions [N]ext and [P]revious this pointer is moved up or down one entry from the current position. If you use the [F]ind function, this pointer will be placed next to the specified entry. More discussion of this pointer will be addressed later.

Num: If the entry is a check entry, the check number is displayed in this column. Any other type of entry leaves this column blank.

Date: The date of the entry is displayed in this column. There is

CHECKS Part 1

Check Register

by Wayne Campbell

This is part one in a four part series of articles about this program. Each part of the article will be accompanied by a companion source listing. Checks is NOT modular, ie. it is only one procedure, but it is about 16K worth of source code, and I decided it would be best to break it up for publication. The following describes the breakdown for the series is.

Part Functions Described

(1) Top, Bottom, Next, Previous Up, Down, Find
(2) Add, Edit, Insert, Remove
(3) Statement Reconciliiation, New Year, Hard Copy
(4) Sub-function for Add and Insert, Sub-function for Abbreviations, Sub-function for Balance Figuring

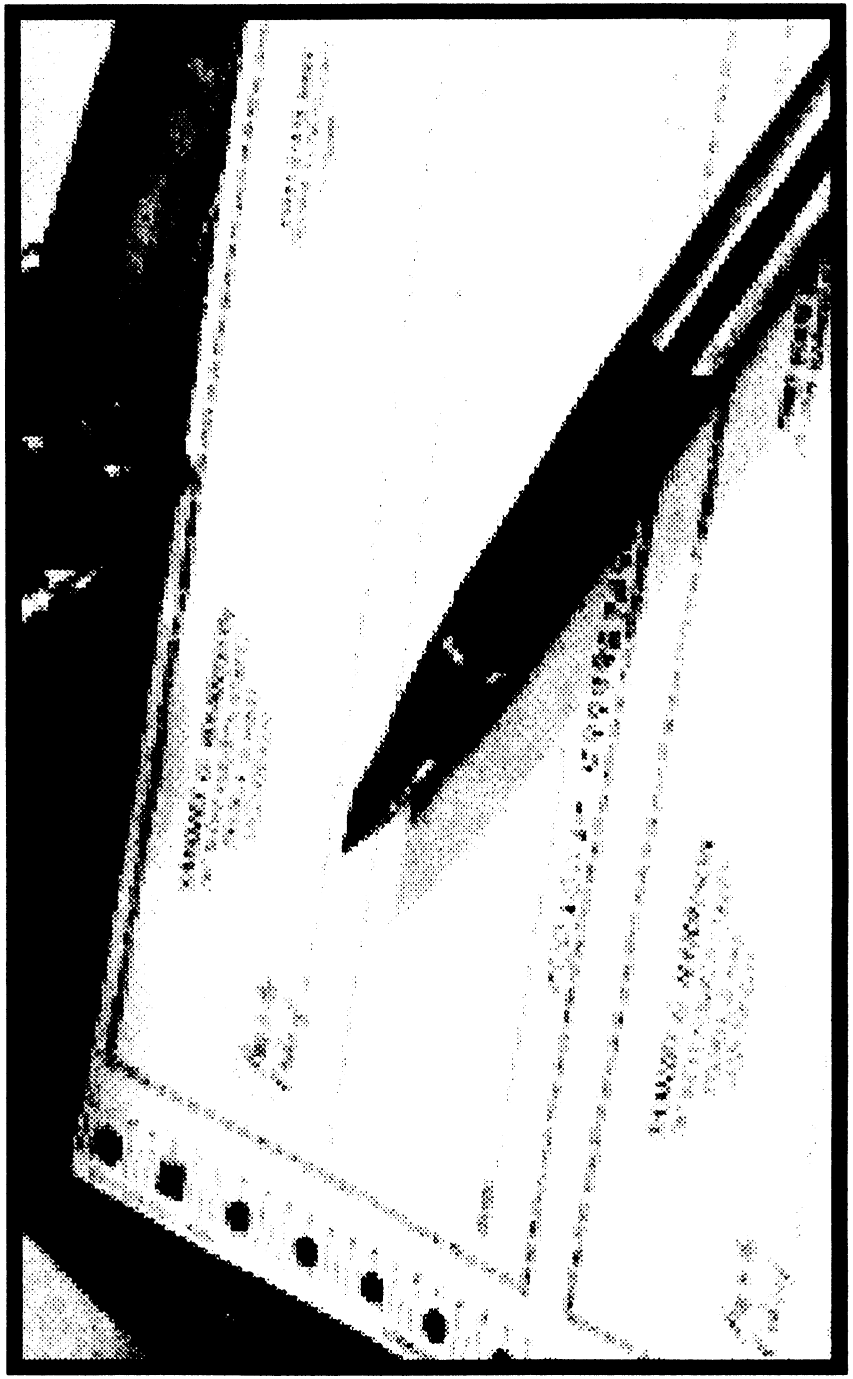
The first part of the source listing will run as is, but you won't be able to do anything with it because the 'guts' are in the other three parts. I will explain each of the functions included in each part as they are presented. When complete, you will have a complete program that will help you keep your checkbook up to date.

