SYSTEM IV

The 68000 Computer serving customers world-wide

This high-quality, high performance 68000 computer was designed for and is accepted by industry. Perfect low-cost work-station, development platform or fun machine. Powerful, flexible and expandable inexpensively. Run MS-DOS software with the optional ALT-86 card. Supports up to 4 operating systems.

Prices start at $999.00 with Professional OS-9

For boards and kits, call Peripheral Technology at 404-973-2156

G-WINDOWS

for the SYSTEM IV and PT68K4/2

Multi-tasking - processes continue running when windows are made inactive or are hibernating.

Windows may be re-sized, moved, overlaid, etc.

GUI to start processes by selecting an icon or, start processes from your custom menu or from the command line.

Copy and Paste between windows.

Adds command line editing, command history, and file name expansion.

Runs existing OSK software without modification.

Number of windows and processes limited only by your memory.

Includes GIF viewer.

Includes G-VIEW demo.

G-WINDOWS with DESKTOP	$199.00
G-WINDOWS Developer's Pak	$299.00
Order both for	$449.00

OS-9/68000 SOFTWARE

NEW - DataDex, a free form data management program designed to keep records like a card file system. No programming language to learn. Variable record sizes permitted.

QUICK ED - Screen Editor and Text Formatter	CALC-9 - Spreadsheet
VED ENHANCED - Text Editor	VPRINT - Print Formatter (for VED)
SCULPTOR - Development and Run-Time Systems	M6809 - OS-9 6809 Emulator/Interpreter
FLEXELINT V4.00 - The C Source Code Checker	DISASM_OS9 - OS-9/68K Dissassembler
WINDOWS - C Source Code Windowing Library	PROFILE - User State Program Profiler
IMP - Intelligent Make Program	PAN UTILITIES

delmar co

Middletown Plaza - PO Box 78 - Middletown, DE 19709
302-378-2555 FAX 302-378-2556

ColorSystems

Quality OS-9 Software for the CoCo3 and the MM/1 from IMS

CoCo3 Software

Variations of Solitaire $34.95

Variations Included:
Pyramid, Klondike,
Spider, Poker,
and Canfield

OS-9 Game Pack $34.95

Package Include:
CoCothello
CoCoYahtzee
KnightsBridge
Minefield
and Sea Battle

WPShel $22.00

(Sorry! ColorSystems no longer carries MVBanner!)

MM/1 Software

Variations of Solitaire $49.95

Variations Included:
Pyramid, Klondike,
Spider, Poker,
and Canfield

OS-K Game Pack $49.95

Package Include:
Flip It
Dice Poker
KnightsBridge
Minefield
and Sea Battle

Coming Soon for the MM/1:
Super Label Printer
X10 Master Control Program
and Much More!

CoCo and OS-9 Club Members: Have your Club President write to ColorSystems to ask about our SPECIAL Club Discount Program!

Offical Member of the Interactive Media Systems Developers Association

Shipping: FREE for Continental US, $3.00 for Canada, $5.00 anywhere else
To order send check (US Bank ONLY!) or Money Order (US Funds) to:

ColorSystems

P. O. Box 540
Castle Hayne, NC 28429
For Additional Information call at (919) 675-1706
North Carolina residents please include 6% Sales Tax

Write or call for a FREE copy of our Catalog!

NEWS ∞ NEW PRODUCTS ∞ CONVENTIONS

• • • CONVENTIONS • • •

The 3rd Annual Atlanta CocoFest!
October 3rd and 4th 1992
Held at the Holiday Inn Northlake
Admission $5.00 at the door
For Motel Reservations call (800) 465-4329 or (404) 938-1026

• • • • •

BUSCON '92 East
September 15, 16 and 17th 1992
Held at the Hynes Convention Center, Boston, Massachusetts
For a Complete Program Brochure call (800) 243-3238
(Stop by Microware's booth #117)

• • • NEW PRODUCTS • • •

DataDex, the Premier Free-Form Data Management program for OS9/68000
Designed to keep records in much the same way as a "Rolodex (R)" does
Many Powerful features. Single-User and Multi-User versions available
delmar co. (302) 378-2555

• • • • •

NitrOS9 gives your 6309 a kick in the Boot! Modifies OS9 Level2 to
take advantage of the features in the HD63B09E.
20% OVERALL increase in speed, some as high as 10 times!
Gale Force (604) 589-1660

• • • • •

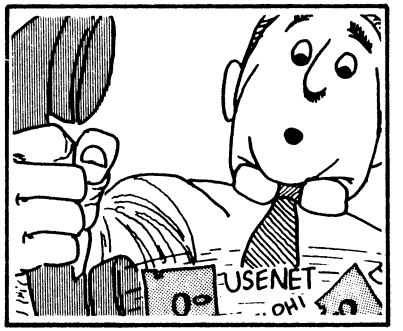
The Micro Chart - C Programmer's Instant Reference Card
Commonly used C functions on a durable 8.5" x 11" Plastic Sheet
Sirius Software/Hardware (818) 894-0012

• • • • •

The "International" OS9 Underground®

Magazine Dedicated to OS9/OSK Users Everywhere!

August 1992 CONTENTS Vol 1, Issue 3



Editor/Publisher:
Alan Sheltra (Zog)

Associate Editor:
Jim Sutemeier

Technical Editor:
Paul Pollock

Submissions Editor:
Jim Vestal

Contributing Editors:
Ron Anderson
Leonard Cassady
Andy DePue
Ed Gresick
Scott Griepentrog
L. Scott Kasten
Bob van der Poel
Zack C. Sessions

Art and Typesetting:
Alan Sheltra

Printing:
Pink Printers

The OS9 Underground Magazine is published monthly by Fat Cat Publications, 4650 Cahuenga Boulevard Suite #7, Toluca Lake, CA 91602. The OS9 Underground and Logo types are registered trademarks. The OS9 Underground is $18.00/year shipped first class in the U.S., $23.00 in Canada and $27.00 overseas (airmail add $22.00/year). Article and program submissions to the OS9 Underground are always welcome and may be sent to the above address in electronic format or to our BBS (818) 761-4135 (8pm to 8am PST). Advertising rates are reasonable, please call or write for our rate card (818) 761-4135 (9am to 7pm PST).

MailRoom

LETTERS TO THE EDITOR

ATTABOYS

Dear Editor:
[Sent via Delphi's Internet Mail]
Hi! I just got the OS-9 Underground in the mail today. All I can say is W-O-W!!! This thing is jam packed! It is excellent.. Great.. Super.. Awsome...etc.. In fact, I'm even writting this message to you during Delphi's Prime Time to tell you this.. (Just to give you an idea about how good the mag is).. It has so much useful information.. It's amazing.. Before when the Rainbows were 350 pages in size.. They didn't come close to this much useful OS-9 info! It took me quite a while just to "scan" through it! Something the Rainbows havn't been able to do in a LONG time.. This is really great! Keep up the awsome work! Thanks for everything!

-Andy DePue
Greensville, SC

●●●●●

Dear Editor:
I just received my second issue of OS9 Underground (Tuesday) and once again, I am impressed! The larger size is great, especially considering the "substance" contained within the pages. Also, the envelope mailer insures it will arrive intact. Great job on the more readable print, too. Best of all, your fast turnaround means we get all the news MONTHS before the "competition". Please keep it going.

-Allen Huffman
(Sub-Etha Software)
Lufkin, TX

●●●●●

UNFAIR TO BASIC

Dear Editor:
I got the july issue of the Underground magazine, good stuff. If the magazine keeps growing (I hope so) soon I will have to rebuild the mail slot in order that the magazine fits there.

Not everything is roses, I have a small complaint; I think the C puzzle is not fair, it limits the number of people that can participate. Not everybody knows C at such level that is perfectly familiar with pointers. And some people don't even know a bit of C.

But anyway it is a great stuff, I can live without been able to participate in the puzzle. Keep the good work Alan!!!.

-Marcelo Katzeff
Poco, BC Canada

●●●●●

Dear Editor:
Actually, this is about the C pointer puzzle.. When I got my copy of the July issue the other day, I made a comment to Alan about how 'easy' the puzzle looked to me, in spite of my inexperience with C... well, it's time to put my foot directly in my mouth! Boy! I can see someone with about 1000 straight hours of C programming under his/her belt figuring it out!

Well, I'm still new at C, so maybe I'll understand it better in a few hundred hours or so.

BTW, Alan, keep up the good work! It's a GREAT magazine!

-Wayne Campbell
Sepulveda, CA

●●●●●

LOOKING FOR SOMETHING GNU

Dear Editor:
How about starting C programming articles about how to set up an OSK system using MW's C, GNU C or GNU C++ for people that want to get started. Ed Gresick's article [July '92] is very good.

-Howard Luckey
Park Forest, IL

[Any taker's? I'd love to print one if someone is willing to write it. -Zog]

●●●●●

CD-I, PIE IN THE SKY?

Dear Editor:
I'd first like to say congratulat-ions on the merger with The 68xxx

("Letters" - Continued from Page 5)

I, for one would very much like to be in a position to capitalize on the potential growth you describe. Not neccessarily in the "Dollars and Cents" sense, but in the sense of being part of (and perhaps influential in) the development of something NEW. At least "something new" as far as the general public is concerned.

A few questions came to mind though as I read your article. First, how is it that we (or any one outside of the BIG electronics giants perhaps) can know that CD-I will be supported (from the computer side) as noted? Second, how can one say definately, at this point, that OS9 will be the "designated" operating system?

One of the reasons behind my making these inquiries is that I am currently weighing pros and cons of investing in the OS9 operating system. I currently have a PT68K-2 with SK*DOS, which I've been very pleased with to date. However, if OS9 is going to become globally more significant, as your research appearently indicates, maybe it makes sense for me to start saving my pennies and take the plunge?

-Frederick J. Craft
Fountain Valley, CA.

●●●●●

KUDOS TO FRANK

Dear Editor:
This letter is in response to to your article in 68xxx. It was one of the best articles that has been run to date. As it happened, I was (wishfully) thinking about getting a CD-ROM drive. DAK Industries is advertising an external drive that has volume control and an output jack. Hmmm maybe I could use it to play audio CD's as well?

-Mike Neary
Spring City, PA

●●●●●

REVIEWING THE REVIEWS

[This is a reply to A.C. Huffman,-Ed]
English is a horrible tool for the exchange of ideas, between two people. Especially written English. For instance, English only provides one word for the process and behavior known as 'sex'. And one more for the action of 'sex' (which cannot be said or written, because of religious and cultural taboos). At the same time, the Hawaiian language has 34 distinct, nonderived, nonconjugated words to describe 'sex'; none of which are restricted from dialog.

What this points out is, that while English is very strong in words and terms which describe physical attributes; it is extremely poor in words to describe context, texture, emotions, or feelings. That is why we use so many words to attempt any description of one of these 'fuzzy' ideas.

The "...buffer capture..." quality of the UNDERGROUND columns, is NOT caused by amatuerish conduct. The phrases like (grin) or (!!!), evolved out of an early need to cram as much meaning into as few words as possible; over electronic medias. These little adjuncts are called 'visual cues'.

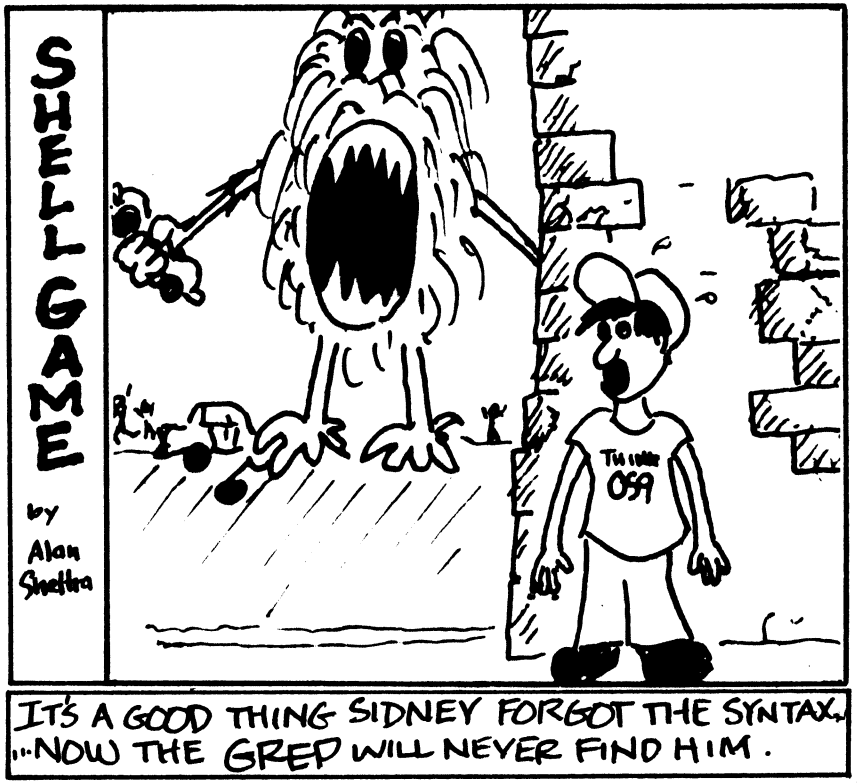
If I were speaking to a person, he can usually watch my face, or hear my tone and inflection to get clues to meanings that might be impossible to gather without a great deal more spoken words. Without these clues, English is monotonous, staid, boring, and non-definitive.

The point is, when a person uses these 'visual cues' in his text, he is NOT producing less. He is producing MORE.

The fact is that the non-BBS community is being offered an opportunity to receive as much information as possible. The idea that '...they might not get it...' is irrelevant. With practice, they will learn that we are doing a better job of conducting ideas with less text.