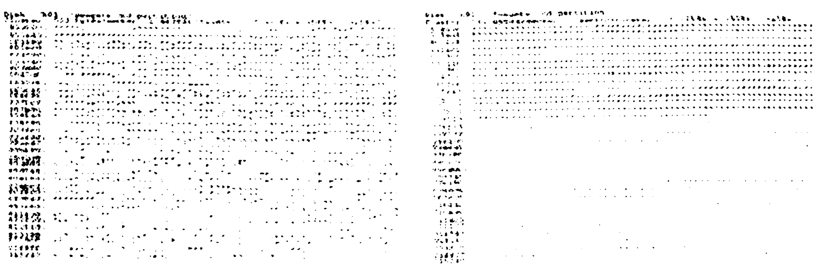
And now, Checks - Part 1.

Checks is a check register program that maintains current balance, reconciles your statements, and clears all entries accounted for, the same way

```
0151  (* end of line mark
0164  DIM line_end
0169  DIM key,up,down,shift_up
       ,shift_down,ereol:STRING[1]
0189  DIM keys:STRING[11]
0195  DIM text_line:STRING[79]
01A1  (* preliminary initializations
01BF  insert.icode:=$1F
01CB  insert.icommand:=$30
01D7  curxy.ccode:=$02
01E3  curxy.xcor:=$20+0
01F2  home:=1
01F9  cls:=12
0200  up:=CHR$(12)
0208  down:=CHR$(10)
0210  shift_up:=CHR$(28)
0218  shift_down:=CHR$(26)
0220  ereol:=CHR$(4)
0228  keys:="
BbQqTt"+up+down+shift_up+shift_down
0246  (* check for valid file
025D  ON ERROR GOTO 20
0263  OPEN #path,filename:READ
026F  SHELL "tmode -echo"
027E  (* this section determines where
       each line in the file starts
02BB  total_lines:=1
02C2  file(total_lines).line_start:=.0
02D4  WHILE NOT(EOF(#path)) DO
02DF    READ #path,text_line
02E9    line_end:=file(total_lines
         ).line_start+LEN(text_line)
02FD    total_lines:=total_lines+1
0308    file(total_lines).line_start:
         =line_end+1.
031E  ENDWHILE
0322  (* set start values and print the
       first 24 lines
0352  PUT #1,cls
035B  total_lines:=total_lines-1
0366  SEEK #path,0
036F  current_line:=0
0376  screen_line:=23
037D  REPEAT
037F    READ #path,text_line
0389    PRINT text_line
```

```
038E    current_line:=current_line+1
0399  UNTIL current_line=23
03A4  READ #path,text_line
03AE  PRINT text_line;
03B4  current_line:=current_line+1
03BF  curxy.ycor:=$20+screen_line
03CF  PUT #1,curxy
03D8  key:=""
03DF  (* here's the main routine
03F9  WHILE key<>"Q" AND key<>"q" DO
040E    REPEAT
0410      RUN inkey(key)
041A    UNTIL SUBSTR(key,keys)>1
0429    (* up a line
0435    IF key=up THEN
0442      IF current_line>1 THEN
044E        current_line:=current_line-1
0459        IF screen_line>0 THEN
0465          screen_line:=screen_line-1
0470          curxy.ycor:=$20
                +screen_line
0480          PUT #1,curxy
0489        ELSE
048D          PUT #1,insert
0496          SEEK #path,file
                (current_line
                ).line_start
04A6          READ #path,text_line
04B0          PRINT text_line;
04B6          PUT #1,curxy
04BF        ENDIF
04C1      ENDIF
04C3    ENDIF
04C5    (* down a line
04D3    IF key=down THEN
04E0      IF current_line<total_lines
           THEN
04ED        current_line:=current_line+1
04F8        IF screen_line<23 THEN
0504          screen_line:=screen_line+1
050F          curxy.ycor:=$20
                +screen_line
051F          PUT #1,curxy
0528        ELSE
052C          SEEK #path,file(current_line
                ).line_start
```





```
053C       READ #path,text_line
0546       PRINT \ PRINT text_line;
054E       PUT #1,curxy
0557     ENDIF
0559   ENDIF
055B ENDIF
055D (* down a screen
056D IF key=shift_down THEN
057A   IF screen_line=23 THEN
0586     current_line:=current_line+1
0591   ELSE
0595     IF current_line=0 THEN
05A1       current_line:
           =current_line+24
05AC     ELSE
05B0       current_line:=current_line+(24
           -screen_line)
05BF     ENDIF
05C1   ENDIF
05C3   IF current_line>total_lines
        OR current_line
        +23>total_lines THEN
05DB     current_line:=total_lines-23
05E6   ENDIF
05E8   GOSUB 10
05EC ENDIF
05EE (* up a screen
05FC IF key=shift_up THEN
0609   IF screen_line=0 THEN
0615     current_line:=current_line
          -24
0620   ELSE
0624     IF screen_line=23 THEN
0630       current_line:=current_line-47
063B     ELSE
063F       current_line:
            =current_line-(48-(24
            -screen_line))
0651     ENDIF
0653   ENDIF
0655   IF current_line<1 THEN
0661     current_line:=1
0668   ENDIF
066A   GOSUB 10
066E ENDIF
```

```
0670 (* top of file
067E   IF SUBSTR(key,” Tt”)>1 THEN
0690     current_line:=1
0697     GOSUB 10
069B   ENDIF
069D (* bottom of file
06AE   IF SUBSTR(key,” Bb”)>1 THEN
06C0     current_line:=total_lines-23
06CB     GOSUB 10
06CF   ENDIF
06D1 ENDWHILE
06D5 (* close file and end
06EA CLOSE #path
06F0 SHELL “tmode echo”
06FE PUT #1,cls
0707 END
0709 (* routine for displaying screen
      for screen up/down and top/bottom
074B (* of file
0755 10 PUT #1,home
0761 SEEK #path,file(current_line
      ).line_start
0771 FOR counter:=1 TO 23
0781   READ #path,text_line
078B   PRINT text_line; ereol
0794   current_line:=current_line+1
079F NEXT counter
07AA READ #path,text_line
07B4 PRINT text_line; ereol;
07BE screen_line:=23
07C5 curxy.ycor:=$20+screen_line
07D5 PUT #1,curxy
07DE RETURN
07E0 (* error trap and built-in help
07FF 20 PRINT
0804 PRINT “wread usage: wread
      <textfile> or”
0828 PRINT “          wread
      (“”<textfile>””)”
084C PRINT
084E PRINT “where <textfile> is a
      pathlist or filename upto 80
      characters”
088F PRINT
0891 PRINT “wread will handle a text
      file of a maximum of 4096 lines of”
```

Continued Page 60

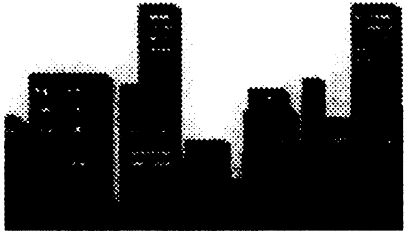
COMPUTER SCIENCE 201
INSTRUCTOR:
SCOTT MCGEE

LESSON 5: TREE TRAVERSAL

Last time, we dealt with insertion into a Binary tree. If you did your homework, you should have code that inserts values into a tree. Before we go on to deletion and searching, it might be nice to be able to see our tree. This is very helpful in debugging. To see the tree, we need to be able to traverse it, and print the values in each node.