We are not UNIX-WORLD, Service News, Dr. Dobbs Journal, or the National Review. We are THE UNDERGROUND. We have our own flavor. Which for the most part, has color, texture, shape and form. And it doesn't have to ascribe to performance standards considered acceptable by other publications. And we shouldn't.

-Paul Pollock (Technical Editor)
Corvallis, OR



Machines. We hope you will deliver more valuable information to more people by this.

Reading Ed Gresick's critiques on Frank Hoggs CD-I utopia issue, we mostly agree with Ed. We felt disappointment with the magazines quality when I read the braggart article of Frank Hogg. It seemed to have some similarity to Lonnie Falk's old phrases such as CoCo never dies or we see more Rainbow subscriptions even after changing to the tabloid format.

Yes. The article is sort of commercial. But it is natural for corporate authors to advertise themselves and their own products a bit in such articles, and I am sure we would do somewhat similarly. At least the commercial portion is not as blatant as others.

The article predicts there will be 100 million CD-I units sold in the US by the mid 90s. Although I am new in this country, if my knowledge is correct and Frank Hogg uses the same simple mathematics as what I learned at elementary school a long time ago, the US population is only less than a quarter billion and 100 million units means more than one unit per every two adults. I wonder if even the number of telephone units in the US has exceeded this.

As the article first mentions, a CD-I player is just consumer electronics, not a computer. You could count bank tellers as OS-9 users if the banks ATM has embedded controller using OS-9 inside.

I have no idea how deep FHL has been involved the CD-I business; nor have I any idea what their K-bus looks like and how it functions with a CD-I player. The article insists that a CD-I player is basically an OS-9/68000 computer and that its EASY (how easy?) to add expansion boards and other things such as disk drives. Yes, that could be true, and, since there are nearly a quarter billion people in this country, there could be at least a few thousand odd balls with bucks in hand to spend. But the question is what can you do on your new OS-9/68000 computer? If you are happy executing only DIR and PROCS, no words are needed. Spreadsheets and word processors? Unfortunately, home TVs shadow mask alignment is staggered to minimize jagginess of natural pictures, not straightly aligned like regular computer monitors. So the reasonable number of characters a home TV could display without irritation would be 32x16 or so (if you are a CoCo user with a tight budget, you may have noticed this already). I wish the article had suggested any useful computer application with this resolution.

CD-I is not new. The project started in 1985. From 1986 to 1987, we were actually involved as consultant in a CD-I authoring machine project using OS-9. Though we succeeded in building a prototype authoring bench but it did not grow into a commercial product. The biggest reason for this was lack of proper project management. We believe this is one of the reasons that have made the entire CD-I almost extinct.

Other reasons may include that it was based on OS-9, not DOS. Yes. OS-9 has a very elegant internal architecture, but consumers do not buy a thing for its internal beauty. It is a publicly known secret that Bill Gates of Microsoft offered a purchase of Microwares business when there was CD-I enthusiasm. If Ware had become part of Soft, CD-I could have been successful with at least as many units as the MacIntosh installed.

Technically, CD-I WAS very interesting at the time it was introduced, but not any more. You can see many interesting interactive programs for DOS and the Mac at reasonable prices. Once we decide, we will try buying a CD-ROM player for our Mac to play Sherlock Homes, Consulting Detective for fun.

We hope Eds article and this prevent your readers from having silly illusions.

-Hiro Sugawara
Santa Clara, CA

• • • • •

TAKING THE PLUNGE

Dear Editor:

I am writing to you after having read your article on CD-I in the May/June Issue of of "68xxx NEWS". It was an interesting and thought provoking article. Thanks for sharing!

("Letters" - Continued Page 53)

Under it all...

The theme for this month's issue is "Communications". Many of us rely heavily on our modems for finding out "what's new" and simply "staying in touch". As editor, I find it extremely important (and convenient) to be able to leave a message ("e-mail") to the various writers and subscribers thru this medium. For your information, letters to the editor may be sent via UUCP to me at the following address:

"zog!os9under@abode.ttank.com"

Those of you that have internet mail access on Compuserve, Delphi or GEnie can send your letters directly from there. If you don't have access to those services (or prefer to write) use the supplied postcard to send your comments (or gripes!) to the editor. If you'd like to address your message to any of our authors, I will forward all mail or email to the appropriate person. Keep those letters coming!

Andy Depue, who took a breif hiatus last month, continues his tutrial on C data modules. This article keeps with our theme because he describes how to make your own "chat" program for use on a BBS or multi-user system.

We feature a pull-out BBS listing for your calling pleasure... as well as a breif overview on some of the BBS and Network packages that are currently available for OS-9.

●●●●●

The OS9 UNDERGROUND is always looking for writer and submissions. If you'd like to submit an article or program (or both), you may send your submission on a diskette in the following formats: 35 or 40 trk COCO format 5.25 diskette. 80 trk 5.25 or 3.5 COCO or OS9-68000 format. Please include your name and address. Submissions may be sent to this magazine:

OS9 Underground Magazine
Submissions
4650 Cahuenga Blvd. #7
Toluca Lake, CA 91602

●●●●●

The response to the C Pointer Puzzle has been practically nil! I've only had 1 (yes, One!) correct response to the puzzle (and he wasn't qualified!).

Anyway, next month will feature a new puzzle... one a bit simpler!
The answer to July's puzzle will appear next month also! -Alan Sheltra (Zog)

program and the machine language program, "Remote". I progressed through several other basic BBS programs, and late in 1988, took "The BIG Step" into OS-9 when I found the basic programs too limiting, or too complex for easy use. (I won't mention the two OS-9 BBS systems I used before being introduced to StG, for they are still popular, even today, among other Sysops.)

I first found information on the StG BBS package in the October, 1989 Rainbow magazine, with comments about "DES password encryption", a list of features that made my eyes water, and the comment that "Any OS9 command can be run from login, no doors required." This, particularly interested me, because of the difficulty of running OS9 commands from the other BBS packages that I had been using. (One even required rewriting programs in order to use the "standard I/O" used by that BBS software! How "anti-OS9" can you get!) One of the other notes that interested me was "Limited free updates -modifications available...", which in truth, turned out to be NOT so limited.

At the prompting of a friend of mine, who was also a user on my system, I contacted StG, and received the software. On Feb. 6, 1990, at 5:45 pm., I officially started this software running, and have never been sorry for it. The "networking" capability has been useful in more than just "keeping in touch" with other OS9 users, as it also allows (and has provided) updates to the software. Furthermore, the Sysops and other users on the network have ALSO been supporting the system, by adding other programs to the package that enhance its operation. The StG Network is truely supported by its users!

Enough about the history, down to the features. Firstly, the original OS9 commands TSMon and Login have been replaced. The new versions (and upgrades) have the capability of auto-sensing the baud rate, by use of the modem's "Connect" message. In order to do so, either the use of a CDI (Carrier Detect Interface) in the RS-232 line, or the use of SACIA with the correct bit patched, will allow OS9 to receive messages from the modem BEFORE the CD line is active by connection to the remote user. Then, Login takes the users name and password, compares it to the password file, and takes action according to the command line in that file. (Note: Due to DES encryption, it is almost impossible for anybody to list the password file and read any users password. Not even the Sysop can do it! Talk about security!

Any command can be passed to Shell from the password file. The default goes to the main menu, (or whichever is specified by the Sysop). This allows special accounts to be setup that will allow, for instance; Special User groups, with their own sub-directory and mailbox. Stand-alone games that prompt for secondary usernames and passwords. On-line catalogs, to allow a user to browse and shop, without going through standard login procedures.

Each user gets his or her own private "workspace" sub-directory. In that workspace, they can upload, download, edit, delete, rename, or do whatever else they wish to do, to their files there, without disturbing files that are designated for system use.

The menus and most of the support software, also have the capability of sensing and using any of three different graphic setups for the individual user; OS9, ANSI, and plain TTY. This opens up tremendous possibilities for the Sysop to customize the menus to his own tastes.

One more note about the network transfers, they are true bi--directional. Data is passed in both directions at the same time. This helps to cut down on total long distance connect time significantly! And, the mail and public message distribution is automatic, through a program module appropriately named "Postman".

So far, StG BBS package is written for, and runs on OS9 Level 2 computers, specifically the CoCo 3. There is work going on, to provide the package, (with full compatibility to the CoCo version), on the OS9-68K family of machines.

In closing, I can only say that I am thankful to that friend who persuaded me to start using the StG BBS system. It has opened up OS9 in a way that no other system I had tried previously could do.

-Scott Proctor
Sysop@Home
StG International Network

RiBBS

(No Bones About It...) As you can see from other examples in this magazine, the OS9 Underground seeks to keep its readers up to date on the BBS systems available to OS9 users. No such effort would be complete without including information on what is arguably the most well-known (and possibly least understood) of the OS9 BBSes; RiBBS.

RiBBS traces its history back about five years and starts with Ron Bihler. Early rumors that RiBBS was a pirated version of PBBS are quite untrue. To set things straight, Ron Bihler did indeed run PBBS at one time, having purchased the program and registered source code from another dis-enchanted sysop. Being the same kind of CoCo-hacker than many of us are, he changed the program to suit his needs - improving and adding to PBBS' existing features. Having offered these improvments (at no charge) to the author of PBBS (the name escapes me now) and being ignored, he set out to write a new system from scratch. Due to the need to keep his BBS up, he rewrote portions at a time - keeping the same names for modules that PBBS had (probably leading to the aforementioned rumors... though a quick check of sizes and CRCs might have quelled the idea, had someone thought to check.)

Suffice it to say that when RiBBS 1.0 was released (as freeware), all of the programs were 100% authored by Ron with the exception of the ML Xmodem block routines... which were written by Bob Montowski (of Graphic PUB fame). Prior to releasing this freeware package, Ron obtained Bob's permission to include those routines in the release. As it happens, those routines have also long-since been replaced by new ones.

My first experience with RiBBS was in late 1989 when the v1.9 pre-release was made. When version 2.0 was released in early 1990, I put up a system in earnest and called it "Arrakis". It has been running continuously ever since. After two major revisions by Ron (and a few contributing Beta authors - including myself), RiBBS v2.02 was released in late 1991 and the future developement of the project was handed to me in April of 1992. Several of us are working hard to come up with a version 2.1 release in the near future.

There, now you have the (somewhat condensed) story of RiBBS development up to the present day. Some of you may be wondering what it is about RiBBS that sets it apart from the other BBS systems available.

The most obvious feature is that to this day, RiBBS remains the only OS9 BBS that is fully capable of operating as a node in FidoNet and/or compatible nets (such as EggNet, AlterNet, FamilyNet, etc...). FidoNet itself is one of the most widespread networks in the world - with over 16,000 autonomous BBSes tied together on six different continents around the globe.... from Auckland to Iceland; from Siberia to Singapore; from Nashville to Nairobi.

Each system has its own address, and there are elaborate routing procedures that allow private and public messages (and in some cases, files) to be sent from place to place at negligible cost.

Other features of RiBBS are: ANSI/OS9 color and cursor control; a free-form menu system that allows each RiBBS to look different from every other one; up to 40 message areas; a file transfer system with multiple protocols, virtually unlimited download areas, and the potential for in-depth descriptions of the uploads (much like Delphi allows); and literally dozens of dozens of online games and support programs written by RiBBS sysops that simply "plug-in" to the bulletin board.

In the future, I'll try to use this space to keep you up to date on what is happening with RiBBS. If your questions just can't wait, feel free to contact me at Arrakis BBS (See Below) or attend the weekly RiBBS help conference on Delphi Thursday nights at 10pm Eastern time.

-Charles R. West
Arrakis BBS
(405) 752-8955

• • • • •

StG Net

I personally started BBS'ing sometime in 1987, using an RSDos basic

C NOTES:

by Andy DePue

DATA MODULES IN C:

Part Two

In the June 1992 OS9 Underground I outlined what an OS-9 data module was and of its various uses. In that article I focused mainly on it's use as a method for inter-process communications and began work on a "chat" program to demonstrate this and was going to finish it up this month. Because of space limitations, I can only give you half of that program now, and the other half next month. This month we will review and explain how to use the chat program.

DESIGN REVIEW

Before we go on, it may be best to review a little. The general idea behind this chat program is to allow conversation between two users on the system and to demonstrate how to use a data module for inter-process communications. Thus, it has been kept to the simplest design.

The data module for "Chat" is arranged with a certain number of "records", each record contains a flag that determines if it is in use, space to hold the user id, user name, and a text buffer for the user. The data module used in our example contains only two of these records, which means only two users can be in "chat" at a time. At the top of the data module is a byte which contains the number of records in the module.

USING CHAT

Since I will not be able to give you the entire source this month, it would probably be best to explain how the source works when we have it all. So for now, I will just explain how to use the program.

Once you have the complete chat program, you will either have to put ChatMod (it's data module) in the CMDS dir or merge it with chat. The latter being most desirable since it will save memory. Typing chat at the OS9 prompt will begin the program and will usually respond with:

What would you like to go by in this conference?

Enter the user name you would like to be known by in the conference. Note that if the max number of users are in the conference, it will error out. If you include a device name on the command line along with chat, it will page that device, for example:

```
Chat /T2
```

Will page device /t2, telling the user on the device that you would like to talk with him/her.

Once you have entered the conference, whatever you type will be placed into a buffer until you press ENTER, at which time the message will be sent to the other user in the conference.

Also available while in conference are "slash" commands; which are commands you can execute by preceding them with a '/' character. The available commands are:

Command	Description
/name [newname]	Change your conference name or display your current name. If [newname] is not specified, current name is shown.
/who	List of users in con-ference.
/time	Current time and how long you have been in chat.
/exit	Exit conference.
/page /device	Page device.
/help	A full help screen.

Global reading from all bases with one command, Unlimited amount of message bases of unlimited size, and all the features from previous versions such as allowing use of color codes in messages etc. etc.. Version 2.4a also alows access to the popular Fido-Net network via a gateway program along with Full support of UUCP via Rick Adams UUCP.

The AcBBS Menu Handler has also gone thru a total re-write to now allow upto 250 menus instead of the 120 menu limit of previous versions and Allows 2 types of Stock style menus that the sysop can customise there colors to there liking. It Also supports use of All custom text menu's that a sysop may wish to use for a customized bbs look.

There are also many outside programs that can be run on AcBBS such as a Call back verifier with sysop definable Calling area, Voteing booth, Over 30 online games, Door Program to Run All RiBBS games, etc. etc..

The above is only a very small part of the new features of AcBBS and there is sure to be many new features in days to come such as a full Fido-Net interface and many more..

-Chris Serino
(Co-Auther of AcBBS)
Delphi :CSERINO

●●●●●

RCIS

Rainbow Connection Information Service (RCIS) was established in 1987, and written wholly by Steve Rottinger of New Milford, NJ. When I met up (by phone) with Steve Rottinger, he explained to me that this system was designed just for him to run, out of his house. He did not plan on marketing or selling the system. I convinced him in 1988 that if we made certain changes to the software it could be marketable. For about two years I helped him develop and beta test the first RCIS dedicated network. This network was comprised of 8 systems. After the bugs were worked out we started sending software updated thru the network. Networked BBS lists and network conferencing followed. We also implemented network file lists, so a user on one system could request a file in a download section on another system. Some of the main features of the system are as follows: Real Time Conferencing, Nodal or Host Net Mail, Sysop creatable On-line games or utilities, Sysop configurable menus, Easy and quick access to messages, file transfers and menu selections, ANSI, OS9 or plain text control and color changes, along with windows, Electronic SOFTmail to send binary files to other systems/users, Unlimited number of SIGS, and unlimited number of File transfer sections, File transfer systems within each sig. Lightning speed I/O even with 2 users online. FULL CD detection without external hardware or software patches. Dedicated log window to see users progression online. Network conferencing - Go into conference mode with other RCIS systems Supports up to 32 lines (however system memory allows only 3), 32 channels. Optional doors package for up to 31 doors! Networked BBS listing program * UUCP package is currently under development. Tandy Express Ordering system (C) available from any system.

Future software updates are provided for free thru netmail Full type-ahead and control codes supported DECB type line editor in the message editor, and if you have DYNASPELL, the editor can utilize it for users Message editor supports foreground and background color changes as well as script variables, to allow you to write form type letters and questioneers with minimal typing, right through the message bases.

The system has been ported to an MM/1 and handles multiple users on the fly. RCRON: job scheduler included to maintain system files, LOGMAN to maintain userlog and over 15 other sysop utilities to make the software easy to use.

We believe that this is truly the best BBS system for OS9 Level 2.

A hard disk, 512K and a color RGB monitor is required.

For more information and pricing, call either (415) 883-0696, Color Galaxy Milky Way BBS (mine) or (201) 967-1061, RCIS Headquarters.

And of course you can write directly to me at:

-Eric Levinson
Color Galaxy Systems
1005 Green Oak Dr #16.
Novato, CA 94949

BULLETIN BOARD SYSTEMS FOR OS-9

I'd be almost "deaf, dumb and blind" if it weren't for the ability to "telecommunicate". Through this medium I can get in touch with people in different parts of the ccuntry, get news, or download the "latest" utility.

Aside from the "pay services" such as Delphi, Compuserve and GEnie, there are many, many privately run Bulletin Boards Systems (or BBS) that provide on a smaller scale what some of these larger systems have to offer.

Many of the newer generation BBSes also provide networking. This means these individual BBSes can now share a common message base and enable the sending of private E-Mail from system to system.

I felt an overview of the various BBS systems for OS-9 was needed. This may eventually help to clear up some misconceptions you may have had about a particular system and give you some insights into what they have to offer as well. This will be a continuing series with updates from the field.

Featured here are 4 of the more popular BBS packages for OS9 L2: AcBBS, RCIS, RiBBs and StG.

-Alan Sheltra

●●●●●

AcBBS

Early in 1990 I was looking to start my own bbs. I attempted to get several Rsdos packages going and either wasn't able to get them to work or didnt like the results I got from them. At about the same time I was doing this a friend of mine gave me a copy of an abandoned OS9 BBS package called AcBBS Version 2.2. At the time I knew Nothing about OS9 and was very skeptical about trying it. After a lot of thinking I said I would give it a try but he would have to teach me how to use OS9 and this software he had.

On April 9 1990 My BBS "The Dutchess CoCo" went online using this very bare-bones and bland software. After about 4 or 5 months of playing with OS9 I started playing with the source code that came with the package making small changes here and there. Thats all it took to hook me on programming in OS9. In December 1990 after making lots of changes and adding a lot of features we decided that others might also want this software. We then packaged it all up and Called it Version 2.3 and shipped it off to Delphi and some other OS9 support BBS's.

Boy did we ever learn alot about releasing software with 2.3 as we had working programs but the docs were a mess. Of all the people who setup that version there was only 1 person who was able to get it running without extensive help from us. We also learned with V2.3 that there were a lot of people out there that were not happy with the other OS9 BBS packages available at the time and thats when we decided to make upgrading and developing for AcBBS our main computer project.

Two 1/2 years later we have gone from a bland bare bones package with only 1 site running to a Full featured and constantly upgraded package that has over 20 sites running Coast to Coast. When I look back to that cold winter day when I was given the disks with AcBBS V2.2 on them and think I feel nothing other then pure amazement at what has happend since then.

Well enough with the past. Now to what AcBBS is now. AcBBS Version 2.4a should be released or very close to release by the time you read this Article. Some of the new features that are added since previous versions are a custom net-mail interface that connects all the AcBBS sites around the country together. The Net-Mail Automatic Transfers use the popular Zmodem transfer protocall to reduce long distance phone charges. The BBS also supports The Following user transfer protocalls; Xmodem, Ymodem, Ymodem-Batch, & Zmodem. Version 2.4a Also has a totaly rewritten message base handler that Supports features such as Quoting messages, Reading messages in threads, true word wrap,

NITROS9

OS9 Level II Expediter

6309 NATIVE MODE

Well, that's all for this month, see ya again next time with the rest of the chat program. Until then, if you have any questions or comments I can be contacted with the methods below.
- Andy DePue

US Mail : 24 Old Batson Road
Taylors, SC 29687
Internet: DRDUDE@delphi.com
StG Net : SysOp@Eagle
Delphi : DRDUDE
Phone : (803) 292-5140


```
*******************************************************************************
* Data Module for use with 'Chat' program.
* Copyright (C) 1992, by Andy DePue.
*
* This module is currently designed for two users only.
*

TypeLang equ $40    * Define Module Type as Data and Language as Data
AttRev   equ $81    * Module Attributes Set to Re-Entrant - Revision is 1
Edition  equ $01    * Module Edition is 1
Stack    equ $00    * Since This is a Data Module, No Stack is Required.

 psect ChatMod,TypeLang,AttRev,Edition,Stack,DataStrt

*******************************************************************************
* Actual Data of the Module
*
* Data Module is Arranged as Follows:
*
* Offset  Size            Description
* ------- --------------  ---------------------------
*  $0000  $0001           Number of Records in Data Module
*  $0001  $0073 * Records Records For Users (Chatting)
*

Records  equ $02    * Number of Records (2)

DataStrt  fcb  Records

UserRecs
     REPT Records   * Set up User Records Now

     fcb  $00       * Entry Usage Flag
     fdb  $0000     * User ID
     rzb  32        * RMA's RZB Statement is Used Here to Reserve a 32 Byte
*                   * User Name Buffer
     rzb  80        * This Will Create an 80 Byte Text Buffer

     ENDR
     endsect
```

```
/*
** Chat.h - Copyright (C) 1992 by Andy DePue.
** Header file for Chat.c
*/

/* Include header files */

#include <stdio.h>
#include <module.h> /* Includes structures for OS-9 Memory Modules */
```

everything up. (Note I used the xmode from the SACIA patch).

```
* Port Set Up Script
load /dd/cmds/login
onerr goto errtrap
echo [1] Bit Banger    - /t1
echo [2] Use Modem     - /t2
echo [3] Use RS232 Pak - /t3
prompt Enter Choice:
var.9
if $9=1
echo Starting Bit Banger Port for Terminal IO
xmode /t1 bau=1 pag=13 eko=1 upc=0 par=1
nice 25 /usr/cmds/tsmon /t1&
else
if $9=2
echo Starting Modem Port for Terminal IO
xmode /t2 bau=4 par=1 pag=13 eko=1 upc=0
nice 75 /dd/cmds/tsmon /t2&
else
if $9=3
echo Starting RS232 Port for Terminal IO
xmode /t3 bau=1 par=1 pag=13 eko=1 upc=0
nice 75 /dd/cmds/tsmon /t3&
else
echo "Invalid Selection"
endif
clrif
goto +finis
*errtrap
echo An error $* occured, check device status.
*finis
```

Once you've started this process load MikeyTerm or some other com package into your other computer, hit the return key and you will get a login prompt. Log in and you now have a multi-user system.

I'm constantly amazed with the power of this machine, at work I use a VAX 8200 under VMS, an AT&T under Xenix and (yech!) a Z-150 under (bigger yech!) MS-Dos. The CoCo compares with the VAX and the AT&T (a little slower), but really outshines the MS-Dog machine.

Since my nine year old son is the biggest user of my terminal setup I looked around for some simple games he could play under OS-9, but much to my chagrin nothing was available. Well in a past issue of Rainbow I saw a RS-Dos basic game simulating battleship called seawar. I uploaded this to OS-9 using the RS-Dos utilities I downloaded from Delphi. But RS Basic and Basic09 are not very compatible, so I was only able to use the bare basic's of this program. Once I finally got it to work on the main computer, I found MikeyTerm couldn't handle the graphics so I had to add an option to allow the user to just run battleship without graphics. I'd like to see some simple games and educational programs written for OS-9 like they have in RS-Basic. This would make it more enjoyable for my son to learn OS-9 and eventually get into something like UNIX which is going to be a standard for years to come.

Attached is the program listing I call BattleShip adapted and modified from the RS-Dos version called SEA WAR by Jeff Hameluck. Notice the major differences in structuring I was able to do under Basic09 that can't be done under RS Basic. [Note: The Battleship game will appear next month]

One final note: I've since downloaded the uucp tools and I have e-mail capability between my two machines now. Good luck setting up your Coco LAN.

-Chris Strickland
(New Horizons Software)

```
/* Setup Constants */

#define MODNAME     "ChatMcd"
#define USED        0xFF        /* $FF=Entry in Use */
#define EMPTY       0x00        /* $00=Entry Empty  */
#define DTYPE       04          /* Module Type for a Data Module */
#define LTYPE       00          /* Language Type */
#define DEFRECS     02          /* Default Number of Records */

/* Setup Data Module Header Structure (Returned From Link) */

mod_data  *modlink(),*modload(),*modptr;

/* Setup Data Module Structures (or "Templates") */

/* Structure for a User Entry */

typedef struct {
     char UsedFlag;              /* This Entry in Use? */
     int  UserID;                /* User ID for This Entry */
     char UserName[32],          /* User's Chat Name */
          TextBuf[80];           /* Text Buffer for This User */
} USER_ENTRY;

/* Structure for Entire Data Module */

typedef struct {
     char NumOfRecs;  /* Number of User Records (Entries) in This Module  */
     USER_ENTRY UI[DEFRECS]; /* Setup Two User Records (UI for User Info) */
} DATA_MODULE;

/* Now Setup "Template" for Data Module */

DATA_MODULE *mod;         /* Data Module Structure */
```

```
/*
** Chat.c - Copyright (C) 1992 by Andy DePue
** This program uses the Chat Module (ChatMod) to link two users together
** so that they may "chat" with each other.
**
** Note: The roots are here for implementing a larger than 2 user conference,
**       but as it is, the current data module architecure makes it impossible.
*/

#include "Chat.h"   /* Header file */
#include <lowio.h>
#include <sgstat.h>

#define  BACKSPACE  0x08 /* Back Space Character */

char *index();

int  yourecord;          /* User's user record # */
char stats[DEFRECS];     /* Previous Status of all User Records */
int  users;              /* Number of users in conference */
```


RS-232 Pak	RS-cable	
pin 2 data out	pin 2 data in	(green)
pin 3 data in	pin 4 data out	(white)
jump pins 4 & 5 (RTS,CTS)	none	
pin 7	pin 3 ground	(red)
jump pins 8 & 20 (CD,DTR)	pin 1 carrier detect	(yellow)

RTS - Request To Send, CTS - Clear To Send
CD - Carrier Detect, DTR - Data Terminal Ready

To connect to the RS-232 Pak you need a male RS-232 plug (Cat No. 276:1574).

The phone line cabling is what I really wanted; since, I could build two short cable that could be connected by phone lines, eliminating the bulky cabling normally required for cables longer than a few feet. The phone lines initially gave me a lot of trouble because the lines are switched as follows:

box 1	box 2
black	yellow
red	green
green	red
yellow	black

This made building the connector real fun. Your phone lines should be the same as what I've listed above unless the phone company is really trying to confuse us.