Traversing a list was easy, you just start at one end, and move along until you get to the other end. With a tree, this method won't work. What we have to do, is find a way to visit each node in the tree. One way of doing so is to start at the top level and visit each node on that level in either a left to right or right to left manner. While accomplishing our goal, this has two problems. One is that the method has little to do with the structure of the tree, so is difficult to program. The other is that if we list the data as we traverse each node, no useful order is returned.

by Tony Podraza



1) Barsoft, Dave Barnes
2) ColorSystems, Zack Sessions
3) DELMAR, Ed Gressick
4) Hawksoft, Chris Hawks
5) Farna Systems, Frank Swygert
6) Crystal Palace BBS, Nelson Howard et al
7) DISTO, Tony Distefano
8) Budgetware, Brian Kitt
9) JWT Enterprises, Jordan Tsvetkoff
10) The National OS9 User's Group
11) The Chicago OS9 User's Group

In addition, there have been verbal intentions of attendance by Kala Software, CoNect, SBUG, Fat Cat Publication Adventure Survivors, and one or two others whose names escape me at this time.

Seminars are being planned for, but are not yet cast in concrete, except for the first meeting on Sunday Morning, which we hope to continue as a tradition. That meeting will be the Meditation, Praise, and Worship Service presided over by Brother Jeremy, a CoCo Comminuty member for as long as I can remember, having seen him at the first Rainbowfest I attended, which I believe was in 1986 in Schaumburg, Illinois.

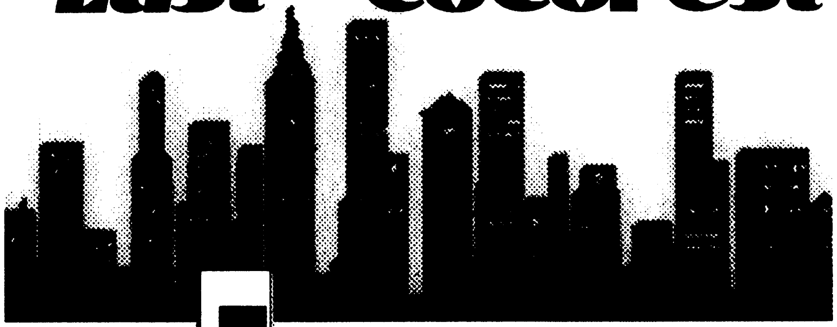
The site of the FEST! is fairly easy to get to, being alongside of I90 at IL RTE 31, with a couple of turns onto West River Road. Air travelers can reach the site from Chicago's Midway Airport by taking I55 west to I355 north to I90 west to Elgin; O'Hare arrivals would take I190 out of O'Hare and follow the signs to I90 west to Rockford, but be sure to get off at IL RTE 31 in Elgin; Arrivals at Mitchell Field in Milwaukee will want to head south on I94 to I294 south to I90 west (same as before); and finally, should you fly into Rockford, head north to IL RTE 20 east to I90 east until you approach IL RTE 31, and follow the earlier directions to the Holiday Inn.

All in all, we at GLENSIDE expect to have a fun-filled weekend filled with friends, food, excitement, and prises....oh, did I forget to mention the PRISES? Well, you'll have to come to find out about those. But rest assured, they will be there, along with the BADGES that we forgot about, last year. BELIEVE ME!!!! After all the requests for BADGES! We won't forget them a second time! Come and join us in the revelry.

-Tony

For ticket information and reservations, please check the Chicago CoCoFest Ad on Page 3

The 3rd Annual "*Last*" CoCoFest



This is the second year that GLENSIDE Color Computer Club is UNDERtaking the sponsorship, planning, and execution of the Chicago-area held CoCoFEST!

Last year brought 30-plus exhibitors to the show, including Microware, the developers of OS-9, which is proving to be the heart of the hard-core CoCo enthusiasts. Disk Extended Color BASIC programs and programmers were also highly visible, as they should be, being the operating system of machine, out of the box. We we further excited that so many exhibitors chose the FEST! to present their wares for the OS9-68000 systems, and we further hope to garner their presence this year, again.

This year, GLENSIDE is officially announcing that tickets for the FEST! will be discounted by the amount of $5.00 for students between the ages of 6 to 16, when accompanied by an adult, and that children 5 years old and under will be admitted free of charge.

As of this writing, there are eleven official exhibitors. They are:

Another way to tackle this problem is to consider the tree structure. Each node in the tree has three item, data and two children. The tree is a recursive implementation of the node itself. If we consider looking at a single node, then use a recursive form of that, we should find a useful method for traversing the tree.