Now to make the cable, you'll need a male RS-232 connector with cover to look nice (Cat No. 276-1549B); a four pin male connector, just like the printer cable for the CoCo; and two Quick-Connect Modular Jacks (Cat No 279-355). All of the parts should be available at your local Radio Shack. Now connect everything up as follows:

RS-232 Pak	Box 1	Box 2	RS-cable	
pin 2 data out	green	red	pin 2 data in	(green)
pin 3 data in	red	green	pin 4 data out	(white)
jump pins 4 & 5	none	none	none	
pin 7	black	yellow	pin 3 ground	(red)
jump pins 8 & 20	yellow	black	pin 1 carrier detect	(yellow)

This will allow you now to connect the two computer with any length of phone line that you have available and makes for a neat set-up.

Just think, you have one of those small laptop computers, with a terminal package and a RS-232 port on it, you could connect phone jacks all through the house and just plug your cable with the RS-232 head and Modular jack into a phone line, plug the phone line into your CoCo jack in the wall and you have access to your main-"frame" CoCo anywhere in the house.

Keep your recipes in a database, put your old CoCo in the kitchen, plug it into the phone jack you installed in the kitchen, and you can access the main-"frame" CoCo. The only drawbacks I've noticed so far is you don't get graphics emulation or Multi-Vue from your terminals (maybe a good communications package could solve that, but MikeyTerm works great for my needs).

If you get lost with the connectors just build a simple continuity checker, I built one with a 330 ohm resistor, a LED, and about 15" of wire along with two 1 1/2 volt C batteries for power. Tape one end of the wire to the negative side of the battery, connect the other to the resistor side of the LED, touch the other side of the LED to your positive side of your batteries to make sure everything works. If it doesn't light, check your connections. If it still doesn't light then try turning your LED around.

Once I had the hardware built, all I had to build the pass-word file in /dd/sys, start the TSMON command with the LOGIN command in mem-ory, and away we go. I used the following shell script to set

```
      pathbf1.sg_echo=
      pathbf1.sg_pause=
      pathbf1.sg_eofch=
      pathbf1.sg_psch=
      pathbf1.sg_kbich=
      pathbf1.sg_kbach=0;
   }
   _ss_opt(1,&pathbf1);
   curoff();
   ospeed=pathbf2.sg_baud;
  /* set terminal speed for termcap */
}

resterml()
{
   if(pathbfv) _ss_opt(1,&pathbf2);
   curon();
   fflush(stdout);
}
```

Well, that's it for this time. Next month we'll write some more support code. If you have suggestions for future columns, want to take issue with me on something I've said, or just want to say hello, drop me at note at PO Box 355, Porthill, ID 83853 or PO Box 57, Wynndel, BC, Canada V0B 2N0 or Compuserve 76510,2203.

-Bob van der Poel

Tell a friend about The OS9 Underground

The Coco-Lan

by Chris Strickland

I have a 512k CoCo III, with a NAP monochrome monitor, 2 single sided drives, a multi-pak, a 30 meg harddisk, a Radio Shack RS-232 pak and a Radio Shack Speech Pak (recently deceased).

This is a long way from the original 4k old gray CoCo I, I started with. I remember having to cut the +12V lans on the circuit board so I could upgrade the old guy to 64k. In fact, I've performed surgery on the old geezer so many times that he lost his screws and the screw posts have velcro strips on them so I could just open and close the case whenever I wanted; you know for things like an LED so I'd remember to turn it off, a shift lock switch, a homemade composite monitor video driver (directions from Rainbow of course) and even once to fix it during his hippie years when he took up smoking; that was easy to fix, the blacken capacitor stood out like a good Cajun dish.

The old guy has since gone to computer heaven to be replaced by a second CoCo III (things just aren't the same).

Now let's leave the reminiscing and get to the point of this article, the CoCo Local Area Network (LAN). With OS-9 you can run extra terminals from your main CoCo and still work on it too (no dedicating one machine as a file server). To start with you need another terminal, I originally used my old CoCo I, the gray dinosaur. Now I use a newer Coco III with 80 columns.

To use the CoCo III needed a terminal emulation program, trying to be cheap so I didn't have to buy a disk controller to use my second drive with my CoCo III, I initially used CC Talk from the November 1984 issue of Rainbow, but switched to MikeyTerm so I could use 80 columns. I copied the MikeyTerm to cassette so I now had a cassette driven terminal package, all I needed now was a cable. I built two cables for the fun of it, one using 25 feet of speaker wire and the other using standard phone line and modular jacks. The cable for the speaker wire was built as follows:


```
/* Slash Commands */
char *commands[]={"NAME","WHO","TIME","EXIT","HELP","PAGE"};
int  comnum=6;                                    /* Number of commands */

long time(),startime;
char *ctime();
int  exitflag;          /* determines if user wants to exit */

/* main() - Links, Checks for Page, Adds User, Calls Chat, Unlinks, Exits */

main(argc,argv)
int argc;
char *argv[];
{
     int sighandler();

     if(getmodule(MODNAME)==-1)    /* Attempt Linking to ChatMod */
          /* If Attempt Fails, Inform User and Exit */
          error("Cannot Link to Chat Module: %s",MODNAME);
     intercept(sighandler);
     adduser();
     initstats();

     /* Check if user included device name on command line.  If so, */
     /* page device. */
     if(argc > 1)
          {
          if(page(argv[1])==-1)
               {
               /* This is to ensure that any users who saw you 'enter' */
               /* will see you leave as well.  If you deinit your entry*/
               /* too fast after initializing it, it may look like you */
               /* entered and never left. */
               tsleep(10);
               deinituser();
               errno=0;
               munlink(modptr);
               error("Exiting...");
               }
          }

     chat();
     /* Clear out User Record */
     deinituser();
     munlink(modptr);
     exit(0);
}

/*
** getmodule(name) - Attempts to link to data module.  If it fails, it attempts
**                   to load data module.  If it still fails, it returns an
**                   error.
*/

getmodule(name)
char *name;
{
     char *databegin;
```

```
     if((modptr=modlink(name,DTYPE,LTYPE))==-1)    /* Attempt Link */
          {
          /* If Link Failed, Attempt Load */
          if((modptr=modload(name,DTYPE,LTYPE))==-1)
               /* If Load Failed, Return Error */
               return(-1);
          }
     databegin=modptr;              /* Assign address of module header */
     databegin+=modptr->m_data;     /* Calculate address of data from offset */
     mod=databegin;
     return(0);                /* Return Non Error */
}

/*
** error() - Prints error message, reports error number, and exits.
*/

error(v1,v2,v3,v4,v5,v6,v7,v8,v9,v10)
{
     fprintf(stderr,"Chat, Error: ");
     fprintf(stderr,v1,v2,v3,v4,v5,v6,v7,v8,v9,10);
     fprintf(stderr,"\n");
     exit(errno);
}

/*
** adduser() - Searches through chat module for an empty record to add the
**             user to.  If an empty record exists, it adds the user and asks
**             for the name the user would like to go by.  If an empty record
**             does not exist, it reports the problem and exits the program.
*/

adduser()
{
     int  counter;
     int  flag;
     char username[32];

     /* Search through data module for an empty record */
     for(flag=FALSE,counter=0;counter < mod->NumOfRecs;counter++)
          {
          /* If empty user record found, set flag and exit */
          if(mod->UI[counter].UsedFlag==EMPTY)
               {
               flag=TRUE;
               break;
               }
          }
     /* If flag is not set, then report no empty user record, and exit */
     if(flag==FALSE)
          {
          errno=0;            /* No real error number goes with this one */
          munlink(modptr);    /* Make sure to unlink data mod before exiting */
          error("Cannot Find Empty User Record!");
          }
     /* If empty user record was found, set it up */
     /* Before anything else is done, set the used flag */
     mod->UI[counter].UsedFlag=USED;
```



```
putpad(s)
char *s;
{
   extern int write1();
 /* function to print 1 character */

   if(s) tputs(s,1,write1);
}

write1(c)
char c;
{
   putc(c,stdout);
}
```

Now we get the needed routines. First off is a routine to position the cursor.

```
gotoxy(x,y)
int x,y;
{
   putpad(tgoto(CM,x,y));
}
```

The next routines clear the entire screen, from the xy position to the end of the screen, and from the xy position to the end of the line.

```
clrscrn()
{
   if(CL) putpad(CL);
   else clreos(0,0), gotoxy(0,0);
}

clreos(x,y)
register int x,y;/* pass x/y coords */
{
   gotoxy(x,y);

   if(CD) putpad(CD);
   else{
      while(y<nrows){
         clreol(x,y++);
         x=0; /* start future lines */
              /* in col 0           */
      }
   }
}

clreol(x,y)
register int x,y;/* pass x/y coords */
{
   gotoxy(x,y);

   if(CE) putpad(CE);
   else while(x++<ncolumns)
      write1(' ');
}
```

Sometimes we want to ring the terminal bell to warn the user that he'd done something wrong.

```
bell()
{
   putpad(BL);
}
```

To finish things off, we have some routines to turn reverse video on and off and to turn the cursor on and off.

```
revon()
{
   putpad(SO);
}

revoff()
{
   putpad(SE);
}

curon()
{
   putpad(VE);
}

curoff()
{
   putpad(VI);
}
```

Since we will be doing interactive keyboard input, we need to set up the terminal in a known manner. Here we turn off echo and pause. We also disable the quit and abort keys. Again, we call this at the start of the program. Note the corresponding routine to restore things. This is called at the end of the menu program and before we do a shell command. After the shell command is finished, we have to set the terminal up again. We end up with two copies of the current path descriptor: pathbf2 has the original settings, pathbf1 has our modified settings.

```
static struct sgbuf pathbf1, pathbf2;
static char pathbfv=0;
/* flag, 1==path ok to reset */

setterml()
{
   if(!pathbfv){
      _gs_opt(1,&pathbf1);
      _gs_opt(1,&pathbf2);
      pathbfv++;
```

Reading the termcap data is prettystraight forward. This routine is called right after the program starts up. It determines the terminal type from the shell environment variable TERM and extracts the needed information. If you need more or fewer capabilities for your application just add/delete them from the list of tgetstr() calls.

"ce" capability is not defined. This method permits the program to run on just about any terminal.

The next two routines are needed for output. The first is used to write a string. It uses the library routine tputs() to do the proper translation. Note how we get tputs() to use our routine writel() to do the actual output. In this case, writel() just

```
init_term_cap()
{
   register char *term_type;
   char tcbuf[1024];
   char *ptr=tcapbuf;
   char *temp;

   if((term_type=getenv("TERM"))==0)
      terminate("Environment variable TERM not defined!");

   if(tgetent(tcbuf,term_type)<=0){
      terminate("Unknown terminal type '%s'!",term_type);
   }

   if(temp=tgetstr("pc",&ptr)) PC_=*temp;
   CL=tgetstr("cl",&ptr);
   CM=tgetstr("cm",&ptr);
   CE=tgetstr("ce",&ptr);
   CD=tgetstr("cd",&ptr);
   SO=tgetstr("so",&ptr);
   SE=tgetstr("se",&ptr);
   BL=tgetstr("bl",&ptr);
   VI=tgetstr("vi",&ptr);
   VE=tgetstr("ve",&ptr);

   if(ptr>=&tcapbuf[TCAPSLEN])
      terminate("Terminal description too big!");

   if(!CM) terminate("termcap entry needs cursor movement!");

   nrows=tgetnum("li");
   ncolumns=tgetnum("co");

   if(!nrows || !ncolumns)
      terminate("Unable to determine screen size");

   if(SO && SE) standout++;  /* signal rev. video avail */
}
```

Once things have been initialized, we are free to use the following routines. Note how certain routines handle not-so-intelligent terminals. For example, clrscrn() first tries to use the hardware "cl" routine. If it's not defined it uses clreos() to do the work. And clreos() uses clreol() if the uses the C buffered output routine; but it could use a custom output routine.

(Next Page)

```
     yourecord=counter;
     /* Blank out user name.  The other chat program will wait until */
     /* something has been placed into the user name before reporting that */
     /* a new user has entered chat. */
     mod->UI[yourecord].UserName[0]='\0';
     mod->UI[yourecord].UserID=getuid();
     mod->UI[yourecord].TextBuf[0]='\0';

     do
          {
          writeln(STDOUT,"What would you like to go by in this conference? ",
                  49);
          readln(STDIN,username,32);
          /* Replace CR with NULL to conform with C */
          *index(username,'\n')='\0';
          } while(username[0]=='\0'); /* Make sure user entered a name */

     /* Now copy it into thier data module record.  The reason I didn't */
     /* directly read the string into the data module is because about */
     /* 50% of the time, if there was another user in the conference, */
     /* thier program would print the user name that this program had */
     /* entered BEFORE it had a chance to replace the CR will a NULL. */
     strcpy(mod->UI[yourecord].UserName,username);
}

/*
** deinituser() - This will clear out the user's User Record for use by someone
**                else.
*/

deinituser()
{
     mod->UI[yourecord].UsedFlag=EMPTY;
}

/*
** initstats() - Initializes the global array 'stats' to the status of each
**               user record.
*/

initstats()
{
     int counter;

     for(users=0,counter=0;counter < mod->NumOfRecs;counter++)
          {
          /* If no user name entered, then just say unused entry for now. */
          /* That way, if a user is in the process of entering thier name, */
          /* this program will still print that they entered. */
          if(mod->UI[counter].UserName[0]=='\0')
               stats[counter]=EMPTY;
          else
               {
               stats[counter]=mod->UI[counter].UsedFlag;
               if(mod->UI[counter].UsedFlag==USED) users++;
               }
          }
}
```


Using the TERMCAP Library Part Three

by Bob van der Poel

This month we'll continue exploring of the termcap by developing our menu program. As we discussed in the first two segments, this program will use the termcap library for terminal displays. In previous installments we have shown you a sample menu data file and the common header file used by all the code modules.

This month we'll build the module which handles the reading of the termcap files. It also does the output for our program. I've interjected some text in between sections of code. Just remove the text and the remaining will be the complete program code.

The following section of code defines some variables used by termcap. Note how we use names very similar to the termcap capability name (the variable CM points to the termcap cm capability). These variables are global to the curses.c file--they have to be common to both the initilization and cursor routines.

FILE: curses.c

```
#include "menu.h"

#define TCAPSLEN  400

extern char **environ, *getenv();
```

This buffer is needed for the extracted termcap data. It has to be static so that calls to the output routines have access to the data.

```
static char tcapbuf[TCAPSLEN];
```

These variables are needed by the termcap library. They have to be defined somewhere in the program--it makes most sense to have them with the other termcap stuff. Note that we don't use the "static" modifier here--the library routines need to know about them.

```
char PC_,       /* pad character */
     *BC,       /* cursor backspace */
     *UP;       /* cursor up */

short ospeed;   /* terminal speed */
```

Termcap defines hundreds of different capabilities; we just need a few of them. Note how we use names similar to the actual capability's name. For example, CM points to the "cm" capability.

```
static char *CM,    /* cursor movement */
            *CL,    /* clear screen    */
            *CE,    /* clear to eol    */
            *CD,    /* clear to eos    */
            *SO,    /* standout start  */
            *SE,    /* standout end    */
            *BL,    /* sound bell      */
            *VE,    /* make cursor visible */
            *VI;    /* make cursor invisible */
```

```
* convert
* d6 is outer loop counter, d5 is inner
*
convrt lea temp(pc),a2
 move.b 7(a2),d6  exponent
 sub.b #$80,d6 remove bias
 neg.b d6
 add.b #55,d6 bits to be converted 55 minus 2's exponent

conv1 lea bcd(pc),a2 pointer to bcd
 move.b #8,d5 inner loop counter
 lsr (a2)
 roxr 2(a2)
 roxr 4(a2)
 roxr 6(a2)
conv2 move.b (a2),d7
 btst.b #7,d7
 beq.s conv3
 sub.b #$30,d7
conv3 btst.b #3,d7
 beq.s conv4
 sub.b #$03,d7
conv4 move.b d7,(a2)+
 sub.b #1,d5
 bne.s conv2

 lea temp(pc),a1
 lsr (a1)
 roxr 2(a1)
 roxr 4(a1)
 roxr 6(a1)
 btst #7,7(a1) have we shifted out a 1?
 beq.s noadd
 lea bcd(pc),a2
 add.b #$50,(a2)
noadd sub.b #1,d6
 bne conv1

* now convert from bcd to ascii inserting d.p. and minus sign
bcdasc lea bcd(pc),a1     a1 => bcd
 lea ascii(pc),a2         a2 => ascii
 lea 18(a2),a3            a3 => end of ascii buffer
 move.b #$20,(a2)
 btst.b #7,bsign(pc)
 beq.s bcda1
 move.b #$2d,(a2) - sign

bcda1 add.l #1,a2
 lea 8(a1),a4          a4 => tens exp
 move.b (a4),d7 tens exponent
 bgt.s bcda2
 move.b #$2e,(a2)+
 move.b (a4),d7 tens exp again
bcda01 beq.s bcda2
 move.b #'0',(a2)+
 cmp.l a2,a3
 beq.s bcdaex
 add.b #1,(a4) increment tens exp
 bra.s bcda01
bcda2 clr.w d7
 move.b (a1)+,d7 get two digits bcd
 asl.w #4,d7  0000ddddddddd0000
 lsr.b #4,d7  0000ddddd0000ddddd
 or.w #$3030,d7
 ror.w #8,d7
 move.b d7,(a2)+
 cmp.l a2,a3
 beq.s bcdaex
 sub.b #1,(a4)
 bne.s bcda3
 move.b #$2e,(a2)+
 cmp.l a2,a3
 beq.s bcdaex
bcda3 rol #8,d7
 move.b d7,(a2)+
 cmp.l a2,a3
 beq.s bcdaex
 sub.b #1,(a4)
 bne bcda2
 move.b #$2e,(a2)+
 cmp.l a2,a3
 bne bcda2
bcdaex rts
```



```
/*
** page() - This will page the specified device once.
*/
page(device)
char *device;
{
     int path;

     if((path=open(device,WRITE))==-1)
          {
          printf("\7Cannot Page '%s'!  Device Cannot be Opened!\n",device);
          return -1;
          }
     if(devtype(path)!=0)
          {
          printf("\7Cannot Page '%s'!  Wrong Device Type!\n",device);
          return -1;
          close(path);
          }
     printf("Paging...\n");
     writeln(path,"\7\7\7",3);
     writeln(path,mod->UI[yourecord].UserName,
                  strlen(mod->UI[yourecord].UserName));
     writeln(path," would like to talk with you!\n",80);
     writeln(path,"If you would like to speak to ",30);
     writeln(path,mod->UI[yourecord].UserName,
                  strlen(mod->UI[yourecord].UserName));
     writeln(path,",\n",80);
     writeln(path,"please run the program 'Chat'.\7\7\7\n",80);
     close(path);
}

/*
** chat() - initiates conference.
*/

chat()
{
     int  inputnum;       /* Number of bytes in 'inbuf' */
     char inbuf[80];      /* Buffer containing the text that the user has */
                          /* typed so far. */

     /* Set time user entered the conference */
     startime=time(0);
     /* Turn Echo Off */
     echo(STDIN,FALSE);
     printf("\7*** %s has just joined the conference.\n",
            mod->UI[yourecord].UserName);
     inputnum=0;
     exitflag=FALSE;
     /* Main Conference routine */
     while(!exitflag)
          {
          /* Check for any user input */
          if(checkinput(&inputnum,inbuf)==TRUE)
               /* If user pressed ENTER, process text */
               {
               parsetext(inputnum,inbuf);
               parsetext(inputnum,inbuf);
```

```
                inputnum=0;
                }
          /* Check for any text from other users */
          checkothers();
          }

     /* Turn Echo Back On */
     echo(STDIN,TRUE);
     printf("%s> - exiting -\n",mod->UI[yourecord].UserName);
}

/*
** checkinput() - This routine will get any input from the user and add it
**                to inbuf.
*/

checkinput(inputnum,inbuf)
int  *inputnum;
char inbuf[];
{
     char key;

     /* Check to see if the user text has exceeded screen bounderies */
     if((*inputnum)+strlen(mod->UI[yourecord].UserName)+2 > 78)
          {
          inbuf[(*inputnum)]='\0';
          writeln(STDOUT,"\n",80);
          return TRUE;
          }
     /* Check for user input */
     if(getstat(1,STDIN) == -1)
          return FALSE;

     /* Read in user input and process */
     read(STDIN,&key,1);
     switch(key)
          {
          case EOL:
               inbuf[(*inputnum)]='\0';
               writeln(STDOUT,"\n",80);
               return TRUE;
          case BACKSPACE:
               if(*inputnum < 1)
                    return FALSE;
               *inputnum = *inputnum - 1;
               write(STDOUT,"\x8 \x8",3);
               return FALSE;
          default:
               inbuf[*inputnum]=key;
               *inputnum = *inputnum + 1;
               writeln(STDOUT,&key,1);
               return FALSE;
          }
}
```


equivalent. The routine inserts the decimal point based on the value of the decimal exponent. If the decimal exponent is zero, the decimal point goes before the first digit. If it is 1, the dp goes after the first digit. If 3 it goes after the third digit etc. If the decimal exponent is -2, the routine outputs a decimal point and two zeros before the first BCD digit. This routine has no provision for outputting the number in scientific notation, though that wouldn't be hard to add. (See Listing Below)

Well, there it is. I think I've covered most of what might look mysterious.

I will be giving you an update of FPMUL that is slightly better than the original, but meanwhile let me give you a bug fix for the original code. The original worked fine multiplying 2*2, but it failed miserably multiplying 1/9 * 9. One of the partial products (one of the lowest order ones) was not properly added to the result. The missing line is just after the label skip1:

```
skip1 move.w 6(a1),
 mulu 8(a1),d7
 add.l d7,6(a5) *was missing first time
 bcc.s skip2
 add.w #1,4(a5)
```

The only really major improvement I've made in any of this was the FPDIV change that I presented last time.

-Ron Anderson

(Listing BINASC)

```
*
* binasc converts binary fp number
* pointed at by (a1) to bcd in bcd
* buffer
*
* then converts bcd to ascii in ASCI
* and returns with (a1) pointing at
* ascii string which is terminated
* with $04 for PSTRNG
*
* constants for conversions
*
zero  dc.l $00000000,$00000000
one   dc.l $40000000,$00000081
two   dc.l $40000000,$00000082
three dc.l $60000000,$00000082
four  dc.l $40000000,$00000083
five  dc.l $50000000,$00000083
six   dc.l $60000000,$00000083
seven dc.l $70000000,$00000083
eight dc.l $40000000,$00000084
nine  dc.l $48000000,$00000084
ten   dc.l $50000000,$00000084

* variables used by binasc
 temp ds.l 2  8 bytes
bcd ds.w 5   10 bytes
asci ds.w 9  18 bytes
ascend dc.b 4
bsign ds.b 1

binasc lea temp(pc),a2
 bsr tstzer   note a1 pointing at variable
 bne.s binal
 lea asci(pc),a1
 move.l #$20302E30,(a1)+ space
 move.l #$30303030,(a1)+
 move.l #$30303030,(a1)+
 move.l #$30303030,(a1)+
 move.l #$30300400,(a1)+ writes over bsign, no matter
 lea asci(pc),a1
 rts
binal move.l (a1)+,(a2)+     move binary to temp
 move.l (a1)+,(a2)+
 move.l #$FFFFFFFF,(a2)+
 move.l #$FFFFFFFF,(a2)+
 clr.w (a2)+

* save sign
 move.b temp(pc),d7
 and.b #$80,d7
 lea bsign(pc),a2
 move.b d7,(a2) save sign for later
 move.b temp(pc),d7
 and.b #$7f,d7
 lea temp(pc),a2
 move.b d7,(a2) clear sign
 ckex lea temp(pc),a1 point at number to be converted
 move.b 7(a1),d7 exponent
 sub.b #$80,d7 remove bias
 tst.b d7 set flags
 bgt.s divten divide by 10 if > 0
 cmp.b #$fd,d7
 bge.s convrt
 bsr ind move num to aestk
 lea ten(pc),a1
 bsr ind
 bsr FPMUL
 lea bcd(pc),a2
 sub.b #1,8(a2) decrement tens exponent
 lea -8(a5),a2
 lea temp(pc),a1
 move.l (a2),(a1)
 move.l 4(a2),4(a1)
 move.l a2,a5 remove result from stack
 bra ckex

divten bsr ind move number to aestk
 lea ten(pc),a1
 bsr ind
 bsr FPDIV
 lea bcd(pc),a2
 add.b #1,8(a2)
 lea -8(a5),a2
 lea temp(pc),a1
 move.l (a2),(a1)
 move.l 4(a2),4(a1)
 move.l a2,a5
 bra ckex
```

Result is .7500, again the correct answer. Binary 1/2 + 1/4 = 3/4 or 0.750

You might think of this process as adding or not adding 0.5 to the BCD value and then dividing that value by 2.0. The shift right and adjust of the BCD is basically a divide the BCD value by 2.0 procedure. You might see that we could come up with a similar algorithm for integers, i.e. add 1 to rightmost BCD, shift left, etc.

When you look at this, you might say the result grows rapidly toward the right. You are correct, and because of that, the process can yield slight errors. The reason is that "even" binary fractions are not "even" decimal fractions. 1/64 is 0.000001 as a binary fraction, but 0.015625 decimal.

The single 1 in the binary representation needs 5 decimal digits to represent it completely. If we start with a fixed number of BCD digits, the results generally eventually fall off the end of the BCD number and we throw them away. Perhaps later we can devise a way of rounding the BCD result by calculating a few extra BCD digits. For now we'll calculate 16 digits.

Now, on to the conversion. First we must reduce the binary number to a number greater than or equal to 0.1 and less than 1.0; BINASC tests the number for those limits.

If number is greater than 1.0 it is divided by 10.0 and the "decimal exponent" is incremented. Later, we shal see that the decimal exponent will be applied to the converted number. At any rate, the binary number is divided successively by 10 and the decimal exponent incremented until the binary number is less than 1.0.

If the initial number is less than 0.1 the number is multiplied by 10 repeatedly and the decimal exponent decremented until the result is greater than or equal to 0.1.

Now the resulting binary number has an exponent in our notation of $7D to $80. If the exponent is less than $80, (the number less than 0.5) the operation must go an extra BCD shift (which essentially divides the answer by 2). If the number is less than 0.25 there must be two extra shifts etc. The binary exponent is used to adjust the number of shifts. At that point we begin the conversion process as outlined above, shifting the binary and adding or not adding $5 to the hightest BCD digit. We then shift the BCD and adjust each half byte by subtracting 3 if the leftmost bit is a 1. When we're done, we have the BCD representation of the number and a ten's exponent.

The last part of the BINASC pulls apart the BCD number and converts each four bit BCD digit to its ASCII

The King is Mad!

by Paul Pollock

Long, long ago; there once was a kingdom, in a little known province; far to the west. This land was ruled by Good King Rudy, and Fair Queen Linda. Growing to pristine womanhood, the Crown Princess Ann, gave the royal parents every joy.

After some time had passed, it seemed clear that Princess Ann was ready to take a husband. And to no-one's surprise, she announced to her royal parents, that she had found a suitable mate.

The necessary bonifides already having been made, King Rudy made the marraige arrangements. There was to be a Royal Ball, and general shindig; to be attended by all who could make the trip to the capital city. All that was needed, was the RSVP returned by a certain date, and anyone could receive a formal invitation.