Lets consider the order in which we can traverse the three items in the tree. There are only six ways to do it, and they can be further divided into forward and backward versions of just three (*depending on the order in which the two children are visited*) and these three are:

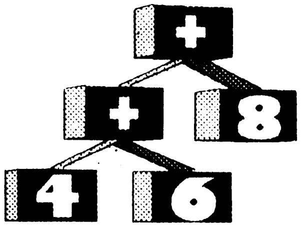
```
1. data, child, child called pre-order or prefix
2. child, data, child   called in-order or infix
3. child, child, data   called post-order or postfix
```

To impliment a recursive form of these, we simply make each visit to a child become a recursive traversal of that child, ending the recursion when no child is found in that direction in that node. In other words, a pre-order traversal may look like this:

```
examine data in node
traverse left child
traverse right child
```

If we consider the tree to be a mathematical expression with operators in all internal nodes and values in all leaf nodes, this traversal method will print the expression in prefix order much like conventional programming command languages use. By this I mean something like +(2,3). This would evaluate to the addition of 2 and 3.

If we take a simple tree structure, but substitute a mathematical expression for the node values, we might get something like this:



If the first step, examine data in node, is simply a print statement, we would see the following happen (starting from the root node)

```
print data
move to left child (+)
 print data
 move to left child (4)
  print data
  no left child
  no right child
  return to parent (+)
 move to right child (6)
  print data
  no left child
  no right child
  return to parent (+)
 return to parent (+)
move to right node (8)
 print data
 no left child
 no right child
 return to parent (+)
return (no parent)
```

We would get the following output: + + 4 6 8

The pre-order traversal of "+ + 4 6 8" which could be interpreted as +(+(4,6),8). Written in normal algebraic notation, it would become (4 + 6) + 8. As I mentioned, this prefix notation is useful in command type languages.

Now, if we turn to method 2 (in-order traversal), we get some very interesting results. The in order method looks like:

```
traverse left child
examine data in node
traverse right child
```

The same tree, traversed in-order, goes like this:

```
move to left child (+)
 move to left child (4)
  no left child
  print data
  no right child
  return to parent (+)
 print data
 move to right child (6)
  no left child
  print data
  no right child
  return to parent (+)
 return to parent (+)
print date
move to right child (8)
 no left child
 print data
 no right child
 return to parent (+)
return (no parent)
```

It produces the output "4 + 6 + 8" which looks suspiciously like our algebraic form. In fact, if you added code to generate a "(" on the move to left child step, and a ")" on the return to parent step, it WOULD generate "((4+6)+8)"!

Finally, lets try post-order traversal. It looks like:

```
traverse left child
traverse right child
examine data in node
```

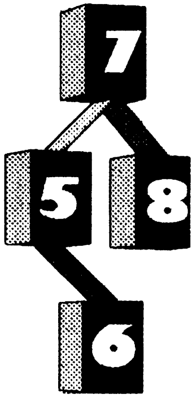
And on the same tree, works like this:

```
move to left child (+)
 move to left child (4)
  no left child
  no right child
  print data
  return to parent (+)
 move to right child (6)
  no left child
  no right child
  print data
  return to parent (+)
 print data
 return to parent (+)
move to right child (8)
 no left child
 no right child
 print data
 return to parent (+)
print date
return (no parent)
```

producing the output "4 6 + 8 +" which any HP calculator user will tell you is perfectly good RPN (Reverse Polish Notation) notation for the same algebraic expression given before.

Now, lets again consider our binary tree. We have three good methods for traversing a tree, but which should we use? Well, the real advantage of a binary tree is that it represents a linear relationship on the values in it. This same relationship is preserved by an in-order traversal, so it is the natural choice for binary tree traversal.

Now consider the tree we built last time. It looked like this:



The in-order traversal will give "5 6 7 8" which is exactly what we had wanted to get. Therefore, we would prefer to use the in-order method to traverse our binary trees. This traversal will allow you to print out your tree values.

The homework assignment is to add to your code for building a binary tree, a routine to print out its values using an in-order traversal. Next time, we will continue on with deletion and searching, which are both very closely related.

Class Dissmissed.

- Scott McGee

Comments to Scott McGee may be sent email:
Internet: smcgee@cymru.UUCP or smcgee%cymru@uunet.uu.net
UUCP: uunet.uu.net!cymru!smcgee
STGNet: smcgee@os9er
(or c/o this Magazine, see page 68)

<EOF>