Well, it wasn't hard to figure, almost everyone who lived at the capital, wanted and received an invitation. And literally thousands of RSVP's were received from the outlying provinces, even other kingdoms!

oooooooooooooooooooooooooooooooooooooooo

The King, Queen and Pricess were overjoyed!

oooooooooooooooooooooooooooooooooooooooo

In an outlying suburb of the capital city, lived an old woman. Some said she was a witch, with evil powers, but few gave any credence to such stories. After all, Good King Rudy ruled, the land prospered, and the people were happy.

Yet, the old woman fretted. She mumbled to herself, as she walked the streets of her neighborhood. People could hear her talking in phrases that boded nothing good.

Yet, to the astonishment of the local constable, the old woman came up to him and handed him a small envelope; looked up at his face with a critical, cautious gaze, mumbled something unintelligable; and stumbled off in the direction she had come.

The constable, a wise and experienced fellow, thought nothing more could surprise him. Until he opened the envelope and read the contents. The old woman had written a simple RSVP, to the royal court, in an effort to receive an invitation to the wedding. But she had mistakenly given her letter to the constable, instead of the postman.

Well; as a kindness, the constable placed a postal stamp on the envelope, and put it in the postbox on the street corner. And promptly forgot all about it.

oooooooooooooooooooooooooooooooooooooooo

But he had done his civic duty, and it gave him a warm feeling in his heart.

oooooooooooooooooooooooooooooooooooooooo

To King Rudy's great joy, the marraige of his daughter to her beloved, went off without a hitch. The Royal Ball was a social triumph. Everyone, as far as Rudy knew, had come to the affair; and all were having a wonderful time.

But while everyone else was celebrating the Princess' good fortune, scurrying through the streets of the city, a dark shadow moved ever closer to the center of the community.

For here, the Royal Water Well, broached by a magnificent fountain; stood at the very heart of the capital city. The well itself, was the springhead for the water supply of the entire Kingdom. Indeed, the spring had never gone dry, or gone bad. And the water from it was a major reason for the health, prosperity and good spirits of the citizens of the Kingdom.

Ever closer, the old woman crept to the edge of the fountain. Even as she heard the joyous celebration at the palace, she mumbled to herself how she had been robbed of her chance.

For she had not received an invitation to the celebration, no-one knew why; but the old woman was going to get even for this terrible wrong, if it was the last thing she did! So she crept up to the lip of the fountain, pulled out a clay pot from her tattered robes; and poured the contents into the water!

A foul stench came from the pot,

difused into the fountain and disipated and seeped slowly into the well. The woman sped away from the well, disappearing forever more, into the night. And as she disappeared, the statue in the fountain glowed a sickly green, dimming finally as the woman dissappeared from view.

oooooooooooooooooooooooooooooo

The Deed was Done!

oooooooooooooooooooooooooooooo

The next morning, the people came to the well for their daily water supply; as they had always done. None noticed the subtly changed flavor of the water. For all were recovering from the best party seen in the Kingdom in living memory.

At first, and for many weeks thereafter, the Kingdom continued to thrive. But here, and sometimes there,... rumors of discontent reached the royal court. Eventually even to King Rudy.

In some places, people argued with city officials, and in other provinces, farmers argued with buyers (both public and royal). It was very strange. Most of the unrest was totally unfounded, yet in some cases violence resulted. No-one in the court could understand the trouble. Some thought a creeping madness had invaded the Kingdom.

oooooooooooooooooooooooooooooooooooooooooooooooo

There was talk, both in public and in the court, that sorcery was at work.

oooooooooooooooooooooooooooooooooooooooooooooooo

Finally one day, Good King Rudy and Princess Ann were having a conversation. And both had found it difficult to believe, that even Queen Linda had fallen prey to the creeping madness. King Rudy was at his wits end. Even some of his own court, were bickering for no reason, and a few had publicly demanded the removal of the King himself.

Princess Ann was dismayed, an puzzled. She had gone with Queen Linda to the city fountain, only a few day before. She had noticed that Queen Lind had quenched herself with a cool drin from it, and gained the favor of all wh saw her. For all loved the Queen. I seemed some things never changed.

Finally, the troubles slowly stopped. The fights cooled, disagreements became less common, until the land was once again the fair place it had always been.

And yet, the Royal Palace was still in an uproar. The Royal Family it seemed, was a divided house. Queen Linda constantly bickered with King Rudy. Princess Ann was constantly caught in the middle, not wishing to offend either of her royal parents. Her own husband completely powerless in the face of the growing rift.

As events unfold, solutions are always found. Mothers eventually settle with daughters. Even royal families find some solution. But one day, in the central courtyard of the palace, Queen Linda and Princess Ann were talking amongst themselves. Both had become increasingly concerned over King Rudy.

oooooooooooooooooooooooooooooooooooooooooooo

For He was still quite mad...
...And the seemed no cure.

oooooooooooooooooooooooooooooooooooooooooooo

Royal Court Law stated that the King would always drink from water kept in great kegs, below the palace. These kegs had been filled, years before, being kept fresh by arts held arcane and secret to all but a few in the royal court.

The Royal Water Supply seemed inexhaustable, and was meant to protect the King in times of trouble. Those who were in the know, were confident that these supplies were sufficient for many years. Perhaps King Rudy's entire remaining life.

And so King Rudy, forever more walked the parapets of his palace, quite alone. For he was mad, and no person could make head or tail of anything he had to say. For he alone knew the meaning of his own counsel.

His loving Queen Linda enventually died celebate, alone and embittered by past events. Princess Ann went on with her life; finally becoming the Royal Monarch of the Kingdom, after the death of her enmaddened father, Rudy.

Not another word was spoken,
about the once mad King.

BEGINNER'S NOTES
Decimal to Binary
by Ron Anderson

Last month's article finished the math functions in Assembler. Now in order to talk to our math routines, we need to be able to get from the decimal numbers we humans understand, to binary numbers, in the representation our math package needs. As I hinted previously, these routines are by no means simple and straightforward.

Our approach as usual will be to simplify the problem and start in the middle of the binary to decimal conversion process. However, first we need to talk about binary numbers and binary fractions. You all are most probably aware of how to evaluate a binary integer number, but we ought to start there. Just as in decimal, the digits from right to left represent 1's, 10's, hundreds, etc; powers of ten.

In binary the numbers represent 1, 2, 4, etc.; powers of 2. Notice I didn't say 1's, 2's etc. The binary number system allows only 0 or 1 in a given place. The zero denotes the absence of a particular power of 2 and the 1 denotes its presence. Binary representation of decimal 10:

```
1010 Binary number.
8421 Value associated with each place.
8020 Number represents an 8, no 4, a 2
and  no 1, which is 8 + 0 + 2 + 0, or
more simplly, 8 + 2, or 10.
```

Binary representation of decimal 30

```
 1   1   1   1   0    Binary number
16   8   4   2   1    Place values
2^4 2^3 2^2 2^1 2^0 Powers of 2
16 + 8 +  4 +  2  + 0 =  30
```

Now since we are in a binary numbering system we can't have a decimal point, but we can have a binary point. Since place in a binary number represents the value of a power of 2 and those powers decrease from left to right, we would expect the next digit to the right of the 1's to represent 2^(-1) power. 2^(-1) means 1/2 by definition 2^(-2) means 1/2^2 or 1/4. The places in a binary number with a binary point and fraction bits represents:

```
2^5  2^4  2^3  2^2  2^1  2^0  2^-1  2^-2  2^-3 ...
```

The value 1.5 decimal would simply be 1.1 in binary, that is 2^0 + 2^-1 or 1 + 0.5. Now don't go away, we're almost ready to convert a number from a binary fraction to a decimal fraction. First we have to talk about an idea called Binary Coded Decimal or BCD for short. The number 256 in binary is 100000000. In BCD it is: 0010 0101 0110. Each four bytes represents a decimal digit, whose value must be between decimal 0 and decimal 9.

The code above represents the digits 2, 5, and 6. It turns out that there is an algorithm (a process or set of rules) that can convert a binary fraction to a decimal fraction. The rules are simple.

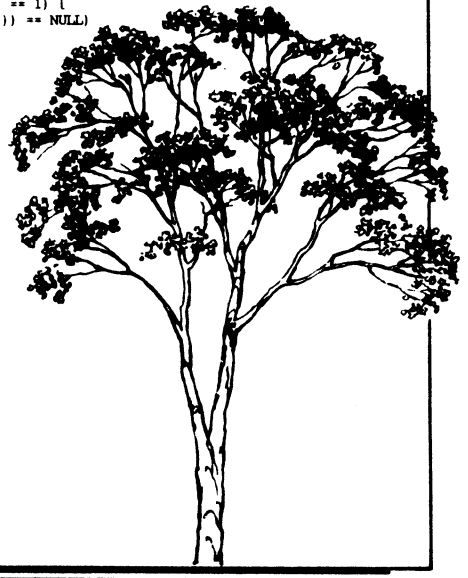
1. If the decimal point lies to the left of the first BCD digit.
2. Shift the binary number one place to the right.
3. If a '1' "falls off the end" (is shifted out), add 5 to the highest order digit of the BCD.
4. If there are no more binary digits goto 9
5. Shift the BCD number to the right one place.
6. If a 1 appears in the leftmost bit of any of the BCD digits subtract 3 from the value in that place.
7. Goto 2.

Let's see how these rules work:

```
Binary              BCD
 1                  0000 0000 0000 0000
  Shift right and a 1 falls off so we add 5 to the BCD
 0                  0101 0000 0000 0000
  The BCD value resulting is .5000 which is the correct answer.

Binary              BCD
11                  0000 0000 0000 0000
  Shift right and get a 1 so add 5 to the BCD
 1                  0101 0000 0000 0000
  Binary is not yet zero so we shift BCD right 1 place
 1                  0010 1000 0000 0000
  Now we subtract 3 from the second BCD digit (8 - 3 = 5)
 1                  0010 0101 0000 0000
  Shift binary again and get a 1, so add 5 to BCD (2 + 5 = 7)
 0                  0111 0101 0000 0000
  Binary is now zero so we're done.
```

```
int new_value;
ELEMENT *entry;
ELEMENT *root;
ELEMENT *ptr;
.
.
.
root = FALSE;
while(fread(&new_value,sizeof(new_value),1,stdin) == 1) {
   if((entry = (ELEMENT *) malloc(sizeof(ELEMENT))) == NULL)
      signal_memory_error;
   entry->value = new_value;
   entry->left = NULL;
   entry->right = NULL;
   if(root == NULL) {
      root = entry;
      entry->parent = NULL;
   }
   else {
      ptr = root;
      for(;;) {
         if(entry->value >= ptr->value) {
            if(ptr->right == NULL) {
               ptr->right = entry;
               entry->parent = ptr;
               break;
            }
            else
               ptr = ptr->right;
         }
         else {
            if(ptr->left == NULL) {
               ptr->left = entry;
               entry->parent = ptr;
               break;
            }
            else
               ptr = ptr->left;
         }
      }
   }
}
```



But What Does That Have to do with OS9...?

Computing is done by people who see the world, and their computing, very much from their own point of view. What one finds normal, another finds quite unreasonable.

How this affects people who use OS9 is more personal than many of us realize. While many people have opted to change to other systems and hardware, some of use have decided to remain where we are. Those that have changed, look upon us who have not, as holdouts; sometimes even insane.

But the real point of who is right, is much more rooted in things more tangible than 'newness' or access to software. Most people who have drifted away from OS9 have gone to systems that have access to software they don't purchase or use. Merely for the advantage of availability. It must be awful to be left out of a club because the members can merely obtain a thing, even when none actually have it.

And then there's the point of 'compatibility'. Few people really have a clear notion of what this actually means. For many, it returns the conversation to the area of obtainability of software. For others, it's the ability to translate software programming from one system to another. It's interesting to note that very few hardware/operating systems are actually moving programs via compatability methods. System authors are still doing an awful lot of 're-porting'. But more importantly, there seems a strong feeling that everyone should belong to one school of thought only. Try telling that to the millions of Apple-][or Commodore-64 users who NEVER cared whether their software was like packages used on other systems.

Few discussions bring to the floor points that matter more. The fact is, that for many, OS9 still serves its users. The hardware still operates fast enough to deal with programs in a useful manner. The programs used, are still efficient and serve in the capacities they were intended. The programming languages used are still functional and evolutionary.

Those folks that are still using the Color Computer-3 variety of OS9 Level-2 are using software facilities which still offer functions still not available or difficult to implement in systems that uses phrases like 'newer' or 'better'.

Coco Level-2 still has functions routed in a primative command structure, that lends itself to networks and communications standards that larger or supposedly more powerful systems still haven't nailed down yet in the data-system consumer mainstream. While the more arcane areas of data-systems are beginning to mature, the average 'hi-performance' system user not only doesn't have access to this maturity, but doesn't even know he is missing it. And therefore doesn't care.

OSK transplants also are coming up against this wall as well. While many are using these newer OS9/68000 hardware systems, many are finding that they have returned to operating standards they abandoned years before when they migrated from OS9/6809 Level-1 to Level-2.

While some functions are faster or have enhancements in the way of 'bells and whistles', most provide nothing in the way of increased functionality over OS9/6809 counterparts. Even programming languages (or parts of them) are not drastically more efficient, from an operator standpoint than obtained from Level-2 standards. Indeed, while the engine is normally 3-5 times faster, the increased storage size, or math requirements of the language force the program to not only be much larger, but tend to promote slower load and CRC-check times, and execution speeds.

For instance, BASIC in OSK, is functionally almost the same speed as Basic09 in OS9/6809, even when used on a 15mhz Signetics 68070 system! OSK BASIC provides much higher math precision, but in most cases this precision is not needed. The user would often have been better served if OSK BASIC were directly translated from the original Basic09.

Users report that they prefer many features of Basic09 itself over the working environment in BASIC. For instance, BASIC has no way of expanding its workspace once BASIC is executed. Memory storage must be called from the command point.

Worse yet, while 'C' programmers

variable called "parent". "What is it for?" you might ask. Just believe me when I say that it will be useful later and take it at that! Obviously, its purpose is to point to the parent of the current entry in the tree.

As with a standard linked list, memory to store an tree node would be a call like this:

```
ELEMENT *entry;

if((entry = (ELEMENT *) malloc(sizeof(ELEMENT))) == NULL)
   signal_memory_error;
```

Now, since we have defined that all values on the left sub-tree are less than the parent and all values on the right sub-tree are greater than the parent, it is useless to even discuss a general loading algorithm for a tree structure. Elements MUST be added to the tree such that this structure is ALWAYS maintained!! So, here is a code fragment to add elements to a this tree: (See Listing page 36)

Some of the braces are technically not needed, but I have found certain implementations of C that have problems in this regard, so sometimes I like to make sure by using some redunant braces.

Here's how this code fragment works. Here, since we are starting with an empty tree, we initialize the root pointer to NULL. The input loop looks identical to the input look in our previous discussions on linked lists, as does the memory allocation as well.

Notice, that after we move the new value to be inserted in the tree to the newly allocated tree element, we also set both branch pointers, left and right to a NULL. Think about it for a second. ANY time we add a new node to a tree, it is going to be a leaf, right? A leaf has no children, right? So, any new element to be added to a tree is going to be added with NULL branch pointers, right? RIGHT!

The next thing to do is to find out where in the tree this new element should go. First a quick check on the root, if it is NULL, this element is the first element to be placed in the tree and is therefore the initial root of the tree. If the root pointer is not NULL, we have a non-empty tree, we need to go hunting were to place the new element. We always start this search at the root of the tree, so we set the special pointer variable ptr to the root entry. The for(;;) loop is one way to code a "do forever" type of loop. But don't worry, your computer will not get hung in this loop, as you should see later.

The body of the loop only contains a single if() statement. If the element to add is greater or equal to the current entry we are looking at, then the element to be added needs to be added somewhere in this node's right sub-tree. So, we look at the right branch pointer. If it is NULL, then there is no right sub-tree and we have found where the new element should go. So we change this right branch pointer from NULL to point to this new entry and we exit the "do forever" loop with a break statement.

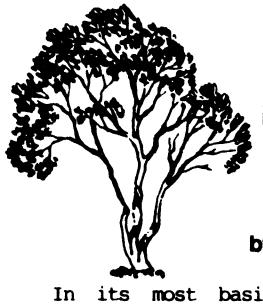
If the right branch pointer is not NULL, we set the variable pointer "ptr" to the right branch and we go through the loop again.

If the new element to be added is less than the current entry we are looking at, then the element needs to be added somewhere in this nodes left sub-tree. So, we look at the left branch pointer. As before, if the left branch pointer is NULL, we have found where we need to add the new entry. We set the left branch pointer to the new element and exit the loop with a break. If the left branch pointer is not NULL, we set the cariable pointer "ptr" to the left branch and we go through the loop again.

Note that in either adding the entry to a right branch or a left branch, the parent of the entry to be added is also set. Remember, we'll need this pointer to an entry's parent for later processing.

So, building a tree is not really all that difficult to understand, right? But, once we have built a tree, just how do we use it? I'll cover that in the next installment.

-Zack Sessions



Building a Binary Tree Part One

by Zack C. Sessions

In its most basic definition in graph theory, a tree is a connected graph with no cycles. A connected graph is a graph in which there is a path from any node in the graph to any other node in the graph. Put another way, there is always a path between any two arbitrary nodes in the graph. A path means that you can get from one node to another by travelling from node to node along the edges until you eventually reach the node you want to arrive at. A cycle in a graph is a path which starts and ends on the same node and never visits any node more than once. However, in a tree, there can be no cycles. This means that there is only ONE path from any node to any other node.

A tree is called a tree because it can be visually represented in such a manner as to physically represent a tree in nature, but it is usually drawn so that the tree is upside down. For this reason, certain nodes in the tree have special names based on their characteristics in respect to the rest of the nodes in the tree. For example, there is a special node in the tree called the "root node". It is important to realize that any node in a tree can be assigned the duty of being the root node without changing the relationship of any node in the tree with any other node.

Connections between nodes normally referred to as "edges" in general graphs are normally referred to as "branches" in a graph which is a tree. Some terms which relate to "family trees" are also applicable to trees. The nodes which are joined to a node above them are called "children" or "decendents" of the node above them. Similarly, the node above them which they come from is referred to as their "parent" or more generally their "ancestor". All nodes above a node are referred to as that nodes ancestors. Getting back to trees in nature, any node which does not have any children is referred to as a "leaf".

We will be discussing a special type of tree called a "binary tree". A binary tree is a tree in which each node has, at most, two children. A node does not have to have any children, or it may have one or two, but it can never have more than two. What this restriction allows us to do is to make a relationship between nodes. If each node in the tree represents the key to a data element in a list of data, we can say that the data represented by the node on the left branch will be less than the data of its parent node, and that the data represented by the node on the right branch is greater than the parent node. This makes a binary tree a particularly useful data structure in computer programming.

The term "sub-tree" is important as well. Each child of a node in a binary tree can be viewed as the root of a sub-tree. The sub-tree whose root is the child on the left branch is the left sub-tree and the sub-tree whose root is the child on the right branch is the right sub-tree. It should be obvious that EVERY data element in the left sub-tree will be less than the parent node, and EVERY data element in the right sub-tree will be greater than the parent.

Now, how do we go about representing a tree structure in a program? Well, each node will be an element in a special linked list. Each branch will be represented as a self-referential pointer, so since each node can have at most two children in a binary tree, each linked list element will have two pointers, one for the left branch and one for the right branch. Let's assume again, that the "data" for the list element will be a single numerical value, an integer. An element of the tree would be typedef'ed as follows:

```
typedef struct tree_def {
     int value;
     struct tree_def *parent;
     struct tree_def *left;
     struct tree_def *right;
} ELEMENT;
```

Now, you may notice, I have snuck in another self-referential pointer

have library extensions to take advantage of TERMCAP requirements in OSK, BASIC programmers have no facilities in BASIC to do the same; even though BASIC was extended to use higher precision math formats. Clearly Microware 'blew it' when tranlating the most powerful BASIC in the world to OSK, for the largest group of programmers in OS9: Basic Programmers.

Moreover, OSK has diverged in areas of graphic terminal standards, in ways that on the surface are very important; but in actual practice matter little or not at all.

The MM/1 is touting the 'K-Windows' package. While it yields the best overall speed of operation, there is still no Graphic User Interface for this package. While 'K-Windows' is the fastest graphic environment in OSK, it's still not drastically faster than the Color Computer-3 OS9 Level-2 environment. Most of this has to do with the larger math precision available in OSK, which yields no increased functionality in graphics, and serves to slow overall operations dramatically.

The other environment being published, 'G-Windows', while significantly more developed and more robust (and contains a terrific Graphic User Interface), is even slower than 'K-windows'. Some of this speed problem is due to its increased development level. Complexity tends to output as a reduction of efficientcy to the user. Another problem is that much of 'K-Windows' is written in Assembler, while 'G-Windows' is written in 'C'. No joke, Assembler is several times faster in execution than similar code in 'C', and is normally more compact, on a factor of 3-to-1! Even when running on systems that use a hardware video card for output, such as the Delmar System-IV. 'G-Windows' is so slow, that it tends to have trouble keeping up with similar operations done in 'MultiVue' in Level-2; even though it is signifcantly more powerful than any of its competitors.

While it might seem that I'm driving a conversational wedge between OS9/6809 and OS9/68000; the opposite is intended. The fact is, that while EVERYTHING in OS9 Level-2 is consistant to all OS9 terminal standards inclusive with it; neither 'K-Windows' or 'G-Windows' can be said to be compatible with any OS9 standards! It may appear that Color Computer OS9 Level-2 is a departure from previous Level-1 and Level-2 standards, the fact is that it merely 'extends' previous tables and methods.

In OSK, 'K-Windows' tries to emulate terminal standards intended for Color Computer Level-2, diverging signifcantly with TERMCAP standards; while 'G-Windows' has no facilities common to any standard for any system in OS9. Both establish standards which are irreconcilable with any 'norm' intended by the authors at Microware.

What does this mean to us? It means that we all have a choice. We can get mad at each other until we either get disgusted and switch to one of the deadhorse standardized systems that the world and Microsoft wants us to use and believe in; or some kind of miracle event results in standardized methods be invented to communicate with all OS9 systems.

Owners of OS9 systems which differ from their fellows CAN learn to absolve themselves from responsibility for systems used by other people, and just go their own way. Just as Apple-][and Commodore-64 users did.

In theory; this is fairly simple, just ignore everybody else. In practice, this will mean that users will find themselves writing alot of their own software. And a massive search will go on, for users with similar systems to contact each other. The only people to profit by this activity early on, will be the phone company and the postal service. Some of this is already occuring.

I predict that users will choose somewhere in-between. In the process, no one will be properly served. And OS9 will continue to stay unmatured. We must remember that the majority of OS9 systems have no graphics capable terminals at all. The Color Computer started an interesting yet difficult to reconcile new trend. It may serve the entire community, to relegate the graphics systems to a 'special' nitch that gets separate levels of support.

Each of us is the King of his own land. If we're not careful, we will find that WE ARE ALL MAD!

-Paul Pollock (Technical Editor)

TECH CORNER

A BETTER PRINTER UTILITY

by J. Scott Kasten

This month, I have a better printer utility to replace the one from the one shown in the last issue. It was rather slow since it only printed with 4 of the 8 pins available in the printer graphics mode selected.

To print 8 bit graphics data, one needs to generate characters in the range 0 to 255. This wouldn't seem like much of a problem, but turns out to be a real hassle. The problem is that your printer, being a sequential character device, will be attached to the system with a driver operating under the SCF manager; either of which may translate certain characters with ASCII values below 32.

Try using the command "xmode /term" and look in the OS9 operating system utilities manual to interpret what you see. One of the uses for character translation is to make OS9 compatible with popular peripherals like printers and terminals. Most peripherals for example, use a two character sequence to delimit a line of text - <CR><LF>.

On the other hand, the OS9 file system uses a lone <CR>. The lf/nolf setting for SCF devices allows for the insertion of an <LF> character after a <CR> is spotted, if your device needs it.

This type of character translation is usualy handy since it works transparently; but it should be obvious how disaster will strike if your bit image graphics get translated on the way to the printer! Previously, I avoided the problem with a quick fix that made sure none of the questionable characters ever got sent out.

Fortunately, OS9 allows for a number of ways to dynamicly activate and deactivate the character translation. The approach I took here was to get the options table for the device to determine if it needed linefeeds or not (the lf/nolf param); I then deactivated character translation on that path. At the end of each print line, I send a CR; if LF is also required (as determined by the options table) for proper paper advancement, then it is sent explicitly as well.

As you edit the original source code, be careful to spot all of the changes, some of them are rather subtle! Unfortunately, there is a lot of explanation that goes into how this program works and why I did what I did to make it work. Due to the lack of press of time and space limitations, that info will have to wait until next time. Until then, happy graphics printing! (Listing on Page 29)

Please send questions and suggestions to OS9 Underground or directly to myself: J. Scott Kasten / 307 Whispering Oaks Dr. / Bethalto, IL 62010-1039


```
            number_of_files:=number_of_files+1
            file_name2(counter):=dir_entries(counter).file_name
            RUN fix_file_name(file_name2(counter))
            fds(counter,1):=dir_entries(counter).file_desc_sect(1)
            fds(counter,2):=dir_entries(counter).file_desc_sect(2)
            fds(counter,3):=dir_entries(counter).file_desc_sect(3)

        ENDIF
NEXT counter

(* ...*-end of procedure-*...current file name info is now stored
(* .........................in the file_name2 array and the file
(* .........................descriptor sector info is now stored
(* .........................in the fdc array.
(* Let's print the directory - might as well make use of this info!

PRINT
PRINT "2 collumn directory program by Jim Vestal:"
FOR counter:=1 TO number_of_files
    (* print every other file entry at a different tab setting
    IF counter/2.=INT(counter/2.) THEN
        PRINT TAB(41); file_name2(counter); TAB(71);
    ELSE
        PRINT
        PRINT TAB(1); file_name2(counter); TAB(31);
    ENDIF
    (* print file description sector info in hex format *)
    PRINT USING "h2",fds(counter,1);
    PRINT USING "h2",fds(counter,2);
    PRINT USING "h2",fds(counter,3);
NEXT counter
PRINT
PRINT
(* this program is done, any questions let me know
END

PROCEDURE fix_file_name
(* procedure called from read_directory to fix the last
(* charactor of the file name, (unset the 8th bit on
(* that charactor).

PARAM file_name:STRING \(* file name info passed from read_directory

DIM char_count:INTEGER \(* position of last letter of the file name

char_count:=0

REPEAT
    char_count:=char_count+1
UNTIL MID$(file_name,char_count,1)>CHR$(127) OR char_count=29
    (* last charactor of file name located!

file_name:=LEFT$(file_name,char_count-1)+CHR$(ASC(MID$(file_name,
    char_count,1))-128)

(* guess what?  this procedure is done... file name is now fixed!

END  \(* this end statement is not needed, but it looks good! *)
```

```
DIM file_name2(100):STRING; fds(100,3):BYTE \(* 3rd DIM
(* -----------^^^--------------^^^---------------------
DIM path:BYTE \(* path of disk file

DIM counter,total,number_of_files:INTEGER
DIM seek_counter:INTEGER \(* used in the seek ommand

(* initialize arrays and variables

FOR counter:=1 TO max

    dir_entries(counter).file_name:=""
    dir_entries(counter).file_desc_sect(1)=0
    dir_entries(counter).file_desc_sect(2)=0
    dir_entries(counter).file_desc_sect(3)=0
    file_name2(counter):=""
    fds(counter,1):=0
    fds(counter,2):=0
    fds(counter,3):=0

NEXT counter

counter:=0 \(* general purpose loop counter
total:=0 \(* total number of file entries in directory file
number_of_files:=0 \(* number of existing files in directory

(* open path to current directory for read access

OPEN #path,".":READ+DIR

(* seek past first 2 directory entries . and ..
(* pointer now located at 3rd filename entry

seek_counter:=SIZE(dir_entries)/max*2
SEEK#path,seek_counter

(* read directory information into dir_entries array

counter:=1
WHILE NOT(EOF(#path)) AND counter<=max DO

        GET #path,dir_entries(counter)
        counter:=counter+1

ENDWHILE

CLOSE #path \(* don't forget to close the file!

(* convert file names into proper string format

total:=counter-1 \(* total number of file entries in the directory

FOR counter:=1 TO total

    (* check for only valid file names that have not been deleted

    IF LEFT$(dir_entries(counter).file_name,1)<>CHR$(0) THEN
```



```
/***** Revised gprint.c program. *****/

#include <stdio.h>
#include <strings.h>
#include <sgstat.h>

#define WIDTH   640   /* Width of file in pixels. */
#define LENGTH  480   /* Length of file in data lines. */
#define NPINS   8     /* Number of pins to use on printer. ( 0 < x <8) */
#define CR      13    /* ASCII carriage return. */
#define LF      10    /* ASCII line feed. */
#define FF      12    /* ASCII form feed. */

main (argc, argv)
int argc;
char    *argv[];
{
    FILE    *infp,*outfp;  /* file pointers */
    int     fc;            /* file counter */
    int     i,j,k,l;       /* position loop indecies */
    int     color;         /* computed pixel color value */
    char    line[WIDTH];   /* array to store current printer line */
    struct sgbuf    stats; /* structure to hold output options table */

    if (argc == 1) {
        fprintf (stderr,"\ngprint usage:\n\n");
        fprintf (stderr,"gprint {<file>}\n\n");
        return 1;
    }

    if ((outfp=fopen("/p","w")) == NULL) {
        fprintf(stderr,"gprint: can't access print device!\n");
        return 1;
    }

    _gs_opt(fileno(outfp),&stats);  /* Get device options table. */
    outfp->_flag |= _RBF;   /* Turn off output character filtering. */

/* File handler loop. */

    for (fc=1; fc < argc; fc++) {
        if ((infp=fopen(argv[fc],"r")) == NULL) {
            fprintf(stderr,"gprint: can't open file '%s'.\n",argv[fc]);
            return 1;
        }

/* Print page header. */

        putc(27,outfp);
        putc('W',outfp);
        putc('1',outfp);
        putc(27,outfp);
        putc('w',outfp);
        putc('1',outfp);
        for (i=0; i<((40-strlen(argv[fc]))/2); i++) putc(' ',outfp);
        fprintf(outfp,"%s\n",argv[fc]);
        if (stats.sg_alf != (char) 0) putc(LF,outfp); /* Add lf if needed.*/
        putc(27,outfp);
        putc('W',outfp);
        putc('0',outfp);
```

```
        putc(27,outfp);
        putc('w',outfp);
        putc('0',outfp);
        for (i=0; i<7; i++) putc(LF,outfp);

/* Set printer line spacing appropriate for graphics. */

        putc(27,outfp);
        putc('3',outfp);
        putc((char) (NPINS*3),outfp);

/* Main print loop. */

        for (i=0; i<LENGTH; i+=NPINS) {
            for (j=0; j<WIDTH; line[j++]=0);
            for (k=i; (k<LENGTH) && (k < (i+NPINS)); k++) {
                l= 128 >> (k % NPINS);
                for (j=0; j<WIDTH; j++) {
                    color = getc(infp) % 2;
                    line[j]+=l*color;
                }
            }

/* Set printer into graphics mode. */

            fprintf(outfp,"     "); /* Move line over 5 spaces. */
            putc(27,outfp);
            putc('*',outfp);
            putc(6,outfp);
            putc((unsigned char) (WIDTH % 256),outfp);
            putc((unsigned char) (WIDTH/256),outfp);

/* Print out the line. */

            for (j=0; j<WIDTH; putc(line[j++],outfp));
            putc(CR,outfp);
            if (stats.sg_alf != (char) 0) putc(LF,outfp); /* Add lf if needed.
        }

        fclose (infp);

/* Reset line spacing to normal and spit out printed page. */

        putc(FF,outfp); /* Page eject. */

        putc(27,outfp);
        putc('3',outfp);
        putc(36,outfp);
    }

    fclose (outfp);
}
/***** End prog. *****/
```


BASIC TRAINING

By Jim Vestal

Welcome to the third month of my column Basic Training. This month I will present a complete Basic09 program instead of the normal discourse on structured programming. Notice that the program was written using the techniques of structured programming that I have been writing about the last 2 months.

"Read_Directory" was written by myself with support from my friend Tim Mohr.

"Read_Directory" is a stand-alone 2 column directory program written purely in Basic09. It can easily be converted into a subroutine type of procedure callable in any basic09 program to read directory information from the current data directory. For further details see the comments within the basic09 source code.

I wrote this program for the mere fun of it, knowing that other programs read the directory info directly from the directory file on the disk. Tim needed a subroutine such as this for one of his projects so I decided to write my own routine for his and my future projects. I'm currently including this program as a subroutine in my menu driven text-file viewer (bulletin) program that will be installed in my bbs menu.

Basically the program is self documenting, the directory file contains records of 32 bytes that contain the filename (8th bit set on the last charactor of the filename), and 3 bytes for the logical sector number of the file desciption sector on the disk for that file. Each directory is readable as if it were a regular random access file by using basic09's READ+DIR option. So reading the informaton into an array is rather easy.

I hope you make use of this routine, or at least use it to learn a bit on how easy it is to program in basic09.

If you have any questions feel free to contact me.

Next month I will continue the discourse of structured programming and introduce BASIC programmers to the C programming language.

-Jim Vestal

```
PROCEDURE Read_Directory
(* read directory procedure by jim vestal and tim mohr

(* released into the public domain, feel free to use or modify
(* this program, please don't abuse it though
(* ...*-start of procedure-*...to convert this into a procedure
(* ............callable by any program as a subroutine just replace
(* ............the 3rd dim statement with param statement include
(* ............all program statements following these remarks until
(* ............the *-end of procedure-* remark in the called
(* ............procedure and the arrays mentioned in the calling
(* ............program.

(* sets lowest array index to 1, 'cause humans count starting at 1

BASE 1

(* declare data structure and variables

TYPE dir_entry=file_name:STRING[29]; file_desc_sect(3):BYTE
DIM max:INTEGER \(* maximum number of files expected-modify to suit
max:=100 \(* value of max should match the array value--
DIM dir_entries(100):dir_entry \(* raw data stored here
(* ------------^^^ ---------------------------------
```

AniMajik Productions

NEW!
CLOUD_09 - by Albert P. Marsh

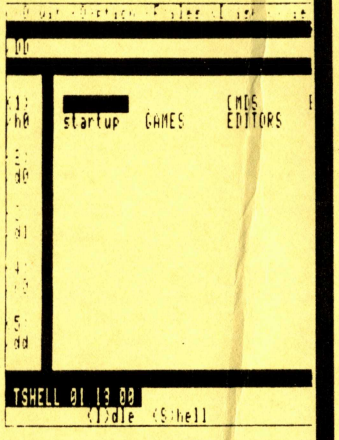
The BEST Graphics/Animation Editor for the CoCo! Tools include: Line, Box, Ellipse, Fill, Pencil, Brush, Flow, Spray, Text, FatBlox, Palette. Work with up to 8 animation pages. Copy one page to another. Complete control of animation speed. Edit/Save/Load VEF picture files.

Req: OS9 Level 2, Multi-Vue and 512K 34.95

TShell V3.13.02 - by Paul Pollock

A revolutionary New Program... "TShell" does most of what Multi-Vue does at up to 5 times the speed! TShell will run most programs with one keypress and use standard MV AIF files. Delete, Copy, Rename files all with 1 or 2 keystrokes! Many utilities included.

WINDINT and Multi-Vue NOT required!
Req: OS9 Level 2 512K



TShell (with printed manual) $39.95

TShell (with "printer-ready" manual, TDoc included to make the job easy!)
You print it... you save! $29.95

SunDialer "WarGames" Dialer - by John Powers
(Includes versions for ACIA and SACIA)
Req: OS9 Level 2 and 512K $19.95

DCom - Basic09 Decompiler - by Wayne Campbell
Req: OS9 Level 2, 512K $24.95

Coming Soon!

TAKENOTE
COCO TYCOON V2.0
COCODEX
MEMMATCH
StG Net V4

(Prices Subject to Change without Notice)

Send Checks or M.O.'s

4650 Cahuenga Blvd. Ste #7
Toluca Lake, Ca. 91602

(818)
761-4135

AWESOME BOOTFILE EDITOR!

KWIKGEN v1.01

Still using OS9Gen, Cobbler, or Config?
Get a real bootfile editor!

EzGen v1.09	*vs.*	**KwikGen v1.01**
5 minutes 40 sec.		44 SECONDS!*

* Identical operations performed on identical fragmented boot disks - 2 deletes and one insert performed by both utilities

- Editing done in memory
- Load boot from disk or memory
- Patch modules
- Change order of modules in seconds
- 100% assembly code
- Make multiple boot disks in one session
- Edit existing boot files in place easily
- Load kernel from disk or mem. and write to disk

KwikGen requires OS9 Level I, or II. ***$24.95***

KWIKZAP v1.1

- display updating is instantaneous
- 'smart' verify command
- work on file or stack
- searching functions
- 100% assembly code
- configurable environment
- dynamic sector stack
- allows editing of nibbles or half bytes
- built in help - easy to use

KwikZap requires OS9 Level II. ***$24.95***

Experience GALE FORCE speed!

Checks: Allow 4 - 6 weeks for delivery.
Money orders: processed immediately for KWIK delivery.

Shipping and handling is $4.00.
Call or write for our free catalogue.
Please call for Canadian prices.

Send check or money order to :

Gale Force Enterprises
P.O. Box 66036 Station 'F', Vancouver,
B.C., Canada, V5N 5L4

(604) 589-1660
8 AM - 5 PM PST (voice)
5PM - 8 AM (support BBS)

1992 First Annual Last CocoFest Report

(May 30-31, Chicago, Illinois)

by Allen Huffman

Just when it looked like there would be no more Color Computer Festivals, Dave Myers of CoCoPro! decided to sponsor a Chicago CoCoFest. Billed as the "First Annual Last CoCoFest", we at least had one more shot at a grand scale gathering of CoCo supporters - and grand scale it was with the likes of Marty Goodman, Steve Bjork, Kevin Darling, Eddie Kuns, and many other CoCo notables in attendance. The hosting club was Glenside, which previously hosted the Chicago Rainbowfests.

To keep costs down, the event was held at the Northlake Meeting Center which was several blocks down the road from the Clubhouse Inn where attendees were staying. A shuttle bus was available for those who didn't feel like walking to the show.

Glenside Coco Club - These guys had one of the most impressive looking booths.

THE PARTY IS ON

Friday night kicked off with the "Party with Marty". A small admission fee ($6) allowed entrance. This "party" actually consited of nothing more than dozens of CoCo owners sitting around tables snacking on junk food, supplied by Glenside. To liven things up, a Midi rig was hooked up playing UltiMuse tunes. Overall, nothing too exciting happend though it did give everyone a chance to rub shoulders with Marty.

-Alan Huffman
(Sub-Etha Software)

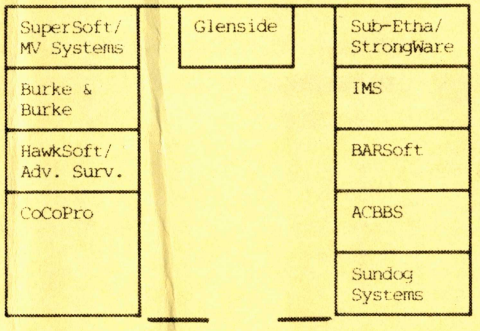
Kevin Darling, Joel Hegberg and Allen Huffman stop to chat.

FEST PHOTOGRAPHER

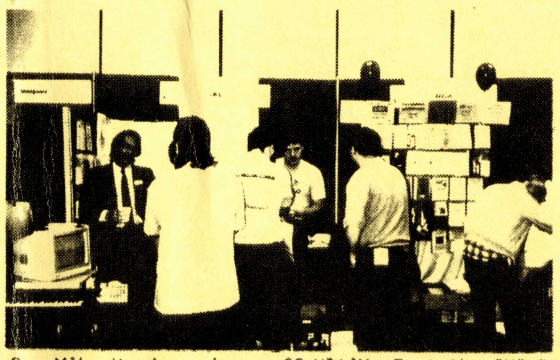
Pictures are by our roving reporter, Allen Huffman (of Sub-Etha Software). Due to space considerations, this article will continue in next month's issue, giving a detailed account of the vendors and wares at the May '92 Coco Fest. -Ed.

Proud owner of a Coco with serial #0001

The SUB-Etha/StrongWare Gang



Steve Bjork poses with our Roving Reporter. Steve gave a seminar on 12 years of the Coco.



Dr. Mike Knudsen shows off UltiMusE at the IMS Booth

SunDog Systems even gets Steve Bjork's attention